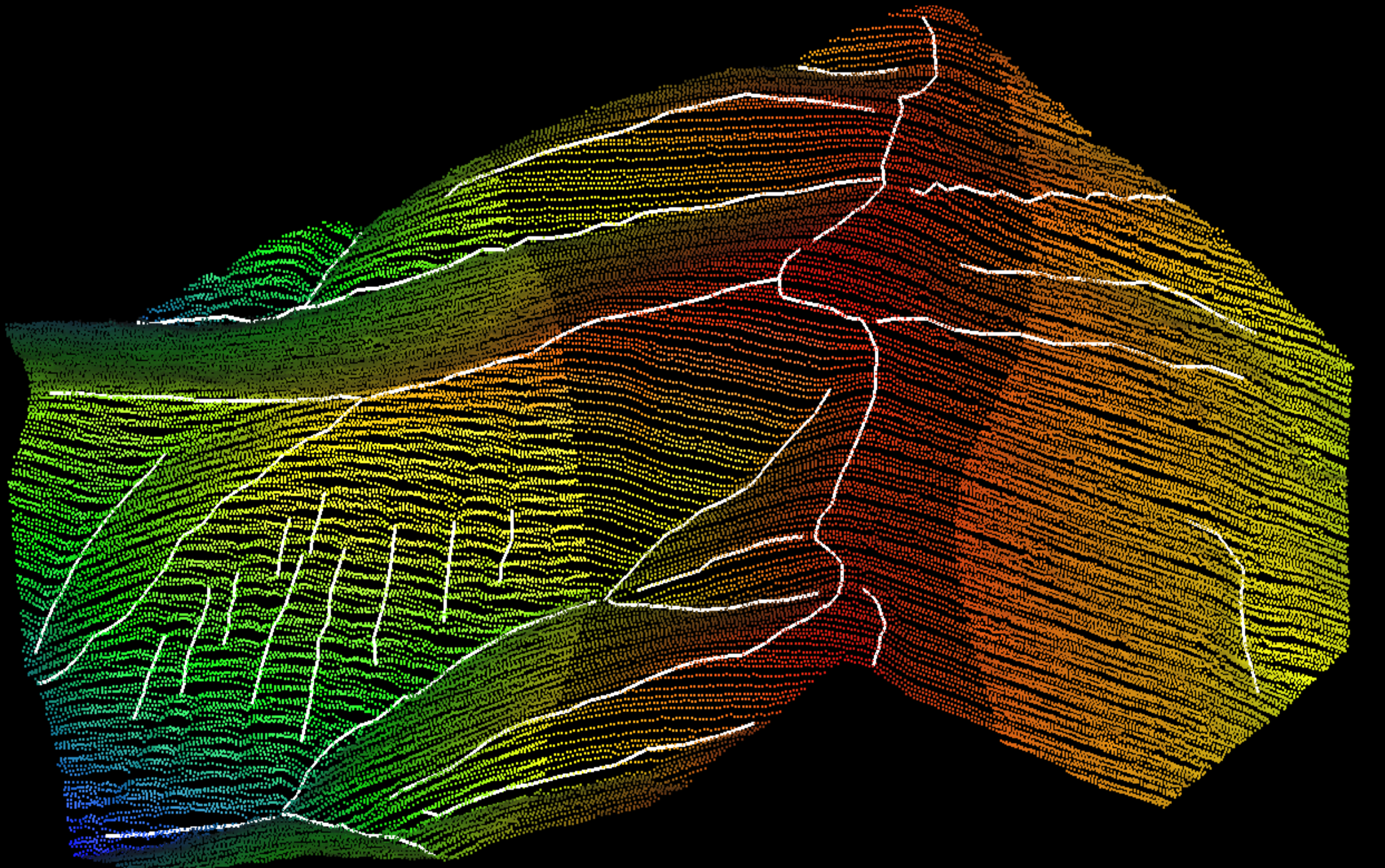
MSc thesis in Geomatics for the Built Environment

