CHALMERS

Automatic bridge detection in airborne laser scanned data

Master of Science Thesis

DANIEL FORSBERG

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, September 2011

Automatic bridge detection in airborne laser scanned data

DANIEL FORSBERG

Examiner: GRAHAM KEMP

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden September 2011

Abstract

Collecting information about objects or areas without being in physical contact with them is referred to as remote sensing. Several techniques for acquisition of remote sensing data are available, one of the most common techniques used today is airborne laser scanning where a laser mounted on an airplane or helicopter is used to acquire remote sensing data. The remote sensing data can for example be used to create digital terrain models used in hydrological modelling. Objects like bridges can potentially disturb hydrological modelling, therefore bridges should be removed from such terrain models in order to to provide accurate hydrological models. This thesis describes a method for removing bridges from remote sensing data acquired by airborne laser scanning. The method is shown to be able to detect bridges of varying sizes in several types of landscapes while maintaining a high detection rate.

Acknowledgements

This project was carried out at Foran Remote Sensing AB, a company located in Linköping focused on providing advanced processing and analysis of remote sensing data. I would like to especially thank my supervisors, Dr. Ulf Söderman at Foran Remote Sensing AB and Graham Kemp at Chalmers for their help and support. I would also like to thank the employees of Foran Remote Sensing AB for all their help during the project.

Daniel Forsberg, Linköping September 25, 2011

List of Figures

Contents

Chapter 1

Introduction

The demand of digital models of the real world is increasing as the ability to analyse and use them for simulations increase. Today creation and usage of accurate digital models have come to play an important role in areas such as forest analysis, flood simulation, navigation and urban planning.

One of the most popular methods to acquire the data necessary to digitalise a landscape, is by using airborne laser scanning. An airborne laser scanner system is capable of registering a large number of points in a landscape together with 3D coordinates of each point. By combining the registered points, a 3D point cloud representing the landscape in a scanned area can be created.

A point cloud captured by airborne laser scanning can be analysed in several ways depending on the intended usage, for example it can be rasterised into an image and analysed by image processing algorithms. Algorithms can also work directly on the irregularly sampled point cloud and use the topology of points to draw conclusions about the bare earth, buildings and vegetation in the landscape. One of the most common objectives of algorithms working on the point clouds is to find a classification of each point where the classification of each point can be for example ground, water, building or vegetation. The classification can then be used for instance to create digital terrain models or to digitally recreate the landscape.

1.1 Outline

This report is divided into the four chapters: 'Introduction', 'Method', 'Results and discussion' and 'Conclusion'.

Chapter 1, 'Introduction', provides a history of airborne laser scanning and presents some applications of airborne laser scanned data as well as a motivation of this project.

The second chapter, 'Method', contains a description and motivation of the algorithms

used to detect and classify bridges. Chapter 2 also describes the steps necessary for the algorithms and gives a brief explanation of the technology behind airborne laser scanning.

Chapter 3, 'Results and discussion', presents the classification results of the implemented algorithm together with a discussion of the strengths and weaknesses of the algorithm.

The last chapter, 'Conclusion', contains a summarising conclusion of the work presented in this project and suggests possible extensions of the project.

1.2 Background

Airborne laser scanning dates back to the 1970s when it was introduced under the name Airborne Profile Recorders (APRs). The APRs could record precise range measurements of the distance from an airborne laser scanner to a point in the landscape. However at the time no technology was available for precise determination of an aircraft's position, and as a result the created point clouds lacked precision. When commercial GPS positioning was made available in the 1990s the development of airborne laser scanners accelerated. According to Baltsavias [1] there was only one company providing commercial airborne laser scanning systems in 1996 but already two years later the number of companies providing airborne laser scanning services had risen to around 40.

Since then airborne laser scanning and algorithms working on the collected data have become increasingly important in several areas, for example as shown by Maas et al. in [2] it is possible to detect and digitally recreate buildings. The created models can, for instance, be used to visualize how new structures, buildings or infrastructure would fit into an existing landscape.

Another area where airborne laser scanning is becoming increasingly important is in the forest industry where it can be used to detect individual trees and their properties as described by Hyyppä et al. in [3]. As forests cover large areas airborne laser scanning can greatly reduce the time and workload needed for large scale forest analysis.

One of the most common applications of airborne laser scanner data is extraction of digital terrain models, a terrain model is a digital model of a landscapes terrain with all objects like buildings and vegetation removed. As shown by Kraus et al. [4] airborne laser scanners can provide accurate terrain models even in forested areas where forest canopies shadow the bare earth, something that cannot be accomplished by photogrammetric methods. The created terrain models can for example be used for hydrological modelling and to produce height models for cartography.

The use of digital terrain models in hydrological simulations is closely related to the work presented in this thesis. As described by Marks et al. [5] digital terrain models acquired from airborne laser scanning can be used to simulate flood models along flood-

plains. Bridges appear frequently along floodplains and because many algorithms for extracting digital terrain models do not handle bridges explicitly, bridges can appear in the digital terrain models and disturb flow patterns during flood simulations.

1.3 Purpose

As mentioned in the previous section digital terrain models can be used in flood simulations where they can be used to identify water routes and flow patterns. To create the ground model used in the simulations a ground classification algorithm is applied to the input point cloud and a continuous ground model is derived by triangulating and interpolating ground points.

In many applications of digital terrain models bridges do not play an important role, as a result current ground classification algorithms typically do not treat bridges explicitly. Bridges typically exhibit properties typical for both ground and other objects, in one direction, along the bridge, it connects smoothly to the bare earth which is typical for ground points. Across the bridge on the other hand the bridge is discontinuous to the ground below which is a property typical for buildings and vegetation. These properties of bridges means that not treating bridges explicitly can result in some parts of bridges being removed from the terrain model while other parts are included thus potentially disturbing flood simulations.

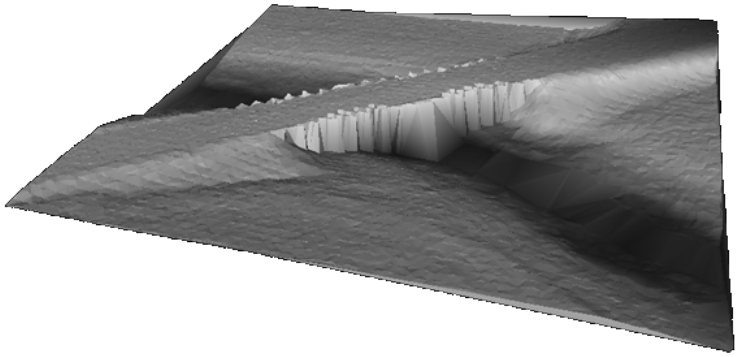
When simulating flood paths, bridges appear in terrain models as dams, by applying a bridge detection algorithm it is possible to find and remove bridge points from a point cloud therefore enabling accurate flood simulations. In Figure 1.1 two different terrain models are shown. Both models are derived from elevation data of points classified as ground points in the shown area. Figure 1.1(a) shows the resulting ground model after a conventional ground classification algorithm that give no explicit treatment to bridges has been applied, note how the bridge appears as a dam in the ground model thereby skewing potential flood simulation results. In Figure 1.1(b) however a bridge removal algorithm has been run and successfully removed the bridge from the ground model therefore allowing water to flow naturally in the ground model.

The main purpose of this thesis is to implement a bridge detection and classification algorithm able to detect bridges and to classify their points. A project is currently carried out by the Swedish mapping, cadastral and land registration authority (Lantmäteriet) to produce a new national elevation model. As a part of this project airborne laser scanning covering the full extent of Sweden is carried out. Data acquired during these scans will be used to evaluate the developed algorithm as the data covers a wide range of landscapes with known and homogeneous quality of the point clouds.

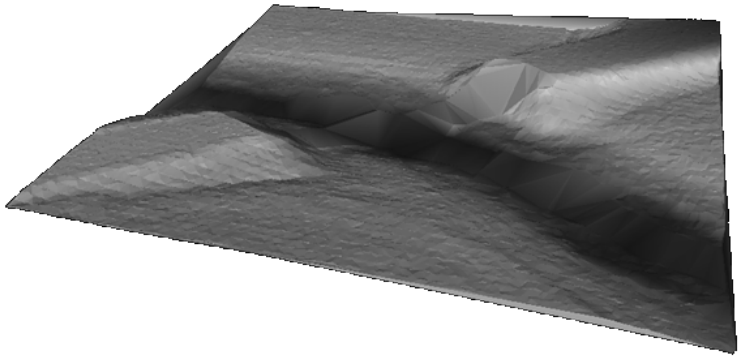
A method for detecting structures that are extensions to the bare earth, for instance bridges was proposed by Sithole et al. in [6]. The method has been further developed

by Sithole in [7] and by Sithole et al. in [8] and is the method chosen to be the basis of this project. The described method does not only include detection of structures as bridges but also include a method for filtering out vegetation and building points which will also be implemented as part of the project as this is a necessary preprocessing step of the bridge detection algorithm.

Data collected by airborne laser scanners typically contain large amounts of points, therefore algorithms performing complex computations on the data can suffer from poor time performance. To reduce the search space for bridge detection the possibilities of using existing geographical data to find coordinates where bridges potentially are located will be examined. The idea is to utilize the large amounts of existing geographical data where rivers, lakes, roads and railroads are described together with their locations. By searching for intersections between roads and water in this data one can acquire coordinates where bridges are likely to occur. By searching for bridges only at these coordinates the search space can be substantially reduced.



(a) Ground model containing bridge



(b) Ground model after bridge points have been removed

Figure 1.1: Illustration of bridges in ground models, the darker region in the figures is a river

1.4 Previous work

The literature on automatic bridge detection in general is sparse and the literature that exists is mostly aimed at bridge detection in satellite imagery and synthetic aperture radar imagery (SAR). SAR is a form of radar able to capture remote sensing information over long distances.

Trias-Sanz et al. [9] propose a method to find bridges in high resolution satellite images. The suggested method works by classifying input pixels as the terrain types water, vegetation, building, railroad, road, bridge, and roundabout. The classification is obtained by using a neural network and the radiometric, geometric and textural features of each pixel. Bridges are then detected in the input file by applying a set of detection rules for example by looking for positions where large regions of water are separated by narrow strips of other types of pixel types.

A technique for detecting bridges in SAR imagery is suggested by Wange et al. [10]. The algorithm uses the assumption that all bridges are part of roads and have sides with parallel straight fences. By using polarimetric and geometric features to identify the fences, roads and bridges can be detected.

When it comes to bridge detection in airborne laser scanner data one method is proposed by Carlson et al. in [11]. The method uses a supervised machine learning approach to identify bridges and other hydrological barriers in digital terrain models. Bridges are located based on several features deemed to be typical for a bridge for instance by looking for areas with a large gradient. When the algorithm has detected a bridge in the terrain model, the bridge is cut to correct hydrological modelling in the terrain model.

As explained in Section 1.3 the method developed by Sithole et al. was chosen to be the basis of this thesis. The algorithm uses only the spatial relationship between points and segments in the point cloud to find bridges. Bridges are found by first extracting all ground and bridge points, after that potential bridge segments are identified among the ground points and in the final step bridges are detected among the potential bridge segments.

The main reason behind this choice was that it is the only algorithm that is working directly on the point cloud, therefore being the only algorithm capable of accurately classifying individual points. This is in contrast to the algorithm described by Carlson et al. which does not explicitly classify bridge points but instead aims to cut through bridges in terrain models. Furthermore the algorithm described by Carlson et al. uses fill maps and hydrological conditioning to specifically target bridges that are problematic in flood simulations as opposed to the more general algorithm proposed by Sithole et al. which is not restricted by these conditions.

Chapter 2

Method

2.1 Airborne laser scanning

Data collection by airborne laser scanning mainly relies on two sensor systems mounted on an airplane or helicopter. The first of the two main sensor systems is a laser rangefinder. The rangefinder sends out laser pulses towards the ground, registers the reflected pulses and measures the time needed for each pulse to travel to the surface and back to the airplane.

By employing the well known relation between distance time and velocity the distance, d, to the point that the pulse was reflected on can be calculated.

$$d = c * \frac{t}{2}$$

Where c is the speed of light and t is the time for the pulse to travel to the reflected point on the ground and back to the detector.

The distance from the laser rangefinder to the point is not enough in order to map the recorded point to a specific 3D point in the landscape. To calculate the geographic 3D coordinates for the reflected point the location and orientation of the rangefinder at the time the pulse was sent also need to be known.

The second of the two main sensor systems is a collection of sensors designed to record the position, height, roll, pitch and yaw of the laser rangefinder when each pulse is sent out. Information about the position can be found using GPS-technology and an inertial navigation system (INS) containing gyroscopes and accelerometers records the roll, pitch and yaw of the rangefinder. By combining all the recorded information including the distance from the rangefinder to the ground point for each pulse and calibration data extracted from overlapping areas from different flights, it is possible to create a 3D point cloud with the coordinates of each point represented in an exterior coordinate system as described by Kilian et al. [12].

To be able to effectively scan larger areas in just one flight scanning is to some extent performed out from the plane's sides. The scanning can for example, as shown in Figure 2.1, be performed in a direction perpendicular to the flying direction. It should however be noted that Figure 2.1 only illustrates one possible scan pattern, other scan patterns include elliptical and z-shaped scanning. The angle between the rangefinder and a scanned point can range as much as between $-30\,^\circ$ and $30\,^\circ$. However as shown in Figure 2.2 a larger angle decrease the number of ground reflections in forested areas as the distance the pulse has to travel through the forest canopy increases therefore also the probability of the pulse being reflected by branches and leafs increases.

Besides collecting information about the coordinates of each point, airborne laser scanners typically have the ability to collect more information about the scanned points. The detector can for instance measure the amplitude of the returned pulse and from that determine an intensity value of each point. Since different surfaces do not reflect pulses equally well this information can be utilized during classification. Figure 2.4 shows an example point cloud where the points are coloured by intensity. In the figure a road, trees and several building roofs can be seen.

Many systems are also able to record several distinct reflections for each emitted pulse, this is made possible by registering several distinct amplitude peaks in the reflected pulse. Multiple returns occur when the laser pulse hits hard edges and part of the laser pulse is reflected while other parts continue toward other points of the landscape. Figure 2.3 illustrates two pulses that each give raise to two returns, the points labelled '1' and '3' represent the first returns of each pulse and the points labelled '2' and '4' illustrate the last returns. Multiple returns often occur at building edges and in areas with vegetation where pulses can be partially reflected by small branches and leafs before continuing toward the ground. Information about the return number of each point can be used for example during ground extraction since the first return of a multiple return pulse should typically not be part of the ground.

The scanner system might also be equipped with high resolution cameras able to capture RGB images of the scanned area at the time of scanning, the images can for instance be used to separate buildings and trees or to identify tree species.

For a more comprehensive explanation of theory behind airborne laser scanning and creation of point clouds see Wehr et al. [13] and Brenner [14].

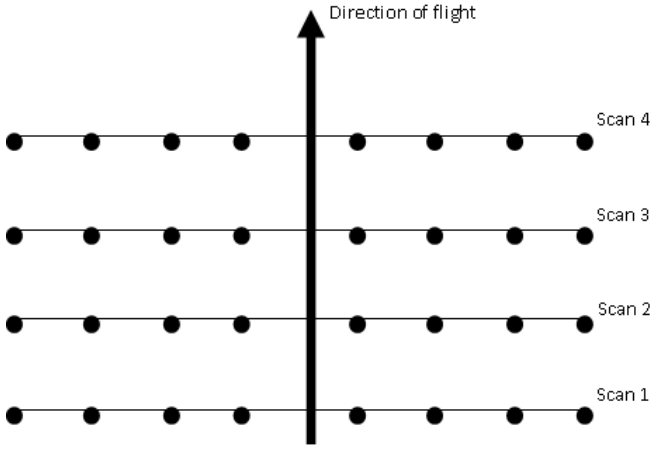
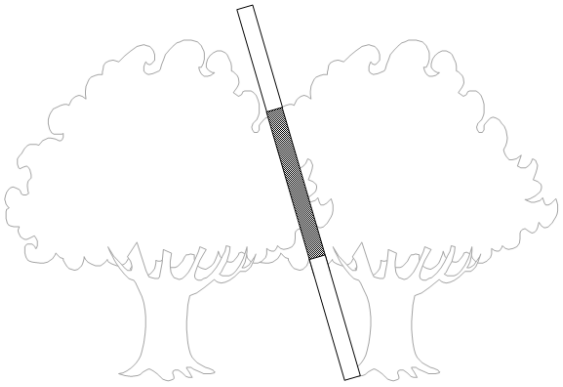
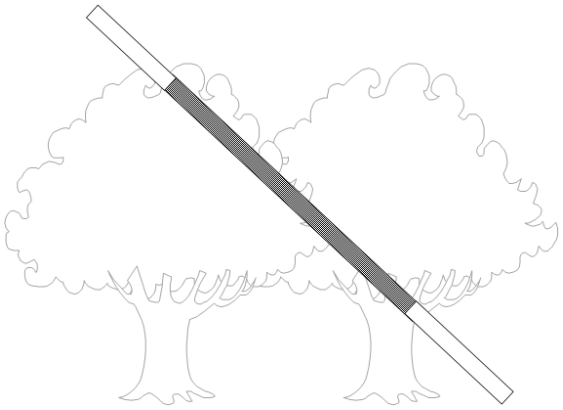


Figure 2.1: Scanning performed perpendicular to direction of flight



(a) Small scan angle



(b) Large scan angle

Figure 2.2: Impact of large scan angle in forested areas, grey area shows distance travelled through canopies

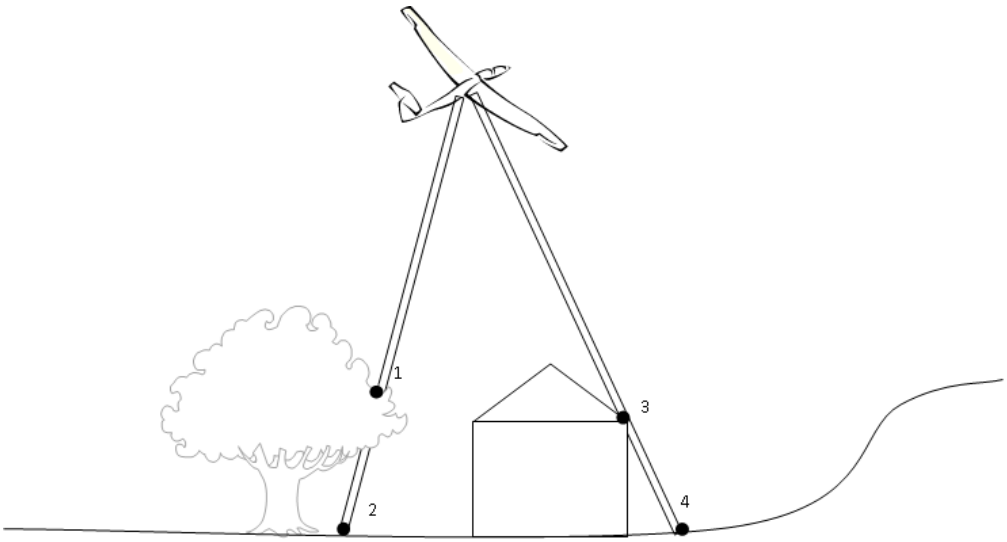


Figure 2.3: Multiple returns resulting from partial reflections of laser pulses

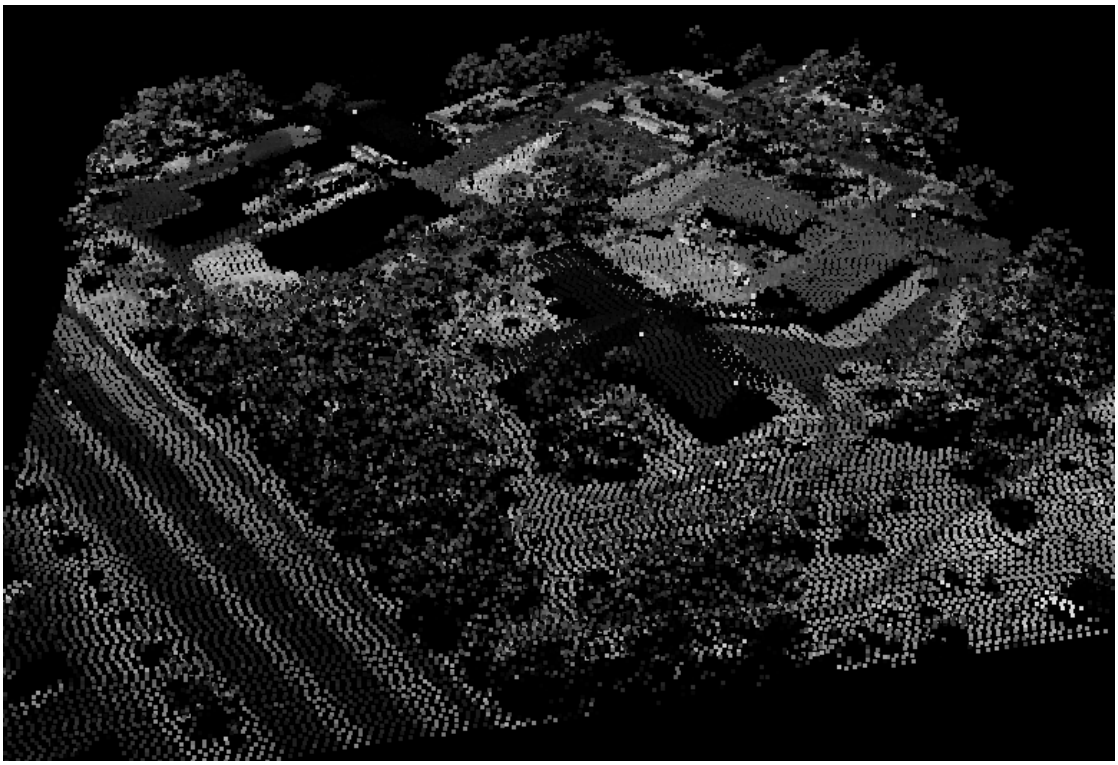


Figure 2.4: Examle point cloud, points coloured by intensity

2.2 Overview of the algorithms

The overall objective of the algorithm is to take a point cloud acquired by airborne laser scanning as input and classify all points belonging to bridges so that they can be ignored during generation of a ground model. To achieve this the algorithm is divided into four main parts, the first two parts are designed to reduce the input and identify potential bridges while the last two steps aim at detecting the bridges and collecting the bridge points.

The first step is designed to remove all points from the input point cloud that do not belong to the bare earth or bridges. As previously noted in Section 1.3, bridges exhibit properties typical for both ground and other objects. By exploiting that bridges extend smoothly from the bare earth, bridges and bare earth can be grouped together and kept while vegetation and building points are removed.

In the second step potential bridge points are identified among the ground and bridge points. In the first step the bare earth properties of a bridge were used, however to separate the bridges from the bare earth the fact that bridges typically are raised over their immediate neighbourhood is used. Most points belonging to the bare earth should not be raised above their neighbourhood, therefore potential bridge points are identified by searching for points that are raised above their immediate neighbourhood.

During the third step bridges are detected among the potential bridge points by creating potential bridge segments and comparing them to two assumptions assumed to be true only for bridges and in the last step the points of detected bridges are collected with the help of a bounding polygon of each bridge.

Section 2.3 to Section 2.6 will explain the necessary parts of object removal and bridge detection. Section 2.7 to Section 2.8 will describe object removal and bridge detection in more detail.

As previously mentioned the ideas behind automatic bridge detection using segmentation of profiles into line segments were first presented by Sithole et al. [6] and several suiting segmentation strategies were proposed by Sithole in [7]. The sections about profiling, segmentation, calculation of segment shapes, object removal and bridge detection presents the ideas introduced by Sithole et al. that are used in this project and explain them in more detail.

2.3 Gridding of data

Data collected by airborne laser scanning is by nature irregularly sampled within an area i.e. the distance and spatial relationship between two consecutive points can vary. The segmentation algorithms used in this project rely on the data being organised such that

the data can be accessed, extracted and analysed in a regular way.

To organise data acquired by airborne laser scanning in a regular fashion a regularly sampled grid is often used. The regularly sampled grid is obtained by overlaying the input data with a regularly sized grid, the value of each cell in the grid can be calculated in different ways according to the intended usage of the grid.

The value of each cell is typically a numerical value that represents one or several properties of the points that fall within the cell boundaries. Since several points can map to the same cell it can be necessary to interpolate the property values of points in order to obtain one value of each cell. Interpolation may also have to be performed to obtain values for cells without any points.

Examples of possible values of the cells and usages of the created grids include:

- Minimum z-coordinate among the points of the cell can be used for example when generating ground models (digital terrain models).
- Maximum z-coordinate among the points of the cell, used to model the surface of the landscape as seen from the sky (digital surface models).
- Number of multiple echoes in each cell. Since multiple echoes often occur in forested areas or at building edges this can be used in classification of such areas.
- Intensity data of points in each cell can be used for example to help identify roads based on the poor reflectivity of asphalt.

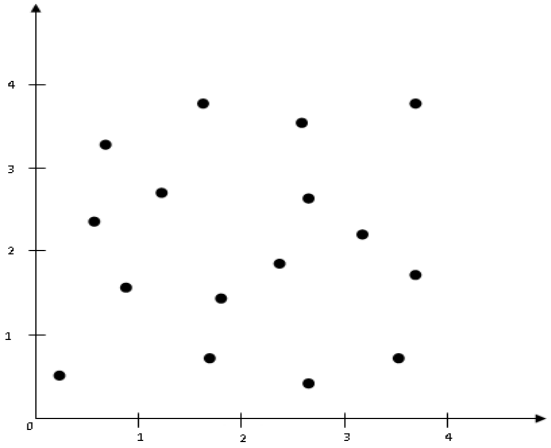
In this project however another approach was taken, the reason is that gridding with interpolation is meant to reduce the complexity of the data and make the data completely regular. The algorithms in this project work on a point basis and are to determine relationships between individual points while using the grid only as a way of accessing the data in a regular and effective way.

To create a grid that does not lose information by interpolation and that enables access to the input data in a regular way, instead of each cell being an interpolated scalar value based on the points in each cell; each cell is designed to contain information about all points that fall within its boundaries. By using this type of grid each individual point can be accessed by first locating the grid cell the point belongs to and then searching among the points in that cell, therefore allowing separate processing of each point.

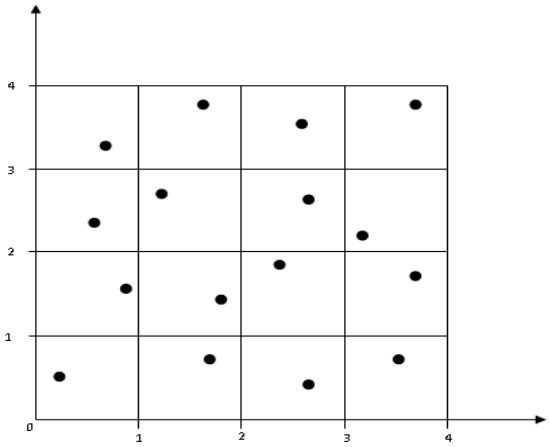
To get an abstraction of regularly sampled data each grid cell should ideally map to only one point. Controlling the number of points that map to each grid cell is accomplished by adjusting the grid resolution. The grid resolution determines how large each cell in the grid should be, larger cells cover a larger area and therefore the probability of points falling inside the cell boundary increases. A too small cell size can cause the

number of cells to become much larger than the number of points in the input data therefore producing a large number of empty cells. If the cells on the other hand are too large the separation of points becomes poor and the advantages of using the grid disappear.