3D breakline extraction from point clouds with the Medial Axis Transform

Qu Wang, 2019



TU Delft

A MEDIAL AXIS TRANSFORM BASED METHOD FOR THE EXTRACTION OF 3D BREAKLINES

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of

Master of Science in Geomatics for the Built Environment

by

Qu Wang

May 2019

Qu Wang: *A Medial Axis Transform based method for the extraction of 3D breaklines* (2019)

ISBN 999–99–9999–999–9

The work in this thesis was made in the:

3D geoinformation group
Department of Urbanism
Faculty of Architecture & the Built Environment
Delft University of Technology

Supervisors:	Dr. Ravi Peters
	Dr. Hugo Ledoux
Co-reader:	Dr. Martijn Meijers

ABSTRACT

This project develop a new method to generate breakline from the point cloud directly with the MAT.
The breakline is a structured line of the object surface which has high curvature. Meanwhile, the reciprocal of the medial edge ball's radius can represent the point with high curvature, which is the fundamental idea of this project.
The key parts of the method are: **(a)** with the help of MAT detecting candidate points having high curvature; **(b)** connect the candidate points to get breaklines with graph theory or polynomial fitting.
The topology of the breakline is considered. In addition, this project also provides the polyline simplification and smoothing.
In the resulting breakline, 27/32 are true positive.
In general, the average correctness is 97% and 96%, the average quality is 90.67% and 91.19%, and the average completeness is 93.84% and 93.92% for valley and ridge. Most of accuracy parameters are above 80%. The accuracy is high.
As to the precision, the average polyline precision is 0.3 and 0.23, and the average vertex precision is 1.14 meter and 1.12 meter. Comparing with the point density $2pts/m^2$. 25 of 27 breaklines' polyline precision is better than 0.5. 16 of 27 breakline's vertex precision is better than 1.0; and 23 of 26 is better than 1.5. The precision is also good.

ACKNOWLEDGEMENTS

With this thesis done, it is the end of my master study. In these two years, many important people appeared in my life and gave me hands both in study and in life. First I would like to express my sincere gratitude to Dr. Ravi Peters. He provided me a lot of help during these months and always available. Also I would like to thank Dr. Hugo Ledoux who always provides valuable advice. I want to thank my co-reader Dr. Martijn Meijers for taking time to give advises and feedback on my thesis.
Special thanks to WeiXiao Gao, Yuxuan Kang and Hong Cao for the programming helps. Special thanks to Erik van der Wal for correcting my writing.
And great thanks go to all the teachers and all my fellow students from Geomatics, Yixin Xu, Teng Wu, Xin Wang, Shenglan Du, Pablo Ruben, Nikolaos Tzounakos, Vasileios Bouzas for all your advises both on study and life. I really value the time we spent together and things you taught me.
I would like to thank my friend Zhiwei Ai,Bin Wu, Mengya Wei, Weiran Li and Rui Yu, thank you for your support when I got depressed and distracted.
Thanks to Zhuo Qin, Jingying Yu, Luyu Jin, Boxuan Han, Bohan Tang for your accompany across the time zones.
I am so grateful to my parents and my families who support me to study abroad. Without their unconditional support, I cannot finish my study successfully.
Last, but not least, I am truly thankful to everyone who appeared in my life in last two years. Although I cannot list all of you here, I will always appreciate you and thank you for making the past two years the most colorful of my life.