Figure 2.5(a) shows 16 irregularly distributed points. If the task is to perform some computation on all points with coordinates $1 < x < 2$ and $2 < y < 3$, the x- and y-coordinates of all points have to be compared to find the desired points. In Figure 2.5(b) on the other hand the points have been gridded into a grid where each cell is 1*1 units large. By storing the grid efficiently in an array and with knowledge of the grid resolution the desired point can be directly accessed and processed.



(a) Irregularly distributed data



(b) Gridded irregularly distributed data

Figure 2.5: Gridding of irregularly distributed data

2.4 Profiling

The underlying concept of the segmentation strategies used in this thesis is based on approximation of surfaces by planar curves. By constructing planar curves that pass through points on a surface, the surface can be approximated by overlaying of these curves, henceforth these curves will be referred to as line segments. The concept is further discussed and motivated by Sithole in [7] pp72-74.

In a typical landscape these surfaces can be overlapping, for instance layered roofs and forest canopies often overlap each other or ground surfaces, therefore also the line segments that approximate the surfaces are allowed to be overlapping. The segmentation methods used to create the line segments in the input data all work by segmenting 2D curves into 2D line segments, where each line segment belongs to one distinct surface.

Since the shape and orientation of surfaces in the landscape are unknown the curves to be segmented will be extracted in several orientations to ensure that all parts of a surface can be captured. Extracting 2D curves from the irregular 3D point cloud is nontrivial as the points are irregularly distributed and not necessarily collinear in any 2 dimensions. Instead of true 2D curves being extracted the gridded data is used to extract sets of points that can be approximated by 2D curves.

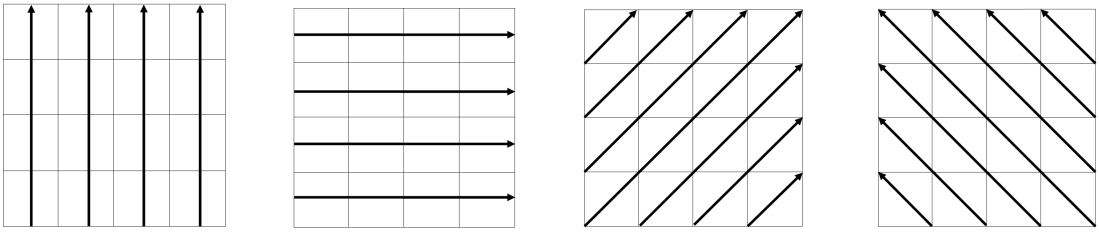
The grid that is created during the gridding process can be seen as a matrix where each cell contains a set of points. If the gridding was performed with a cell size such that the grid provides an abstraction of regularly spaced data each row, column and diagonal of the grid can be approximated by a 2D curve. To see why this is the case, consider one column of the grid created from a typical input file. The height (variation in y-coordinates) of the column will be approximately the size of the area covered by the point cloud, typically $> 500m$. The width of the column on the other hand will be equal to the cell size which should be proportional to the point spacing and will typically be in the order of $0.5m - 2m$. This means that a column can be approximated by a 2D curve by discarding the x-coordinate of each point, as the variation in x is many times smaller than the variation in y. By this reasoning a 2D curve is created from each column by using only the y- and z-coordinates the same argument can be applied to the rows and diagonals of the grid which are approximated by 2D curves in a similar fashion.

Each extracted 2D curve will henceforth be called a profile and consists of a set of points with only two coordinates called x and y, the points are ordered by the value of their x-coordinates.

Profiles can be extracted in a number of directions; more directions yield more accurate approximations of surfaces. In this project four main directions have been used to extract profiles. These directions are shown in Figure 2.6.

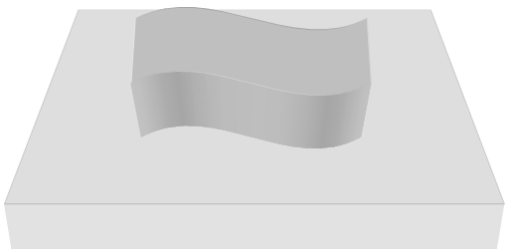
The extracted profiles cannot be used directly to approximate individual surfaces in the input data as they cover the full extent of the input and not only one surface. The profiles are segmented by the segmentation algorithms described in the next section to become a set of line segments. To transform the line segments into surfaces the line segments from different directions are overlaid to create a disconnected graph where each subgraph represents a surface segment in the input data.

An example of this is shown in Figure 2.7. Figure 2.7(a) and 2.7(b) show the area to be segmented into surface segments. Figure 2.7(c) and 2.7(d) show profiles from two directions that have been extracted and segmented into line segments based on the vertical distance of consecutive points in the profiles. In Figure 2.7(e) the overlaid and connected line segments are shown, it can be seen that by using two directions and overlaying the points of the complex shape and the ground is partitioned into two disjoint subgraphs representing the two surfaces.

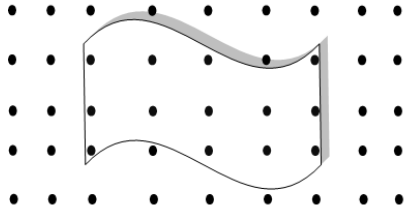


(a) First direction (b) Second direction (c) Third drection (d) Fourth direction

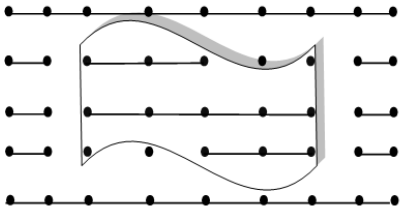
Figure 2.6: Directions used to extract profiles in this project



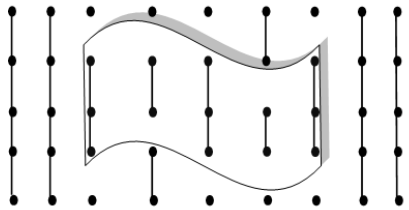
(a) Example area to segment



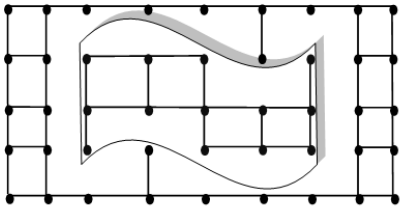
(b) Points registered in the area



(c) Horizontal line segments



(d) Vertical line segmenrts



(e) Overlaid and connected line segments yielding surface segments

Figure 2.7: Example segmentation and overlaying of line segments

2.5 Segmentation

Several of the algorithms used in this project depend on the ability to segment the input data into surface segments. Each surface segment ideally only contains points that are located on the same surface in the landscape and the segments are disjoint i.e. no point can belong to two segments.

Segmentation will in this project be used for three main applications: removal of objects like buildings and vegetation, identification of potential bridge points and grouping of potential bridge points into potential bridge segments. A segmentation algorithm was chosen and implemented for each of these applications according to the challenges of the application. All the segmentation algorithms operate on profiles and produce line segments, however the line segments are postprocessed and used in different ways for the three applications.

The segmentation strategies are named segmentation by proximity, segmentation by minimum spanning tree and segmentation by consecutive slope, based on the techniques they use to segment profiles into line segments. While segmentation by minimum spanning tree and segmentation by proximity uses overlaying to create surface segments, segmentation by consecutive slope does not.

It should also be mentioned that segmentation by consecutive slope creates line segments by successively connecting points of the profiles by edges, while the two other segmentation algorithms use Delaunay triangulation to create an initial proximity graph of the points in each profile. Edges are then removed according to different criteria to create the final line segments.

Delaunay triangulation is a triangulation method often used for height interpolation. In this project it is used to create an initial proximity graph as the triangulation T of a set of points P exhibits several desirable properties, for example the smallest angle of the triangles in T is maximised and the minimum spanning tree of P is a subgraph of the triangulation. A more comprehensive discussion of Delaunay triangulation and its properties is provided by de Berg et al. in [15].

2.5.1 Segmentation by proximity

Segmentation by proximity is the most straightforward method of the three segmentation methods used in this project. When a profile P has been extracted, segments are created based on the proximity between points in the profile. However this simplicity also implies that it is not as accurate as the more elaborate segmentation by minimum spanning tree as will be explained later.

The steps in segmentation by proximity are:

1. Delaunay triangulate the profile P, and create a graph with the points of the profile as nodes and the triangle edges as edges.

2. For each of the triangle edges (p_1,p_2) in the resulting triangulation, calculate an edge weight,

$$weight(p_1,p_2) = \big(x(p_1) - x(p_2)\big)^2 + k * \big(y(p_1) - y(p_2)\big)^2$$

   where $x(p)$ and $y(p)$ is assumed to return the x- and y-coordinates of p. In other words the weight is calculated as the squared Euclidian distance with an extra weight k on the vertical distance.

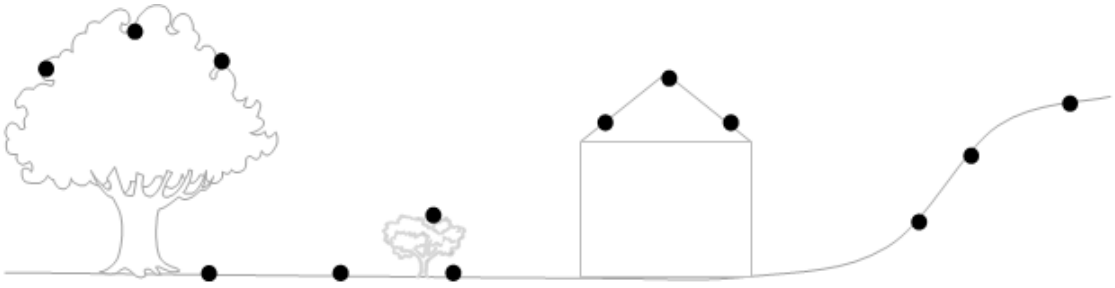
3. Remove all edges in the graph where the edge weight exceeds a threshold α.

4. When all profiles have been segmented into line segments the line segments are overlaid as previously shown in Figure 2.7 and the surface segments can be obtained by extracting the connected components of the created graph.

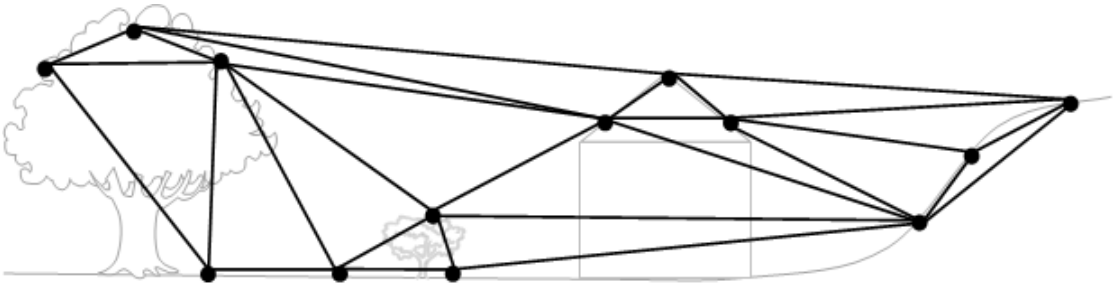
Recall that each profile is transformed to a 2D frame and y will in this frame be the height of a point while x will differ depending on the direction of the profile to be segmented. The variable k in the weight calculation is chosen to be > 1 to prefer connecting points that are close in the horizontal direction over points that are close in the vertical direction. In practice the scaling of the y-axis means that around a point p there exists an elliptic neighbourhood where the weight between p and other points can be smaller than α. By varying α and k and it is possible to control the semi-major and semi-minor axis of the ellipse. α and k should be chosen according to the vertical and horizontal separation between segments that are expected in the input data.

Figure 2.8 shows the steps of segmentation by proximity. In Figure 2.8(a) the profile to be segmented is shown and in Figure 2.8(b) the constructed triangulation of the points is shown. Figure 2.8(c) shows the result of removing triangle edges whose weight is larger than the threshold α.

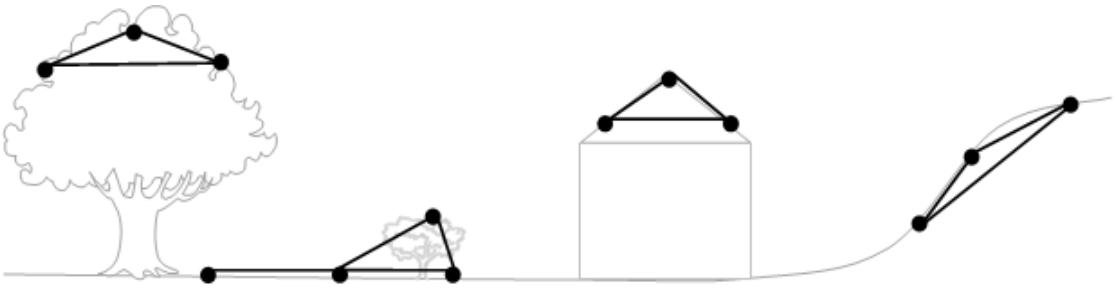
The main problem with segmentation by proximity can be seen in Figure 2.8(c) where a point registered in low vegetation has been incorrectly segmented together with the ground segment. One way of overcoming this problem would be to increase k and flatten the ellipse in where neighbouring points are allowed. However in doing so also the pitched roof, the steep slope in the terrain and the tree canopy could become disconnected. The next segmentation strategy is more elaborate and aims to overcome problems with low lying vegetation points.



(a) Points in profile



(b) Initial triangulation of profile



(c) Edges filtered based on their weight

Figure 2.8: Example of segmentation by proximity

2.5.2 Segmentation by minimum spanning tree

Segmentation by minimum spanning tree is similar to segmentation by proximity and uses Delaunay triangulation with the same weight calculation to construct an initial graph, however instead of just thresholding based on edge weights a more elaborate removal scheme of edges is employed.

The steps in segmentation by minimum spanning tree are:

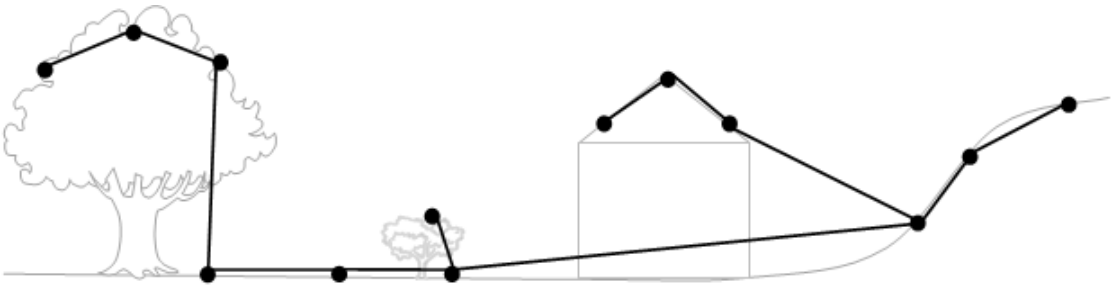
1. Delaunay triangulate the profile P, and create a graph with the points of the profile as nodes and the triangle edges as edges.
2. For each of the triangle edges (p_1,p_2) in the resulting triangulation, calculate an edge weight,
$$weight(p_1,p_2) = \big(x(p_1) - x(p_2)\big)^2 + k * \big(y(p_1) - y(p_2)\big)^2$$
3. Transform the graph into a minimum spanning tree based on the edge weights.
4. Remove all edges in the graph where the edge weight exceeds a threshold α.
5. Remove all dangling edges from the graph. A dangling edge is an edge that connects two points that are separated mainly along the vertical axis, more formally an edge is dangling if
$$\|x(p_1) - x(p_2)\| < \delta \text{ and } \|y(p_1) - y(p_2)\| > \epsilon$$
that is if the horizontal distance between the points is within a threshold δ and the vertical distance is greater than a threshold ϵ.
6. When all profiles have been segmented into line segments the line segments are overlaid as previously shown in Figure 2.7 and the surface segments can be obtained by extracting the connected components of the created graph.

The functions $x(p)$ and $y(p)$ and the variables α and k have the same function, and should be chosen in the same way, as in segmentation by proximity. The effect of removing dangling edges with the threshold values δ and ϵ is similar to the effect of increasing k. However by choosing the values carefully it is possible to only remove the dangling edge in the low vegetation without disconnecting the roof and steep slope segment, something that would have been impossible by only adjusting k.

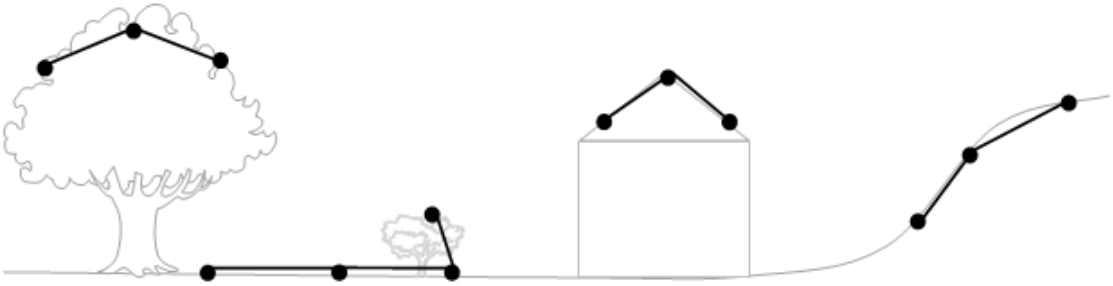
An example of segmentation by minimum spanning tree is shown in Figure 2.9. The initial profile and triangulation is the same as shown in Figure 2.8. In Figure 2.9(a) the minimum spanning tree of the initial triangulation is shown. Figure 2.9(b) shows the minimum spanning tree after edges have been thresholded against α. After removing dangling edges the resulting line segments are shown in Figure 2.9(c) where two distinct differences compared to the results of segmentation by proximity can be noted. Firstly

because of the removal of dangling edges, the point in the low vegetation was successfully disconnected from the ground segment. Secondly the created segments are not as tightly connected as the segments created by segmentation by proximity, this is because the minimum spanning tree does not contain any cycles.

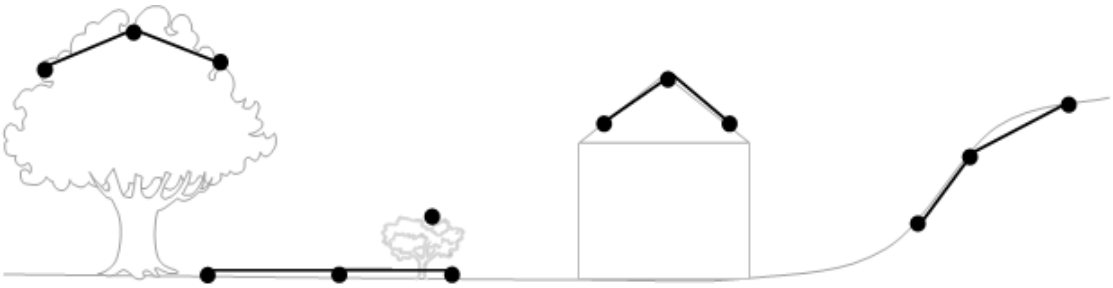
Adding the two steps, creation of a minimum spanning tree and removal of dangling edges, was motivated by the shortcomings of segmentation by proximity in separating low vegetation from the ground. The reason dangling edges cannot be removed directly after segmentation by proximity is that low lying vegetation points are typically connected by cycles to the ground segment. If cycles are present a dangling edge cannot be easily identified and removed, therefore a minimum spanning tree has to be determined first to remove such cycles and allow for simpler detection of dangling edges.



(a) Minimum spanning tree of triangulation graph



(b) Thresholded minimum spanning tree



(c) Dangling edges removed

Figure 2.9: Example of segmentation by minimum spanning tree

2.5.3 Segmentation by consecutive slope

Segmentation by consecutive slope differs from the two other segmentation methods in two aspects. Firstly its objective is not to create surface segments in the input data. Instead it aims to find individual points that are raised (potential bridge points). Secondly an initial connected graph is not created by triangulation; instead of removing edges from a connected graph consecutive points are connected to form line segments.

The reason for this is the intended input data of segmentation by consecutive slope, which is supposed to be only bridge and ground points. Overlaying of line segments would likely result in the merging of all line segments into only one segment covering the full input area since bridges are connected to the ground.

The steps in segmentation by consecutive slope are:

1. Find the first point $v_i \in P$, where P is the profile to be segmented, that has not yet been assigned to a line segment and assign it to a new line segment. If no such point can be found the segmentation is complete and all points belong to a line segment.

2. Find v_j the next point after v_i that does not belong to a line segment and where the horizontal difference is within a closeness threshold α. If no such point can be found then the line segment containing v_i is complete and segmentation proceeds at step (1)

3. Test if
$$\angle_{v_i,v_j} < \angle_{threshold}$$
i.e. if the angle between the points is smaller than an angle threshold.
   - If the condition holds then assign v_j to the segment that v_i belongs to, set $v_i := v_j$ and continue the segmentation at step (2).
   - If the condition does not hold then continue to search for another point belonging to the line segment at step (2)

Where $\angle_{v_i,v_j}$ represents the angle between the two points v_i and v_j.

Even though the segmentation by consecutive slope is performed only on the points belonging to ground and bridges, the steps of the segmentation will be illustrated using the same example profile as previously used for the other segmentation methods. The profile is shown in Figure 2.10(a) where the points have been labelled according to their position in the profile.

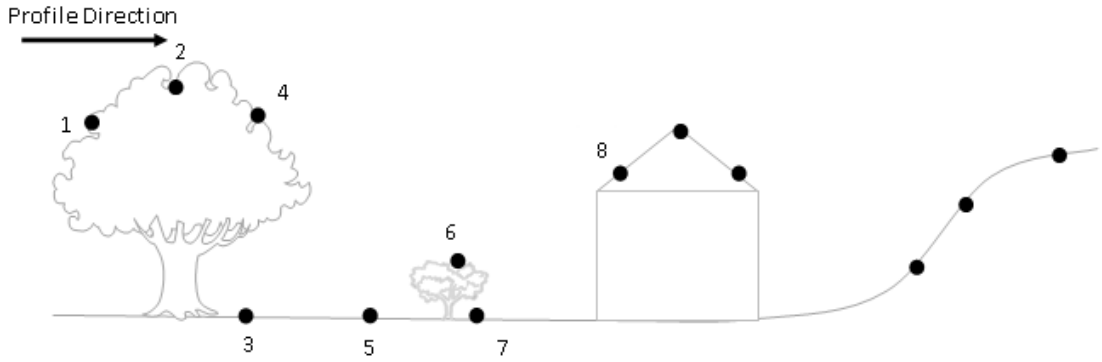
Figure 2.10(b) Initially v_i and v_j will be mapped to the points labelled '1' and '2' respectively. As the angle between them is smaller than the threshold they will be

put in the same line segment.

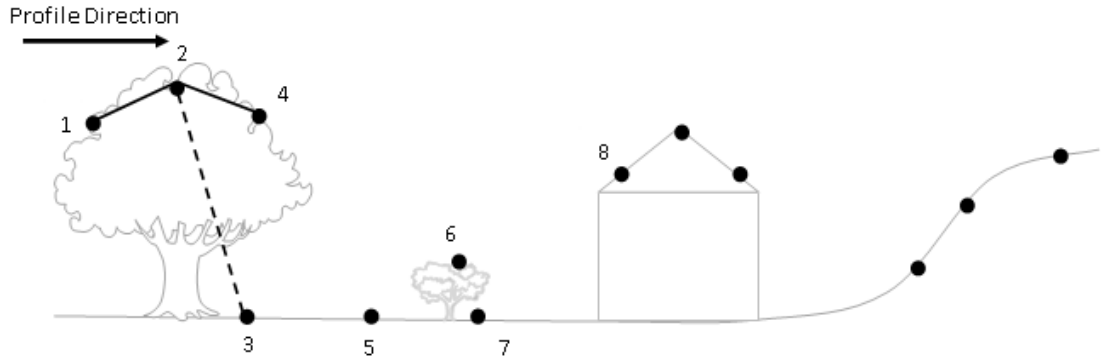
When v_i and v_j is mapped to the points labelled '2' and '3' the angle will be larger than the threshold and segmentation will proceed with v_i mapped to '2' and v_j mapped to '4' without mapping '3' to a line segment. When the point labelled '4' has been added to the current segment no more points that can belong to the segment can be found and a new line segment is started by setting v_i to the first unsegmented point, '3'.

Figure 2.10(c) Continuing from Figure 2.10(b) the first line segment is completed and contains the points labelled '1', '2' and '4'. Starting with v_i as '3' both the points '5' and '6' are added to the line segment according to the steps of the algorithm. With v_i mapping to '6', point '7' is not included in the line segment because the angle between the points is too large and the point labelled '8' is not tested because the horizontal distance from point '6' is too large. Therefore no more points can belong to the line segment and a new line segment is started with v_i mapping to the point labelled '7' as it is the leftmost unsegmented point.

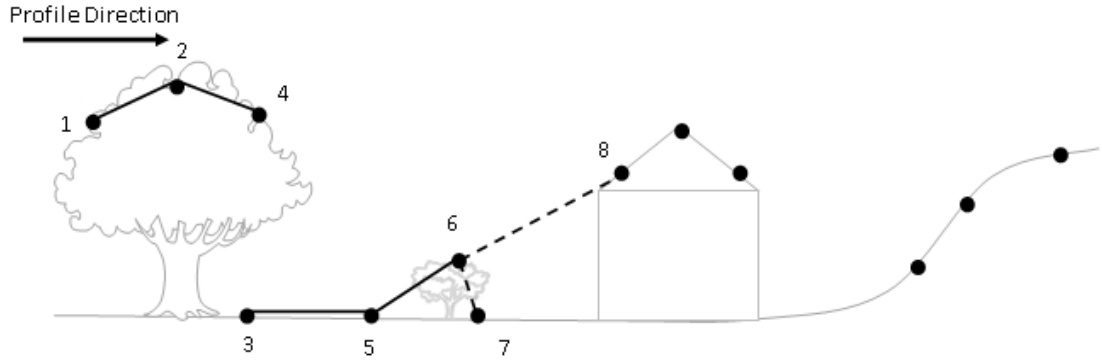
As previously mentioned the consecutive slope algorithm is used to find raised line segments in the ground model. The ground model is assumed to contain only two main segments: bare earth and bridges. The angle threshold $\angle_{threshold}$ should be chosen in such a way that it does not connect the bridge and the points below the bridge. If the bridge and ground segments are separated like this it is possible to find the bridge points by identifying raised line segments. The threshold should also be chosen in such a way that it does not oversegment steep slopes in the ground, therefore potentially creating artificial bridges.



(a) Initial profile with points labelled by their order in the profile



(b) First line segment found, dashed line indicates an edge that was not added due to its large angle



(c) Second line segment found, dashed lines indicate edges to points that were not added based on their angle or horizontal distance to the line segment

Figure 2.10: Example of segmentation by consecutive slope

2.5.4 Calculation of segment and line segment shapes

In some of the uses of segmentation in this project, knowing which points belong to a particular segment or line segment is not enough. To draw conclusions about the role a segment or line segment plays in the input data it is necessary to find its shape. Since a surface segment is made up from line segments naturally the shapes of the line segments will determine the shape the surface segment. Shape in this context is not related to the shape of the segment itself, e.g. if the points of the segment form a parabolic surface. Instead the shape describes a segment's relationship to surrounding segments.

The shape of a line segment l will solely be determined by the line segments immediately before and after l. By this limitation a line segment l can have six different shapes based on the spatial relationships to two neighbouring line segments.