Qu Wang
Delft May 2019

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACRONYMS

1 INTRODUCTION

1.1 MOTIVATION

A breakline presents a discontinuity of the earth surface, and it can be defined as a structured line of the object surface which has high curvature. In the natural world, a breakline can be a ridge, a valley or drainage line, and is essential for the description of the terrain surface. Breaklines were formed by long-term action of nature. In the geometry point of view, they constraint the surface height change. Thus it has a critical role in terrain modeling. Furthermore, a ridge has the property to separate two adjacent river systems, also called watershed. This physical property takes a unique position in the engineer construction.
Many applications require high-quality breaklines, such as geology scientific research Bryan and Moore [1977], surveying and mapping engineer, GIS application Zhu et al. [2001]. It is used as a constraint to generate a better DTM and data compression Linde et al. [1980]. In hydrological field, breaklines are essential for flood simulation and basin simulation models Bridge and Leeder [1979]. It can also monitor identical man-made objects, such as, dams or dikes, by comparing morphological features (breaklines) at different time epochs. As a consequence, developing an efficient method to generate high accuracy breakline is necessary.
The traditional way to extract breaklines is from raster data or contour line. With the appearance of Lidar point clouds, users can get more accurate and uniform ground elevation data. Besides, for a given laser pulse, a lidar system can record multiple returns, which makes filtering out some man-made object and vegetation possible. If a user converts point cloud data into raster or vector data, depends on the raster resolution or number of remained vertices, there is a loss of information. The accuracy is degraded since the exact altitude of the original points might be lost especially for the irregular surface like mountain and seafloor. So developing new approaches to extract breakline directly from point clouds is useful.

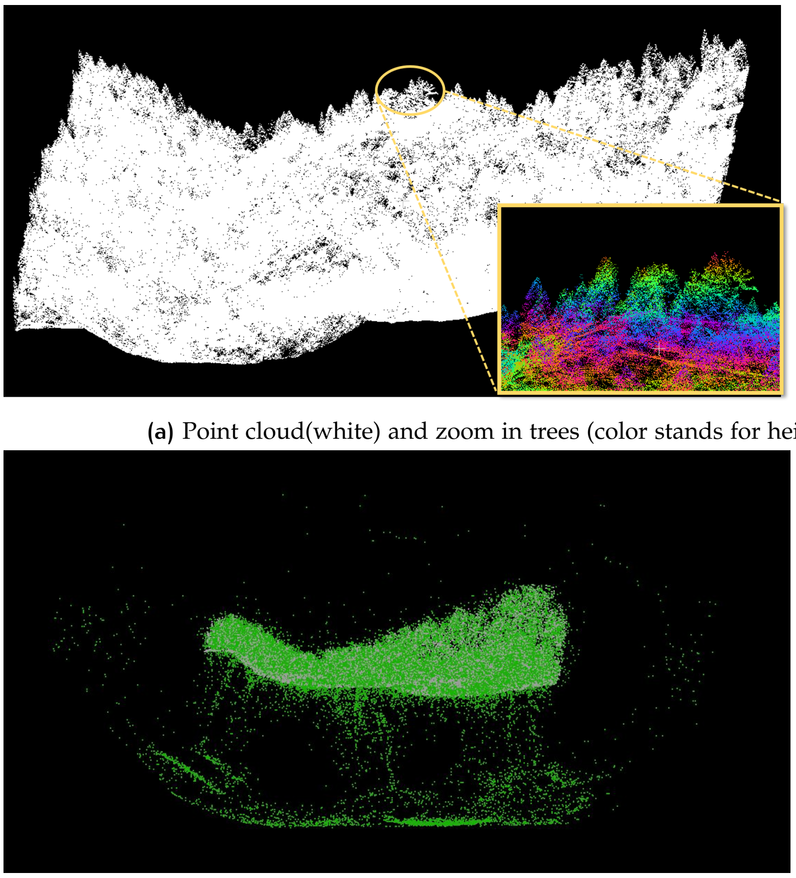
1.2 RESEARCH QUESTIONS AND SCOPE

The research question is "How to generate a 3D breakline directly from point cloud with the MAT?" And to solve this question, some sub question will carry out:

1. What is the link between Medial Axis Transform (MAT) and a breakline?
2. If MAT is suitable for generating a 3D breakline?
3. What are the requirements for the point cloud to extract breaklines?
4. Is it possible to automatically generate a breakline without user defined parameters?