Raised l is raised above both the neighbouring line segments, Figure 2.11(a)

Lowered l is lowered below its neighbouring line segments, Figure 2.11(b)

Terraced l is neighboured by one line segment lower than itself and one line segment higher than itself, Figure 2.11(c)

High l is neighboured in only one direction and by a line segment lower than itself, Figure 2.11(d)

Low l is neighboured in only one direction and by a line segment higher than itself, Figure 2.11(e)

No shape No line segments are in the vicinity of l, Figure 2.11(f)

As previously mentioned a surface segment consists of a collection of line segments that determine the shape of the segment. This imposes a problem in that surface segments can cover large areas and the shapes of the line segments may vary over the surface. Therefore the shape of a segment has to be determined from a combination of line segment shapes.

The most straightforward way to determine the shape from this combination is to choose the shape of the segment to be the most common shape of the line segments. The intuition behind this is that the most common shape among the line segments is the dominant shape of the surface segment. This approach was discarded because of the fact that large segments may consist of a large number of line segments, therefore when using this approach much information about the shape of a segment can be lost, especially in segments where the distribution of line segment shapes is fairly uniform.

To prevent shape information from being lost, the shape of a segment was chosen to be six values, representing shape grades for the six different shapes, raised lowered, terraced, high, low and no shape. The shape grade $g_{\mu,s}$ of a given shape μ and segment s

is a value that should reflect how much the segment resembles this shape. For example the raised shape grade, $g_{raised,s}$, of a segment s should be proportional to the number of raised line segments and the total number of line segments in s. One possible way to calculate the six shape grade values of a segment is by calculating the quota of each shape type. Using this approach the following equation can be used for calculating a segment's shape grade.

$$g_{\mu,s} = \frac{\|M_{\mu,s}\|}{\|M,s\|}$$

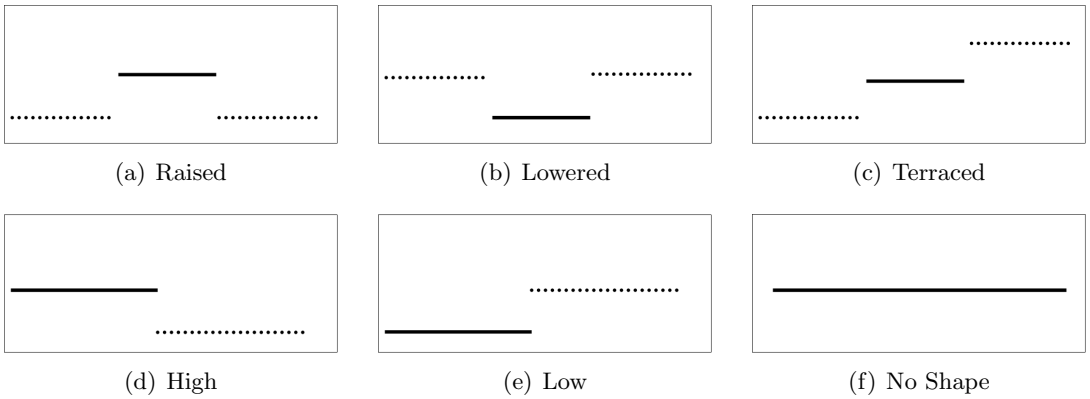
where $\|M_{\mu,s}\|$ is the number of line segments of shape μ in s and $\|M,s\|$ is the total number of line segments in s of any shape.

In [7] pp 100-101, Sithole argues that this approach produces biased shape grades because of the assumption that all line segments were extracted in the same direction. He further proposes a calculation that utilises the directions of the line segments to correct for this bias.

$$g_{\mu,s} = \sum_{i=1}^{D} \frac{\|M_{\mu,s,i}\|}{\|M_{s,i}\| * \|D\|}$$

Where D is the total number of directions that profiles were extracted in, $\|M_{s,i}\|$ is the number of line segments in direction i of any shape in the segment s. $\|M_{\mu,s,i}\|$ is the number of line segments of shape μ in direction i in the segment s.

This is the calculation that was used to calculate shapes of surface segments in the object removal since it more adequately handles segments whose shape varies in different parts of the segment and in different directions.



(a) Raised (b) Lowered (c) Terraced

(d) High (e) Low (f) No Shape

Figure 2.11: Possible line segment shapes(shape of filled line in relation to dotted lines)

2.6 Parameter estimation

The performance of the bridge classification depends on a large number of input parameters. Many of the parameters are independent of properties of the input data, for example the minimum width of a bridge should not vary between input files. However some parameters, for example the grid cell size, should be chosen based on the point spacing for the grid to provide an abstraction of regularly spaced data.

Another example of parameters that should be chosen depending on the point spacing are the parameters deciding the allowed horizontal distance between points during segmentation. To see why the allowed distance between two points should differ between two input files, assume that two different input files are to be segmented. The average point spacing varies greatly between the two files. Let the two files be denoted F_{dense} and F_{sparse} where

$$PointSpacing(F_{sparse}) \gg PointSpacing(F_{dense})$$

If the allowed distance δ during segmentation is chosen such that segmenting F_{dense} result in perfect separation of points belonging to the ground and buildings, then using the same δ while segmenting F_{sparse} where the points are further separated would likely cause over segmentation since δ is not large enough to connect neighbouring points in the ground segment. Therefore in input with relatively large point spacing δ should be chosen such that it is large enough to accommodate for the larger distance between the points.

On the other hand if δ is chosen to yield perfect separation between objects in F_{sparse}, using the same δ on F_{dense} would connect points that are separated by several other points and might cause the ground segment to climb up building walls and cause under segmentation. This shows that δ needs to be chosen for each input file and that δ depends on the point spacing in the input file.

Based on these observations it is clear that parameters dependant on the input data should automatically be determined based on the input data. The dependant parameters rely on estimation of the average point spacing in the input which can be estimated in several ways. Perhaps the most intuitive way is by calculating the related value, the point density of the input. If the number of points n and the area a covered by the points are known, calculation of the point density ρ is simply

$$\rho = \frac{n}{a}$$

The area covered by the input points can be estimated by a bounding box that encapsulates all points in the input file. Since all of the x- and y-coordinates of the input points are known we can define the bounding box to be the rectangle with its lower left corner at $\{min(x),min(y)\}$ and its upper right corner at $\{max(x),max(y)\}$, the area of this

rectangle is an approximation of the area covered by the points.

However the point density is not an accurate enough estimation of the point spacing. There are two reasons for this, firstly the point density of a file might be skewed in files containing areas with very high or very low point density. High point density usually occurs in the overlaps between different scans and low point density is found in areas containing deep water bodies. Since deep water generally generates few laser returns and can cover large areas the point density tends to underestimate the point spacing in these areas.

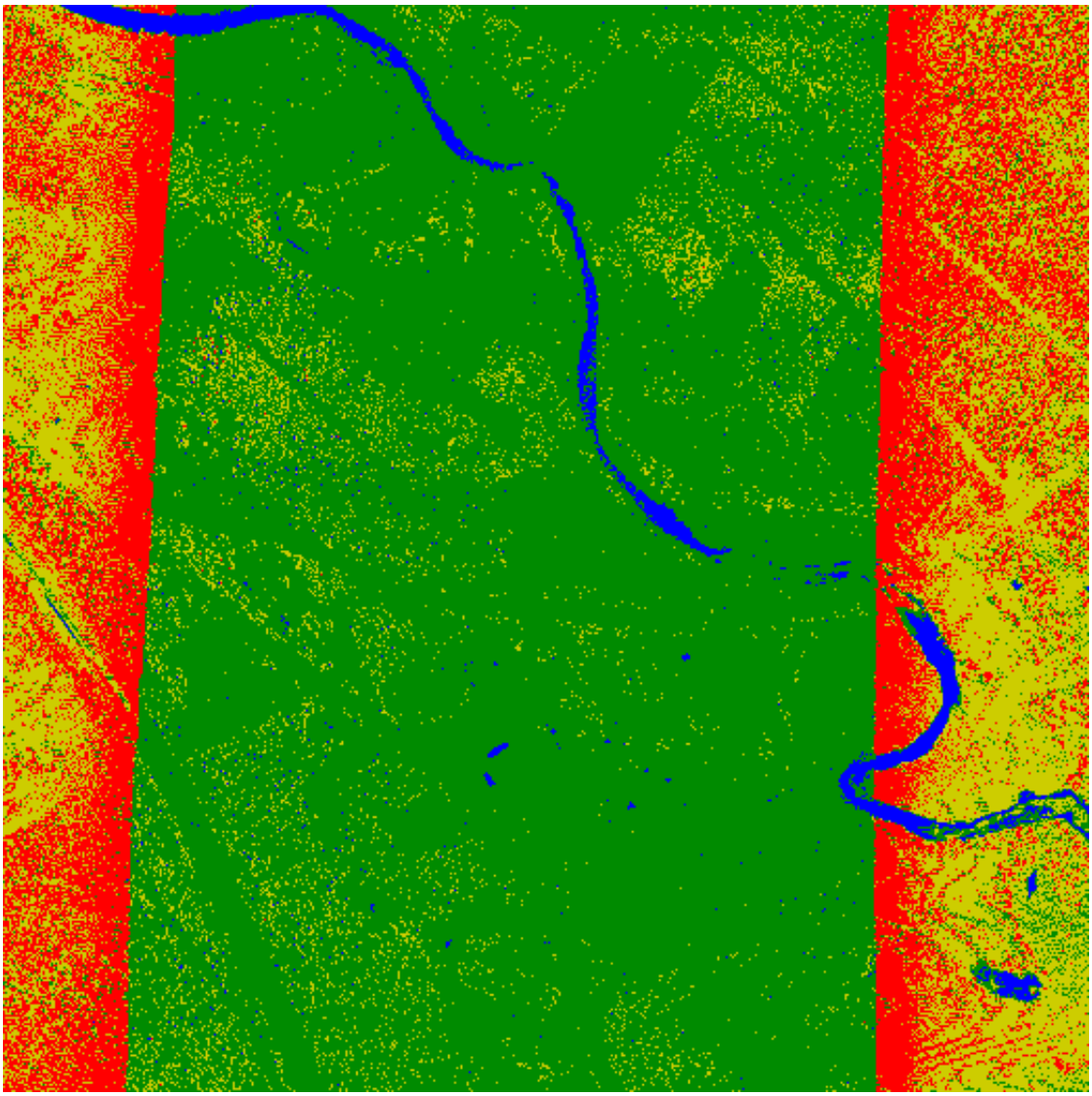
Secondly the point density can systematically vary in different directions within an input file as a result of scanning perpendicular to the direction of flight as shown in Figure 2.1. Since the segmentation operates on profiles extracted in different directions it is desired that the point spacing is known in at least two directions.

Figure 2.12 shows how the point density varies within an input area covering $1km \times 1km$. The blue areas where no points were registered are concentrated around a river. Merging of point clouds acquired during different flights produce the red regions with high point density. The yellow areas typically occur where trees cover the ground causing multiple returns of each pulse while the green region covers areas that are covered with sparse or no vegetation.

To estimate the point spacing of an input file the gridded data is used. However, as explained earlier the grid size should be chosen according to the point spacing. To overcome this dependency problem an initial estimation of the point spacing is made by calculating the point density. To overcome the problem of water bodies skewing the calculation of point density, an estimation of the empty area is calculated by creating a temporary grid based on the point density. The empty area is estimated by calculating the area covered by empty grid cells.

By subtracting the empty area from the area covered by the input data a new value for the point density can be calculated and used to create a grid that will be used to obtain the final estimation of the point spacing in the input data. As explained in Section 2.4 the points of an extracted profile can be simplified by a 2D line. By extracting all profiles in two directions, and for each one calculating the average horizontal distance between two consecutive points, an estimated point spacing can be acquired in two directions. The distance between two consecutive points in a profile should only included in the calculations if they fall within a given range since they can be part of a data gap or be aligned vertically in the landscape and not a result of the natural spacing between points.

With the values of point spacing acquired from these calculations the allowed horizontal distances in segmentation and the size of grid cells can be chosen individually for each input file.



Blue	0 pts/m^2
Green	$>$ 0 pts/m^2
Yellow	$>$ 1 pts/m^2
Red	$>$ 2 pts/m^2

Figure 2.12: Varying point density within an input data file

2.7 Object removal

The algorithm for detecting bridges is designed to work under the assumption that all the input points that are not part of bridges or ground have been filtered out. Because of this assumption it is necessary to find and remove all points not part of bridges or ground prior to detecting bridges. Many algorithms that extract ground points exist, however, as noted in Section 1.3, most algorithms were not designed to treat bridges explicitly and as a result the classification results of bridge points is unreliable.

The laser data available in this project have already undergone such classification for other purposes, for example for generating digital terrain models. It would be possible to use these classification results to directly extract the ground points needed for bridge detection. However, as previously discussed, the algorithms used do not handle bridges explicitly and bridge points may have been removed from the ground model. Therefore only partial information about bridges, their shape and relationship to surrounding points is available.

To enable accurate evaluation of the bridge detection algorithm a method for filtering out unwanted points while keeping bridge points had to be implemented as part of this project. The implemented method uses segmentation by minimum spanning tree together with the previously described method for calculating segment shapes to detect objects that are not part of bridges or the bare earth. Segmentation by minimum spanning tree was chosen based on its ability to remove low vegetation points from the bare earth segments as described in Section 2.5.2. Because of the function of a bridge it necessarily has to connect to the ground, therefore it should be connected to the ground segment by the segmentation algorithm via a line segment stretched along the bridge.

The first step of the removal is to segment the input data using segmentation by minimum spanning tree. When all segments have been obtained their shapes can be calculated as explained in Section 2.5.4. The obtained shape grades will be used to classify segments as either ground or object segments.