The scope of this study will limited in natural ground surface, such as extracting ridge and valley for mountains, not include detecting edge of roof. Because the edge of roof is a "jump edge", which is a regular breakline, and has different geometry property to the ridges and valley (more details in Section 2.1.4).
The terrain can not contain trees (see Figure 1.1). Because the MAT can not find the structure for point cloud with tress (see Figure 1.1b). Moreover if there is a river or

lake in the ground surface, it will cause a hole in the point cloud. And this project can not manage holes.


(a) Point cloud(white) and zoom in trees (color stands for height)

(b) Point cloud(white) and MAT (green)

Figure 1.1: Terrain point cloud contain trees and the MAT result.

1.3 READING GUIDE

There are in total 6 chapters in this thesis, and the rest chapters are organised as following:
Chapter 2 first introduces some existing methods to generate breaklines. Then it explains some necessary concepts and functions of Medial Axis Transform.
Chapter 3 explains conceptual framework, workflow and principles of some algorithms related to this project.
Chapter 4 shows the details of implementation and experiments, also discuss the problems and possible solutions for each step.
Chapter 5 first shows the resulting breakline, then makes comparisons with the breaklines extracting from other methods. Finally validate the accuracy and the precision of the result. Chapter 6 summaries the result of this project, answers the research questions, The main issue is discussed in this chapter. Moreover the future works is given in the final chapter as well.

2 RELATED WORK

This chapter consists of two parts. The first part introduces existing methods to generate breaklines based on different input data: a raster (Section 2.1.1), contour lines (Section 2.1.2), a mesh (Section 2.1.3), or a point cloud (Section 2.1.4). Then a short summary of those methods is given in Section 2.1.5.
The second part (Section 2.2) explains the Medial-Axis Transform(MAT) in detail, i.e. the definition, the medial geometry, the algorithm to approximate the MAT and MAT segmentation.

2.1 EXISTING METHOD TO GENERATE BREAKLINE

Since the 1980s, how to automatically extract breaklines implicit in the digital terrain model is an essential study. Many researchers have developed different methods focusing with various purpose. Due to the physical properties of breaklines, it was especially popular for the segmentation of water bodies at the beginning. With the development of new technology to obtaining ground height, such as lidar, the point cloud has come to the stage. The point cloud is widely used in 3D building modeling. Thus roof edges detection as a special kine of breakline has been studied extensively recently.
Now there are four kinds of different input data that is able to generate breaklines: raster, contour line, mesh and point cloud. This section will provide an overview of existing method to get breaklines from these data types.
Because raster based method and contour line based method have different characteristics compared to point cloud base method, this section will only introduce several principal methods of these two methods, and focus on point cloud based method.

2.1.1 Raster input

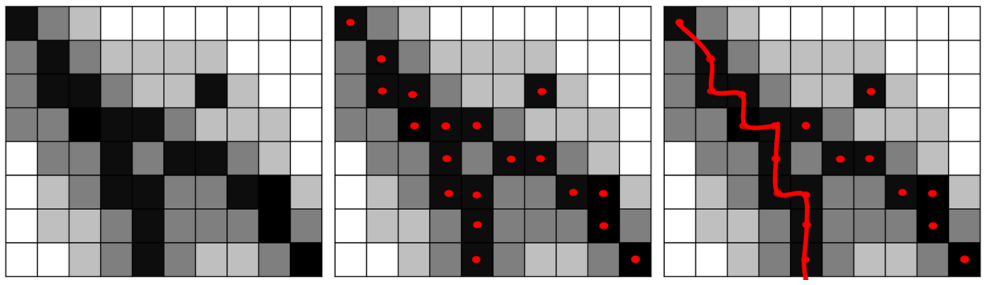


Figure 2.1: General process to extract breakline from raster image (the hue represents height)

The raster method extracts breaklines from a DEM. In this context, the local maximum curvature stands for the local change in slope. Martz and Garbrecht [1992] carried out a series of researches focusing on the D8 method, and Greysukh [1967]

implemented the 8-neighbours method to extract the catchment area from DEM. These two methods were similar, and they calculated the height difference between the central raster cell and eight local neighbours respectively to get the slope in 8 directions. The 8-neighbours method detects the slope variation, while D8 sorts the height difference and gives a feature code to the cell. D8 tries to find a code pattern to extract surface feature.
Peucker and Douglas [1975] found an easier method to detect pit and peak points to confirm ridge and valley. This study used a 2x2 moving window on DEM to compare altitude change between neighbours. But this method resulted in many isolated or incorrect pit or peak points, and can not find a complete breakline. Band [1986] enhanced this method by using binary refinement algorithm to get a single cell's width segments set, and then join the segments, finally accomplished a complete tree-like network of valleys. But it only works well on valleys, and is not adaptable to most other land form types.
O'Callaghan and Mark [1984] suggested another approach. This approach aimed at calculating each pixel's upstream pixel number, called the confluence cumulative value. If this value is larger than a threshold, the point was recorded as a pit. This method used eight connection points to find the flow direction, then it calculated the number of input path (NIP) and the number of output path (NOP). A pixel with NOP = 0 might be a pit or noise. Confluence area calculation started at where NOP = 0, and searches its next downstream pixel one by one. This approach was time-consuming. And different valleys might need different thresholds, using one single threshold is not suitable for all kinds of terrain. This approach was improved and changed to fit many other studies, such as Band and Robinson [1992].
Chorowicz et al. [1992] developed the Profile scan algorithm and hydrological flow modeling to extract drainage networks. This method analyses raster data in a vertical and a horizontal section to get characteristic points. Since it only analyses two orthogonal directions, the main drawback is that this method highly depends on the choice of section direction. Information on other directions might be missing.
Jenson and Domingue [1988] used digital simulation of overland water flow in 3D topographic surface. The watershed algorithm is also popular in some GIS software to extract breakline. Because the property of raster data, all the method suffer from one of the following: miss candidate characteristic points, not completely breaklines, sensitive to noise, over-segmentation.

2.1.2 Contour line input

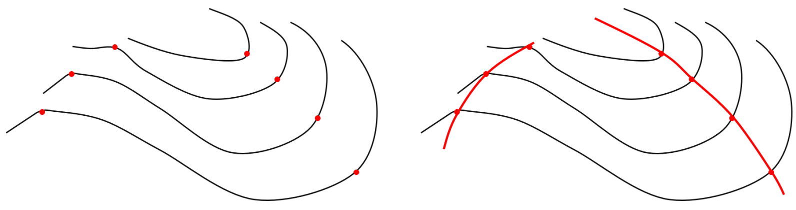


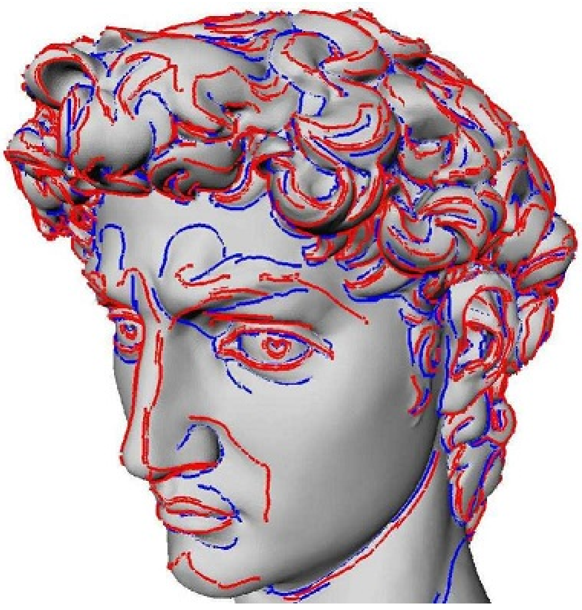
Figure 2.2: General process to extract breakline from contour line.