As previously explained, the shape grades of a segment consist of six different shapes: raised, lowered high, low, terraced and no shape. Based on the six shape grades an object grade, β_s, will be calculated for each segment s. The object grade can be said to represent how likely it is that a segment is part of an object e.g. a building or a tree. Calculation of the object grade is based on the observation that some shapes are more commonly found in buildings, trees and other objects than in ground segments. Raised and high line segments typically only occur in objects as building roofs and forest canopies are discontinuous to the bare earth. Terraced line segments on the other hand may occur both in the bare earth and in objects, they can for example appear in both sloped ground and in low vegetation.

Based on these observations the object grade of a segment s was chosen to be

$$\beta_s = g_{raised,s} + g_{high,s} + 0.5 * g_{terraced,s}$$

Classification is then performed by thresholding against a predefined object threshold β_{object}. All points of a segment s will be considered object points if

$$\beta_s > \beta_{object}$$

β_{object} should be chosen to be ~ 0.5 so that a segment is considered as an object segment if it is mainly raised, high or terraced.

Parts of a bridge will give raise to line segments that are raised and high however as the bridge will be part of the ground segment the frequency of such line segments will be relatively low compared to the frequency of other line segments.

When all segments have been classified as either ground or object and all object segments have been removed from the input, the object removal is iteratively applied to the remaining input. The reasoning behind iterating the object removal is that objects in many cases can be layered, the layering can for example be a result of high vegetation with underlying low vegetation, buildings with layered roofs and points registered in both the upper and lower parts of the forest canopy. During the first iteration underlying object layers can be considered as mostly lowered or terraced because of the top layer. During each iteration at least one layer is likely to be removed therefore exposing underlying object layers.

2.8 Bridge detection and classification

The detection and classification of bridges will be accomplished by three steps. The first two steps detect bridges while the aim of the third step is to gather all the points belonging to the detected bridges.

The steps of bridge detection and classification are:

1. Identification of potential bridges.
2. Filtering of false positives, i.e. detecting bridges among potential bridges.
3. Collection of points belonging to detected bridges.

2.8.1 Identifying potential bridges

After the filtering of objects has taken place, the input points should only include points belonging to bridges and the bare earth. Instead of having to separate bridges from all points the problem has now been reduced to separating bridges from the bare earth.

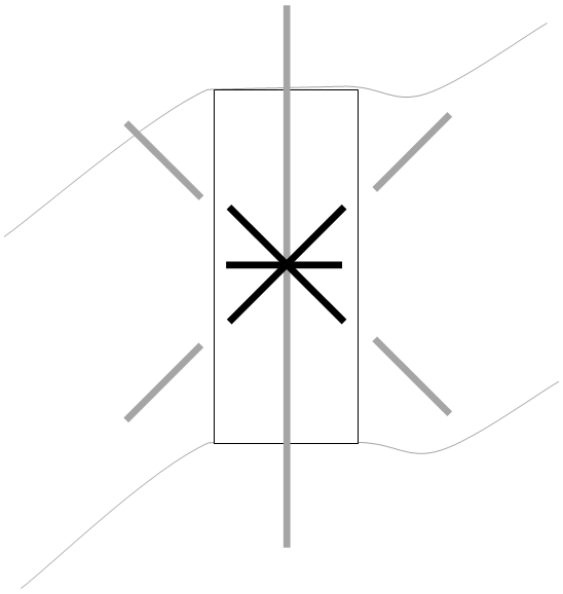
To detect the bridges, potential bridge points are identified based on properties that are expected to be different between points on bridges and points in the bare earth. The separating property used in this project is the height of a point relative to its neighbourhood. Finding raised points will be accomplished by finding raised line segments in the input. The reasoning behind choosing this approach is that, by definition, a bridge is something that is connecting two points in the bare earth by spanning above the underlying landscape. Therefore the majority of the line segments passing through a bridge are likely to be raised since the height difference between points at the edge of the bridge and points in the underlying landscape is typically large compared to other discontinuities in the ground.

This assumption is only true as long as a bridge is longer than it is wide. If a bridge is wider than it is long, many points on the bridge surface cannot be identified as being raised. Figure 2.13(a) shows a bridge that is longer than it is wide, together with line segments in four different directions: black lines represent raised line segments. It can be seen that a point in the centre of the bridge is considered as raised in three directions and as not raised in one direction and can therefore be identified as a potential bridge point.

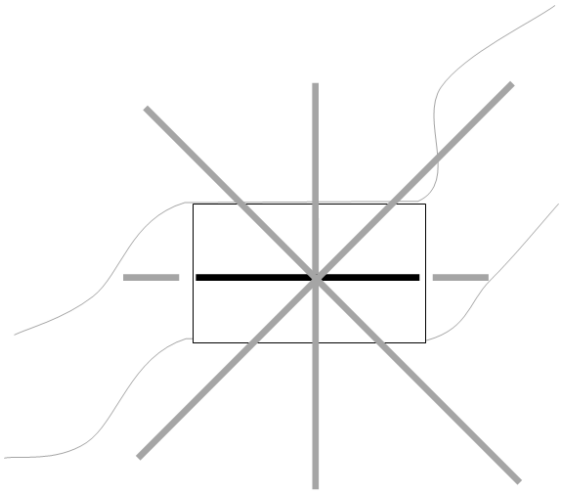
In Figure 2.13(b) a bridge that is wider than it is long is shown. A result of the increased width and decreased length is that the diagonal line segments no longer experience a drastic change in height therefore the majority of the line segments cannot be considered as raised.

As mentioned in Section 2.5.3 segmentation by consecutive slope is used to identify raised points in the input. The identification is performed by extracting all profiles in the different directions and applying segmentation by consecutive slope on the profiles to acquire line segments. The result is a set of line segments where each point is part of one line segment from each direction. By calculating the shape of each line segment the raised points are found by identifying the points whose majority of line segments are raised.

From the raised potential bridge points, potential bridge segments are created by grouping potential bridge points together. For this task, segmentation by proximity is used. Segmentation by proximity was chosen in favour of segmentation by minimum spanning tree because the input should not contain any low vegetation points or other problematic points that segmentation by minimum spanning tree was designed to handle. Therefore the more elaborate segmentation by minimum spanning tree would incur a larger time penalty without improving classification results significantly.



(a) Bridge that is longer than it is wide



(b) Bridge that is wider than it is long

Figure 2.13: Effects of bridges that are longer than they are wide, black lines represent raised line segments while grey lines represent other line segments

2.8.2 Detecting bridges

Assuming that potential bridge segments were found in the input data, the potential bridge segments need to be filtered as being raised is not a guarantee that a point belongs to a bridge. Raised line segments also occur for example at steep river banks and in sloped terrain with data gaps.

To filter out the true bridges from false positives two assumptions about bridges are employed:

Assumption 1 Bridges connect to the ground at a minimum of two locations.

Assumption 2 Points located at the bridge sides are raised above the surrounding neighbourhood.

To be able to compare potential bridge segments against these assumptions the assumptions have to be reformulated into:

Assumption 1 Each bridge contains at least 2 across edges.

Assumption 2 Each bridge contains at least 2 along edges.

The meaning of across and along edge will be explained towards the end of this section.

Each potential bridge segment is tested against these assumptions, if a segment satisfies both assumptions it is deemed to be part of a bridge. It can be observed that both of the original assumptions involve only the points at the bridge boundaries. Points in the interior of the bridge cannot be connected to the ground since they are neighboured only by other bridge points and they cannot be raised above their neighbourhood for the same reason. To test each potential bridge against the assumptions it is therefore necessary that the points at the bridge boundary are extracted.

A straightforward way of finding a boundary that fully encapsulates the potential bridge is by determining a convex hull of the set of potential bridge points. Bridges are however typically not convex by nature therefore a boundary determined by convex hull, in many cases, contains too few points for bridges to be properly tested against the assumptions. A typical bridge segment with its points is shown in Figure 2.14(a), notice how the bridge widens at the ramps where some points have been considered as raised and included in the potential bridge segment. In Figure 2.14(b) the convex hull of the bridge is shown. The convex hull of the potential bridge segment only consists of six points and no boundary points are collected from the sides of the bridge running along its length.

The approach taken in this project instead only include boundary edges if their length is smaller than a threshold, therefore by adjusting the maximum allowed distance between two consecutive points on the bridge boundary, the number of points included in the boundary can be controlled. To obtain such a boundary the points are Delaunay

triangulated, one of the properties of a Delaunay triangulation is that the convex hull of the points can be found by tracing along the outermost triangle edges. By removing triangle edges larger than a threshold all long edges of the convex hull can be removed.

When the long edges have been removed a boundary can be obtained by tracing along the outermost triangle edges of the modified triangulation. In Figure 2.14 the differences between the convex hull and a thresholded boundary are illustrated. It can be seen that the thresholded boundary contains points at the bridge edges which is necessary in order to detect that a bridge is raised above its surroundings.

To test the boundary of a potential bridge segment against the two assumptions two values need to be calculated for the boundary points, one for each assumption. The first value will be referred to as the smoothness value and is used to detect where a bridge makes a smooth transition into the ground. The second value will be referred to as the discontinuity value and is used to detect points located at the bridge sides, raised above surrounding points in the landscape.

The discontinuity value of a boundary point p is calculated as follows:

1. Determine the k nearest planimetric neighbours to p among the ground and bridge points.
2. Find the points with maximum and minimum height among the neighbours and call them p_{min} and p_{max}.
3. Calculate the height difference between p and the two points p_{min} and p_{max}.

$$d_{min} = z(p) - z(p_{min})$$

$$d_{max} = z(p) - z(p_{max})$$

4. If $\|d_{min}\| > \|d_{max}\|$ and $d_{min} > 0$ then set the discontinuity value to d_{min} otherwise set it to d_{max}.

Note that the discontinuity value can be negative, if that is the case then p cannot be located at the edge of a bridge since it means that the point in the neighbourhood of p with the largest height difference to p is located above p which should not occur at a bridge edge.

The smoothness value of a boundary point p is calculated as follows:

1. Determine the k nearest planimetric neighbours to p among the ground and bridge points.
2. Fit a plane to p and the neighbours, and calculate the standard deviation σ of the distances from the points to the fitted plane.

3. Threshold σ against a threshold value σ_{smooth} such that the smoothness value of the point p becomes a binary value, smooth or not smooth.

Fitting a plane to a number of points means to find the plane that minimises the standard deviation σ of the distance from the points to the plane. In [16] Shakarji shows that the problem of finding the plane that minimises σ can be expressed as an eigen-problem that can be solved by single value decomposition, which is the approach taken in this project.

The threshold value σ_{smooth} should be chosen such that points on sloped ramps leading up to bridges are considered smooth while points at the discontinuous edges of bridges are considered as non smooth.

The smoothness and the discontinuity values of the boundary points cannot directly be used to compare a potential bridge segment to the reformulated assumptions, it is also necessary to segment the boundary points into edges based on their smoothness value. The segmentation is accomplished by walking through the boundary and grouping two consecutive points to the same segment if they both are considered to be smooth or if they both are considered to be not smooth. Ideally this segmentation will result in four created edges for a bridge, one for each of the locations where the bridge connects to the ground and two edges that runs along its length.

The created edges can be organised into three different types:

Across edges Represents the locations where a bridge connects to the ground. Across edges are chosen to be the edges where the points have been labelled as smooth and where the total length of the edge is larger than a minimum bridge width.

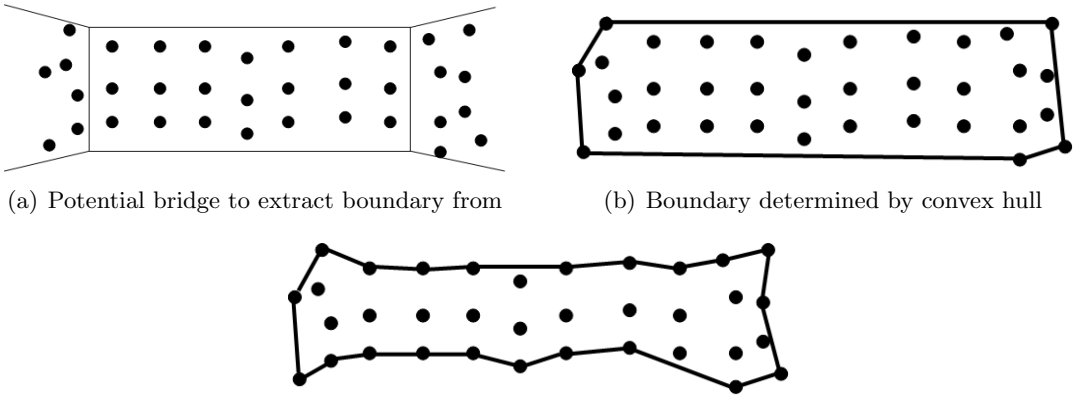
Along edges Represents the edges that run along a bridge side along its length. Along edges are chosen to be all edges whose points are considered as not smooth and where at least one point has a discontinuity value that is larger than a minimum bridge height

Other edges All created edges that do not qualify for the two other edge types

Figure 2.15 shows a simplified version of the bridge in Figure 2.14 where the created edges have been coloured according to their type, the thick black edges represent along edges and the grey edges represent across edges. It can be seen that the bridge consists of two along edges and two across edges. Therefore it fits the reformulated assumptions stated in the beginning of this section and can be detected as a bridge.

Filtering out true bridges from false positives is accomplished by extracting a boundary for each potential bridge segment. By calculating smoothness and discontinuity values of each boundary point the boundary can be segmented into edges that can be directly tested against the reformulated assumptions. If a potential bridge segment satisfies

the reformulated assumptions it is detected as a bridge, otherwise the potential bridge segment is discarded.