Contour lines are leveling lines that do not cover the whole ground surface. So to get the elevation of a point not on a contour line requires an additional process. Therefore, contour line is incomplete DEM data, but it is very suitable to extract breaklines.
The most common algorithm is the maximum curvature discriminant method [Chang et al., 1998]. The characteristic candidate points in the structure of contour lines have the local maximum curvature. From a geometric approach, the curvature is the second derivative of an object's surface. In other words, a characteristic point is where the contour line curve changes or has the maximum curvature. The approach to ex-

tract candidate points is similar to the theories of line simplification or compression line segmentation. After acquiring the feature point, this method connects them by the vector perpendicular to feature point's normal vector. Furthermore, Ebner et al. [1988] tracked the vertical vector of the contour line to analyses the surface water flow. The method may also produce breaklines with missing connections and isolated segments.

2.1.3 Mesh input

Except for traditional vector and raster input, the 3D mesh can also be used to extract breaklines. Based on the research of Cazals and Pouget [2005], CGAL includes a package[1] for approximating the ridges of a smooth surface discretized by a triangle mesh (see Figure 2.3).
It defines a ridge as a curve along which one of the principal curvatures has an extreme along its curvature line on a smooth surface. It requires the differential quantities associated to the mesh vertices as input.



red: ridges; blue: umbilics

Figure 2.3: Crest ridges on the David,model provided by the Digital Michelangelo Project

2.1.4 Point cloud input

A point cloud is a set of 3D points, that represents the boundary of an object. Now a point cloud is primarily perceived as a means for gathering detailed information about the surface of the terrain. It has a different character from the raster, contour line and mesh data. The point cloud is sparse data, and all the properties related to points (such as the normal of a point) are also sparse. Since point clouds are widely used and have different characteristics, researches focusing on detecting breaklines from the point cloud are necessary.
Different methods specialise in different types of breaklines, for example man made objects (building roofs) and natural object (mountain ridges and valleys, shorelines).

- Different edge types and characteristics

1 http://t.cn/AiCuocQk

As explained in Chapter 1, breaklines present the discontinuity of the terrain surface. The discontinuity can be classified as jump, crease or curvature edge (see Figure 2.4), and they have different properties (see Table 2.1).

Because the ”jump edge” has a clear ”gap” in range in the point cloud data, it

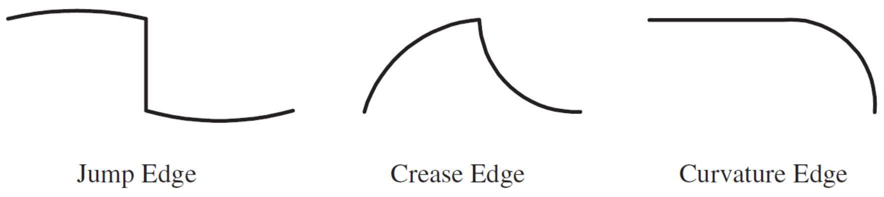


Figure 2.4: Different edge types [Ugelmann, 2000].

Edge type	discontinuity in	continuity in
jump edge	range	
crease edge	surface normal	range
curvature edge	curvature range and surface	normal

Table 2.1: Characteristics of edge types [Ugelmann, 2000]

can be detected using standard edge operators designed for intensity images such as the gradient, Sobel, Kirsch, Laplacian of Gaussian, and Canny edge operators [Ugelmann, 2000]. Furthermore, the ”jump edge” breaklines appear in buildings mostly (e.g. roof ridge). The building ridges can be derived by the intersection of two facade surface. Those man made roof ridges are more regular, and can be characterised as the combination of different simple geometries such as line segments, curve etc. In 3D building reconstruction, roof edge extraction has been studied extensively (Vosselman et al. [2001]; Schwalbe et al. [2005]; Verma et al. [2006]; Wang and Shan [2009]).
The ”crease edge” and ”curvature edge” are different from the ”jump edge”. They are the common breaklines in a natural terrain surface. They can not be summarised as combination of simply geometries easily and look more irregular. They require special methods to extract. This project focuses on these two type of breaklines in natural world.

- Generating breaklines from a point cloud

Ugelmann [2000] developed an edge-extraction based approach to find breaklines of dikes in dense laser dataset. The strategy is shown in Figure 2.5. It turns the point cloud into grey value raster image first. The approach is based on a hypothesis test to find points with extreme curvature values in the image, that broadly indicate breakline-regions. Then the regions are reduced to one pixel wide breaklines by means of non-maxima suppression taking into account maximal curvature direction, and one can get a raster breakline. The next step is turning the raster breakline into vector, and this yields rather zigzag breaklines. Finally, a cubic polynomial fitting method is used in x, y coordinates to get a smooth breakline. The result is demonstrated on a dense point cloud with 7 points/m^2, and shows a set of relatively distinct lines of a dike. The flow diagram shows that, this approach does not directly process the point cloud to extract breaklines. It turns point cloud into a raster first, and the core part (finding points with extreme curvature values) is in raster data. This method is still the area of image processing, and not a real 3D method.
Briese [2004] developed a technique to extract breaklines for dikes. This technique requires point cloud and a 2D approximate breakline as input. It creates a buffer

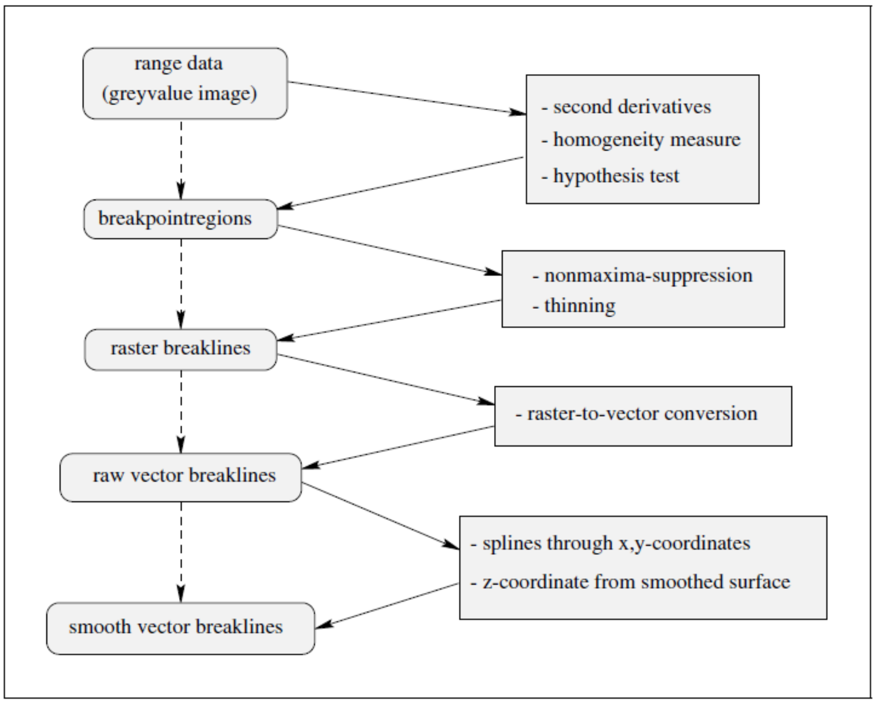


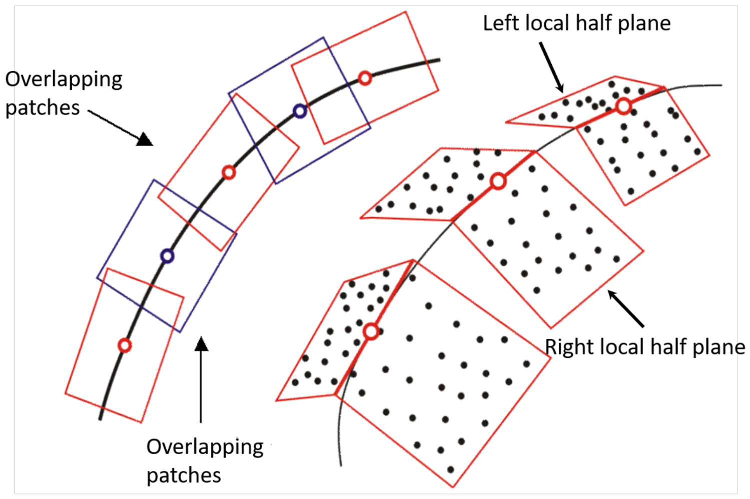
Figure 2.5: Flow diagram of Ugelmann [2000] proposed strategy

around the 2D approximate breakline in the *xy* plane. Along the 2D breakline a number of overlapping patches are created (see Figure 2.6 left). For each patch, the method fits a left local half plane and a right local half plane to the point cloud. The intersection of each pair of planes is a breakline line segment. If the 2D approximate breakline is missing, an operator needs to manually draw a vector which defines the direction of each breakline. Based on this vector, the patch can be found, and the intersection line segment direction is used to found the next patch. The breakline is growing gradually until it touches the boundary of