(a) Potential bridge to extract boundary from

(b) Boundary determined by convex hull

(c) Thresholded boundary traced from triangulation

Figure 2.14: Example of boundary determination of potential bridge segments

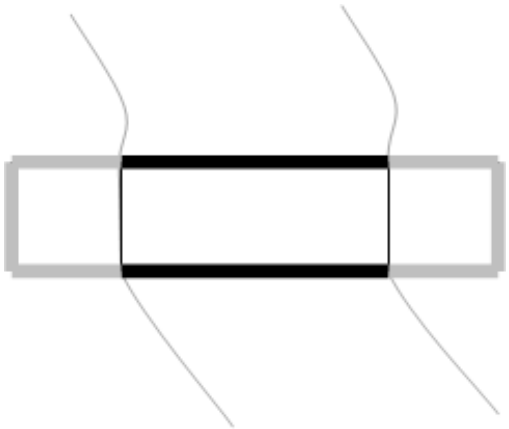


Figure 2.15: Bridge edges, coloured by type, the two grey edges represent across edges and the two black edges represent along edges.

2.8.3 Determining bridge points

As the bridges have been detected in the input the last step of the algorithm aims to find the points belonging to these bridges.

One possible solution would be to select only the points belonging to the bridge segment and classify those points as bridge points. This solution results in poor classification results because of two reasons. The first reason is related to the problem with bridges that are wider than they are long (Figure 2.13). This problem also arises to some extent in bridges where the difference between the bridge width and the bridge length is relatively small. In those cases points in the interior of the bridges will sometimes not be considered raised while points closer to the bridge boundary will. This is a result of line segments crossing through points in the centre of the bridge have a smaller probability to be raised for the same reasons as shown in Figure 2.13(b). The bridge can still be detected as a bridge but all its interior points will not be included in the potential bridge segment. The second reason can be seen in Figure 2.14(a) where it is shown that some points that are part of the upper part of the ramps tend to be included in the bridge segment even though they are not part of the bridge.

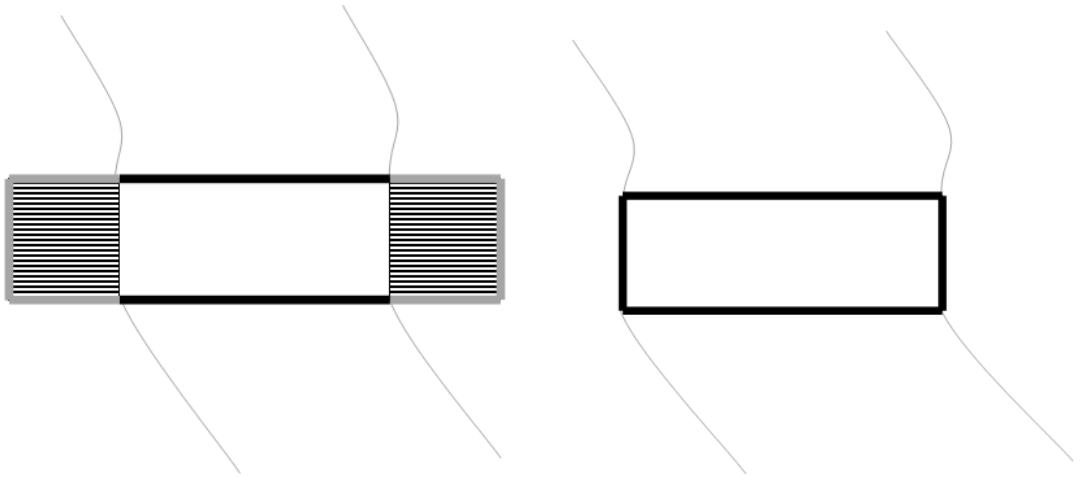
During detection of a bridge the boundary needs to be obtained. As explained in Section 2.8.2, this boundary can be used to collect the points of the bridge. One approach is to create a polygon from the along and across edges and collect all points within this polygon. This solution would include all the interior points of the bridge, however as shown in Figure 2.16(a) the bridge would still become slightly overestimated at the top of the ramps (striped areas).

To prevent points at the ramps from being misclassified as being part of bridges a polygon is instead created from only the along edges. Because the across edges cover the area where a bridge connects to the ground, at ramps, by removing the across edges the points on the ramps are excluded from the polygon. By collecting the bridge points as all points that fall inside the planimetric projection of this polygon points located at the bridge can be collected even if they are not part of the potential bridge segment.

The drawback of using this approach is that a new problem is introduced, namely that points registered directly below the bridge can fall within the polygon. To separate such points from points at the bridge surface, the vertical separation between the bridge surface and underlying points is exploited. The collected points are segmented by proximity, because of the vertical separation between the bridge points and underlying points they will be segmented into different segments. The segment containing the bridge surface can then be found by discarding all segments that do not include any point from the bridge boundary.

For each one of the bridges that were detected in Section 2.8.2 a polygon consisting of the along edges is created. By collecting the points inside each polygon, segmenting

and removing erroneous segments as described above, all bridge surface points in the input file can be collected and classified as bridge points.



(a) Overestimated bridge as a result of including across edges

(b) Polygon created from along edges

Figure 2.16: Creation of bridge polygons from across and along edges

2.9 Potential bridge coordinates

As stated in Section 1.3 detection and classification of bridges is a problem mainly related to flood simulations where bridges that cross water have been erroneously included in the digital terrain model. Input files can cover several square kilometres and contain millions of points, the algorithms described in this project will have to process all input points and extract all these points as part of profiles, thus consuming relatively much resources. Optimisation of source code and algorithms can to some extent reduce the consumption of resources but to further reduce the resource and time consumption of the algorithm a method for reducing the search space was implemented as part of the project.

In flood simulation bridges spanning water are mostly interesting, these bridges can be found by using the fact that there exists geographical data describing the road, rail and water networks in many areas today. The geographical software system ArcGis was employed to automatically overlay and find intersections between the road and water network and between the rail and water network. Coordinates of these intersections could then be extracted and used as coordinates of potential bridges.

This method can include coordinates that do not describe bridges for example due to poor accuracy of the geographical data but the number of such locations should be relatively low therefore reducing the search space substantially. An obvious drawback is that locations of bridges that do not span water will not be included in the resulting coordinates.

2.10 Software and file formats

The algorithms for object removal, bridge detection and bridge classification described in this project were developed using the programming language IDL, a dynamically typed and interpreted language used for analysis and visualisation of scientific data. IDL is developed and maintained by ITT Visual Information Solutions.

To find potential bridge coordinates described in Section 2.9, ArcGis and Python were used. ArcGis is a software package for analysis of geographical data that automatically can carry out several advanced operations, for example finding the intersections between the road and water network. Python scripts were used to automate the procedure of finding intersections and extracting their coordinates.

The point clouds collected by airborne laser scanning used in this project were distributed in the LASer (LAS) file format. Point clouds include large amounts of data. As a result, storage in ASCII induces performance penalties and, because no standard for storing airborne laser scanned data in ASCII files exist, exchange of data becomes inconvenient. To overcome these problems the LAS file format was developed by the American Society for Photogrammetry & Remote Sensing (ASPRS) Standards Commit-

tee. Information and the specification of the format can be found at [1].

The LAS format is a binary format and IDL does not provide standard functions for importing point clouds. To extend IDL with this functionality BCAL LiDAR Tools[2] developed by Idaho State University, Boise Center Aerospace Laboratory was used. The tools are licensed under GNU GPL v3 and provide means to read both header and point data from the LAS files in which the input data was delivered.

[1] http://www.asprs.org/a/society/committees/standards/lidar_exchange_format.html
[2] http://code.google.com/p/bcal-lidar-tools/

Chapter 3

Results and discussion

3.1 Results

To evaluate the performance of the bridge detection and classification algorithm a test area covering $200km^2$ was chosen. $185km^2$ was chosen from a continuous area consisting of a mixed landscape containing both urban and forested areas. As bridges occur with a relatively low frequency in the average landscape an additional area of $15km^2$, containing landscapes known to have a high frequency of bridges was selected to further test the algorithm's ability to detect and classify bridges.

The input data were distributed in square blocks covering approximately $1km^2$ each. The number of points in each input file ranged from approximately 700 000 points in unvegetated areas partly covered by water to approximately 2 million points in densely forested areas. The bridges in the input have lengths ranging from $\sim 5m$ to $\sim 600m$.

The number of bridges in the test data does not include bridges that cannot be detected by the algorithm because they do not adhere to the assumptions described in Section 2.8.2. These bridges are mainly of two types. Firstly bridges that span several input files cannot be guaranteed to have two points that connect to the bare earth and were therefore not included in the results. Secondly bridges whose edges are not raised over the surrounding terrain cannot be detected by the algorithm and were therefore discarded. Bridges that are not raised over the terrain typically occurs when the bridge surface is located very close to the underlying terrain in which case the bridge cannot be separated from the bare earth by the algorithm.

The results were evaluated based on three criteria:

- Number of bridges detected
- Number of bridges correctly classified
- Number of false positives

The number of bridges detected will include any bridge that is detected to some extent e.g. bridges where only one third of the bridge points were classified as bridge points will be counted as detected.

For a bridge to be correctly classified the requirement is not only that the bridge is detected but also that the four corners of the bridge are correctly detected so that the points of the bridge surface are correctly classified.

Figure 3.1 shows the determined polygon of a detected bridge. The bridge will be registered as a detected bridge although as its four corners have not been correctly identified it will not be registered as a correctly classified bridge.

A false positive is a segment that has been classified as a bridge segment even though the points in the segment do not belong to an actual bridge.

The results of applying the algorithm on the test area are shown in Table 3.1. The results show that the algorithm is able to detect the majority of the bridges that it was designed to detect. Seven bridges remained undetected, six of them were narrow and low bridges. Low bridges are hard to identify as their points are not easily identified as raised therefore not considered as potential bridge points. Narrow bridges are hard to identify as they connect to the bare earth by only a few points. The remaining undetected bridge could not be fully identified as a potential bridge segment because it was shadowed on one side by another bridge and on the other side by dense vegetation, therefore only a few points indicating that the bridge was raised could be found.

The results also show that there is a difference between the number of detected bridges and the number of correctly classified bridges. As described in Section 2.8.3 bridge points are found and classified by creating a polygon from the along edges and collecting the points inside the polygon. Therefore incorrect determination of along edges affects classification results in a negative way. The most common reason of incorrect along edges was found to be lack of data below the bridges, especially in the areas around bridge corners. Recall that the along edges are created from points that are located high in a nonsmooth neighbourhood. If there is a lack of data in the area below the bridge, the neighbourhood of a point at the bridge edge will only include other points on the bridge surface, thus incorrectly determine the neighbourhood to be smooth. Lack of data in the area below a bridge often occurs if the bridge is spanning a deep water body or if the water is shadowed by vegetation where few ground points can be registered.

A quite large amount of false positives were registered in the area, the false positives typically occur at steep slopes and in man-made structures that were incorrectly not identified as object points and removed during the object removal. It should also be mentioned that 50% of the false positives occur within $4km^2$ which is 2% of the total area. This suggests that false positives are much more likely to occur in certain types of

input.

The number of detected and correctly classified bridges can be increased by relaxing the parameters of the algorithm, although there is a trade off as more relaxed parameters can result in more false positives. To illustrate this trade off the algorithm was evaluated with more relaxed parameters. More precisely the minimum allowed width and length of a bridge were decreased to improve the detection rate and the size of the neighbourhood used to find across and along edges were increased to improve the number of correct classifications. The results can be seen in Table 3.2. As can be seen both the number of detected bridges and correctly classified bridges increased but with a relatively small amount compared to the number of false positives that increased with more than 50%.

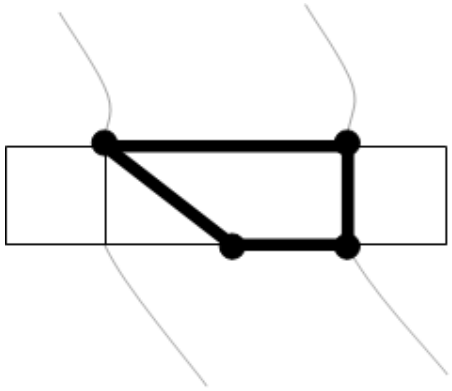


Figure 3.1: Example of detected but not correctly classified bridge

Area	185 km^2	15 km^2	Overall, 200 km^2
Bridges	46	33	79
Detected bridges	41	31	72
Detected bridges %	89.1%	93.9%	91.1%
Correctly classified bridges	35	30	65
Correctly classified bridges %	76.1%	90.9%	82.3%
False positives	22	4	26
False positives/km^2	0.12	0.26	0.13

Table 3.1: Results of bridge detection

Area	185 km^2	15 km^2	200 km^2
Bridges	46	33	79
Detected bridges	42	32	74
Detected bridges %	91.3%	97.0%	93.7%
Correctly classified bridges	38	32	70
Correctly classified bridges %	82.6%	97.0%	88.6%
False positives	33	8	41
False positives/km^2	0.18	0.53	0.21

Table 3.2: Results of bridge detection with less restrictive parameters

3.2 Discussion

The following sections will discuss potential problems with the implemented algorithms and problematic input types in a more general sense.

3.2.1 Object removal

Before detecting the bridges it was necessary that all points that might potentially disturb the detection were removed. During evaluation it was discovered that many of the errors in classification, especially the number of false positives were a result of poor object removal. The problems mainly occur in terrain with steep slopes, which can cause several false positives, and in areas covered by multi-layered vegetation. As explained in Section 2.7 the object removal iteratively removes object layers, in areas with many vegetation layers the lower layers can be included in the ground model.

Improvements in this part of the algorithm would not only decrease the number of false positives but would also allow the detection step to be run with less restrictive parameters which could result in more bridges being detected. In the paper where the object removal is described by Sithole [7] a method for removing micro objects is also proposed which potentially could be used to improve object removal results. However the proposed method cannot be run before bridges have been classified as it otherwise would identify parts of bridges as micro objects.

Extraction of ground points is perhaps the most well studied area within airborne laser scanning but no universal best method exists. The method used in this project was used because of its ability to retain bridge points in the ground model. The drawback on the other hand is that it proved to perform worse than expected in removing low vegetation points and in correctly classifying steep slopes, which gave rise to some of the false positives.

3.2.2 Bridge detection

Bridges are detected among the potential bridge segments by identifying potential bridge segments that have at least two across and two along edges. The most common reason for a bridge to remain undetected is that it is not identified as a potential bridge segment in the first place.

The algorithm for identifying potential bridges can fail for example in the following situations:

- Very thin bridges
- Lack of points below the bridge, due to deep water bodies or dense vegetation
- Bridges that are wider than long

- Data gaps in bridges for example due to railings, pylons or cables
- Bridges spanning several input files
- Two or more parallel bridges close together
- Poor removal of objects
- Small separation of bridge and ground or water below

Thin bridges can cause problems during object removal if the bridges only consist of one or two points along its width. For a bridge to not be removed during the object removal it needs to connect to the bare earth segment. The connection between the bridge and bare earth takes place along the width of the bridge. If the bridge is only one or two points wide the number of places where such a connection can take place is very limited and for example an erroneous chosen grid size can cause the bridge to be detached from the ground segment.

Thin bridges can also be incorrectly discarded even if they are identified as potential bridges, recall that a bridge needs to have at least two across and two along edges to be considered as a bridge during the last step of the bridge detection. If the bridge only has two points along its width then those points are likely to be considered as part of the along edges and since there are no boundary points located between those two points no across edges can be found.

For a bridge to be considered as a potential bridge the points of the bridge have to be raised. If the points are not found to be raised the bridge will be thought to be part of the bare earth and discarded. The specular reflection of laser pulses on water surfaces cause the number of points recorded at water surfaces around bridges to be very low in many cases. Dense vegetation around the bridge can cause few ground or water points to be registered around the bridge. If there is a lack of ground points in the neighbourhood of the bridge there are no references that indicate that the bridge is raised and therefore it can incorrectly be considered as part of the bare earth in the area.

As discussed in Section 2.8.1 and shown in Figure 2.13, bridges that have a large width compared to their length cannot be detected since potential bridges as many interior points cannot be identified as raised, therefore the algorithm is unable to detect such bridges. Bridges usually are longer than they are wide however some bridges that are wider than they are long were observed during evaluation of the algorithm.

Not all bridges have the surface of the bridge as its top layer. Some bridges, for example cable-stayed bridges, have a topmost layer of pylons and cables. During laser scanning the pylons and cables can shadow parts of the bridge causing it to be divided into separate regions. If the bridge is divided the different parts do not by themselves satisfy the assumption that a bridge connects to the bare earth at two points and therefore they cannot be detected as bridges.

Input necessarily have to be divided into separate files to handle the large amounts of data, however the division is not performed with respect to bridge locations. Occasionally this results in bridges that are divided into different input files. If a bridge is divided such that no one of the locations where it connects to the bare earth is part of the input, then the bridge is indistinguishable from for example a roof of a building and will be removed during object removal. If one of the locations where the bridge connects to the bare earth is included in the input the bridge will not be removed during object removal but as it will not satisfy the assumptions of a bridge it will not be classified as a bridge. This problem can of course be corrected by merging the input files that the bridge is part of or by more careful splitting of the data according to bridge locations. However this solution is unfeasible if no prior knowledge about bridge locations exists.

Sometimes two or more bridges are situated closely together, parallel to each other, they can for example be designed to handle different types of traffic or just different traffic directions. Such parallel bridges are harder to detect because they shadow each other's sides, therefore decreasing the possibilities of identifying along edges and making it harder to identify their points as raised.

As previously mentioned poor object removal can cause problems with thin bridges by discarding them as object points. It can also cause problems by not removing all object points. For instance if the object removal algorithms fails to remove objects points in the vicinity around a bridge the bridge risks to be considered as not raised because of surrounding elevated object points. This problem mainly arises in areas with dense vegetation.

In Section 3.1 it was stated that bridges that were not raised over the surrounding terrain were not included in the results. These bridges cannot be detected by the algorithm as the vertical separation between the bridge and ground or water below is very small. These bridges typically appear where the water surface is located just below the bridge surface, in which case the separation is not enough to identify the bridge points as raised potential bridge points.

3.2.3 Bridge classification

If a potential bridge segment contains at least two across and two along edges it will be detected as a bridge and its points should be classified as bridge points. The boundary of the bridge will be determined solely by a polygon created from the along edges. The along edges are created from points that are considered to be in a non-smooth neighbourhood and have a maximum discontinuity to the neighbourhood that is larger than a minimum bridge height. The most critical areas when establishing the bridge boundaries are at the corners of the bridge. Lack of data in the corner regions of a bridge may cause the along edges to be underestimated, thus causing the algorithm to fail in classifying all points on the bridge surface as bridge points.

Lack of data around the edges of a bridge can occur for a number of reasons. The ground or water surface below the bridge can be shadowed by the bridge itself, especially at the end of flight lines where the scan angle is large. It is also possible that no ground or water points are registered in these areas because of problems with the laser pulse's specular reflections in water, or because of shadowing dense vegetation. Shadowing vegetation is especially problematic when classifying bridges spanning narrow rivers, as vegetation at the river shore can shadow the full length of the bridge.

One classification problem that occasionally arose during evaluation was overestimation of the bridge. The problem can arise if the road leading up to a bridge continues to run on raised ramps after the bridge has terminated. If the ramps have steep side slopes or if the side slopes are shadowed by dense vegetation the ramp can incorrectly be classed as an extension of the bridge. A similar problem can also cause two consecutive bridges to be misclassified as one bridge if the road between them is raised and the road points are identified as potential bridge points.

3.2.4 False positives

As previously shown in Table 3.1 the algorithm performs well when it comes to detecting bridges in the input files. However accurate results also rely on that classified bridges actually are bridges and not points erroneously detected as bridges.

During evaluation some types of landscape and objects in the landscape proved to be more commonly misclassified as bridges than others:

- Steep slopes in the terrain
- Roofs and buildings that make a smooth connection to the ground
- Dense multi-layered vegetation

The problem with steep slopes arises during object removal and is partly caused by the fact that steep slopes in the terrain resemble building walls. Due to the steep slope many points that are part of the sloped area may be removed. The top of the slope often connects to the rest of the bare earth through some other less sloped area. This means that the top of the slope will at one side be discontinuous to the points below since there exists few points in the steep slope between the top and the ground below. At some other side the slope will connect smoothly to the ground. This in itself is not a problem since the top of the slope still only has what can be considered as one across and one along edge. However as a result of data gaps or some remaining points in the steep slope the edges can be divided. This can cause parts of the top of the slope to become incorrectly classified as a bridge.

The second type of objects that often appear as false positives are roofs and other

man-made structures that make a smooth transition to the ground. These structures will be connected to the ground segment during object removal and therefore included among the ground and bridge points. Roofs should typically be removed from those points however some types of buildings make a transition to the ground points that is very similar to the transition that bridges make. Some of the examples noted during evaluation were parking decks that are raised and have an outside ramp, and barns with two levels that have an outside ramp leading up to the second floor. Some buildings can be connected to the ground segment in other ways, for instance low garages where small objects stored closed to the building can cause the roof to appear to be connected to the ground.

Like steep slopes these buildings risk being classified as bridges because they are discontinuous to the ground at several locations and also contain points that make a smooth transition to the ground. As with the steep slopes, shadowing vegetation or shadowing nearby buildings may increase the probability of erroneous classifications. This is because the shadowing objects are removed during object removal thus creating a hole where no ground points are registered. As a result even points at roof edges appears to be smooth since there are no ground data available in the neighbourhood around the roof. Some industrial areas include complex structures matching the assumptions of bridges that might not be removed during object removal.

As explained in Section 3.2.1 dense vegetation with many layers cause problems with the iterative object removal and tend to leave the lowest layers in the ground model. These layers are raised above the ground and the low vegetation can appear to connect to the ground at several places thus matching the assumptions of bridges.

3.2.5 Finding potential bridge coordinates

A method for finding potential bridge coordinates was implemented to potentially reduce the search space. The method was not evaluated together with the bridge detection algorithms since it mainly was implemented to prove the concept of finding potential bridge coordinates. Another reason for not using this during evaluation of the bridge detection algorithm is that reduction of search space was not wanted because the objective also was to evaluate the number of false positives that appear in landscapes without bridges.

The finding of potential bridge coordinates suffers from one obvious flaw: it will only include bridges that span water and for example highway overpasses, bridges over railways and pedestrian bridges are not included in the results.

In hydrological modelling this may not be a significant problem since bridges that span water are most likely to hinder water flow, however in a more general setting bridges should be detected independently of whether they span water or not.

Instead of finding potential bridge coordinates it is possible to use existing bridge

databases to directly extract coordinates. BaTMan (Bridge and Tunnel Management) is an example of an existing bridge database covering bridges in Sweden. It covers $\sim$ 27000 bridges but information about the completeness and accuracy of this database have not been found.

Chapter 4

Conclusion

4.1 Conclusion

In this project a method for detecting bridges and classifying bridge points in airborne laser scanner data has been implemented and evaluated on an area of varying terrain types containing bridges of different sizes. The project shows that the ideas of detecting structures that are extensions of the bare earth as introduced by Sithole et al. in [6] can be transferred to an implementation and used for large scale detection and classification of bridges. It is able to detect bridges of varying shape and size in both urban and forested landscapes while registering only a small number of false positives in most landscapes. As shown in Table 3.1 the algorithm is able to detect bridges with a detection rate of around 90%.

A method for finding potential bridge coordinates in existing geographical data has also been implemented to show that it is possible to substantially reduce the search space when using bridge detection to correct hydrological simulations.

4.2 Possible Extensions

One of the things that were found to affect all parts of the algorithm was the object removal. The object removal implemented in this project was chosen for its ability to keep bridges in the digital terrain model. However it proved to perform worse than many other ground classification algorithms in successfully removing all object points. These errors were mainly the result of low lying object segments that were loosely connected to the bare earth.

A new object removal algorithm that better handles steep slopes and more accurately removes loosely connected object segments, while keeping bridges in the terrain model, would significantly decrease the number of false positives and therefore allow bridge detection to run with less restrictive parameters and potentially detect and classify more bridges accurately.

Another way to potentially improve classification result would be to change the assumption of what a potential bridge point is. Since the current assumption that bridges are raised may prevent some low lying bridges from being detected by the algorithm. To detect and classify these bridges, potential bridge points and bridges need to be detected based on some other measure than their elevation. One approach to identify these low lying bridges could be to use laser data together with satellite imagery or geographical data to identify the road and water network. By combining this approach with the algorithm described in this project a larger portion of the bridges may be detected.

The bridge detection algorithms can be integrated to work with existing bridge databases, not only to reduce the search space but also by using existing data about bridge heights and lengths to optimise parameters used during detection to improve classification results and minimise the number of false positives.

Bibliography

[1] E. Baltsavias, Airborne laser scanning: existing systems and firms and other resources, ISPRS Journal of photogrammetry & Remote Sensing 54 (1999) 164–198.

[2] H. Maas, G. Vosselman, Two algorithms for extracting building models from raw laser altimetry data, ISPRS Journal of photogrammetry & Remote Sensing 54 (1999) 153–163,.

[3] J. Hyyppä, O. Kelle, M. Lehikoinen, M. Inkinen, A segmentation-based method to retrieve stem volume estimates from 3-D tree height models produced by laser scanners, IEEE Transactions on Geoscience and Remote Sensing 39(5) (2001) 969–975.

[4] K. Kraus, N. Pfeifer, Determination of terrain models in wooded areas with airborne laser scanner data, ISPRS Journal of photogrammetry & Remote Sensing 53 (1998) 193–203.

[5] K. Marks, P. Bates, Integration of high-resolution topographic data with floodplain flow models, Hydrological Processes 14 (2000) 2109–2122.

[6] G. Sithole, G. Vosselman, Automatic structure detection in a point-cloud of an urban landscape, Proc. 2nd GRSS/ISPRS Joint Workshop on Remote Sensing and Data Fusion over Urban Areas, URBAN (2003) 67–71.

[7] G. Sithole, Segmentation and classification of airborne laser scanner data, Ph.D. thesis, Delft University of Technology (2005).
URL `http://repository.tudelft.nl/assets/uuid:64c82a6e-6db0-4457-9c4f-2446653e5b9d/ae_sithole_20050524.pdf`

[8] G. Sithole, G. Vosselman, Bridge detection in airborne laser scanner data, ISPRS Journal of photogrammetry & Remote Sensing 61 (2006) 33–46.

[9] R. Trias-Sanz, N. Loménie, Automatic bridge detection in high-resolution satellite images, Lecture Notes in Computer Science 2626 (2003) 172–181.

[10] Y. Wang, Q. Zheng, Recognition of roads and bridges in SAR images, Pattern Recognition 31 (1998) 953–962.

[11] R. Carlson, A. Danner, Bridge detection in grid terrains and improved drainage enforcement, Proc. ACM Symposium on Advances in Geographic Information Systems (2010) 250–260.

[12] J. Kilian, N. Haala, M. Englich, Capture and evaluation of airborne laser scanner data, International Archives of Photogrammetry and Remote Sensing 31 (1996) 383–388.

[13] A. Wehr, U. Lohr, Airborne laser scanning an introduction and overview, ISPRS Journal of photogrammetry & Remote Sensing 54 (1999) 68–82.

[14] C. Brenner, Aerial laser scanning, accessed 2011-03-14, (2006).
URL `http://www.ikg.uni-hannover.de/fileadmin/ikg/staff/publications/sonstige_Beitraege/Brenner_tutorialSommerSchool2006.pdf`

[15] M. de Berg, M. van Kreveld, M. Overmars, O. Cheong, Computational Geometry: Algorithms and Applications, 3rd Edition, Springer-Verlag, 2008, pp. 191–218.

[16] C. Shakarji, Least-Squares Fitting Algorithms of the NIST Algorithm Testing System, Journal of Research of the National Institute of Standards and Technology 103 (1998) 633–641.