the point cloud. The biggest disadvantage is the need of a 2D breakline as input or requiring a manually drawn point or line segment with orientation as start point for each breakline. It can not find breaklines that are not in the 2D input breaklines or missed by the operator. This technology is limited by the quality of input 2D breaklines or the operator. It is suitable for extracting pair vise structure line such as dikes. But for a mountain with complex breaklines, missing 2D breaklines in the input or too much manual intervention will be a problem. Furthermore, the breaklines of a dike changes gradually, but the resulting breakline has a strong change in the direction where two patches are overlapping.

Brzank et al. [2005] also focused on dikes, and he improved Briese's method. He solved the "strong change" problem by adding more patches at the "strong change" area.

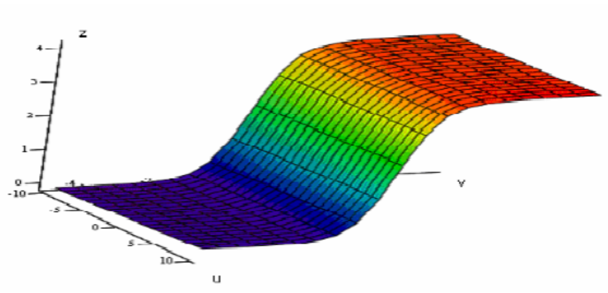
Brzank et al. [2005] proposed another approach to extract breaklines. It turns the point cloud into a raster image and applies an edge detection algorithm on it to derive the 2D approximation of the breaklines. Then a hyperbolic tangent surface model is estimate by a non-linear least-squares adjustment to express the coastal surface. The breaklines are derived from the surface model (see Figure 2.7). To be noticed, the *xy* approximate location of breaklines is estimated from a raster image. Hence, this method is not a real 3D method.

Pang et al. [2012] extracts valley-ridge lines from point-cloud-based 3D fingerprint models. They first apply the moving least-squares method to fit a local paraboloid surface and to represent the local point cloud area. On the basis of the fitting surface, they calculate the curvature and curvature tensors, and then detect potential valley-ridge points. Second, through statistical means of local neighbours, they

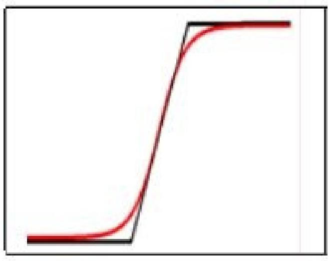


Left: Ground view of overlapping patches; Right: Perspective view of a reduced number of patches and half plane pairs.

Figure 2.6: Basic concept for breaklines extraction on point cloud by Briese [2004].



(a) Hyperbolic tangent surface.

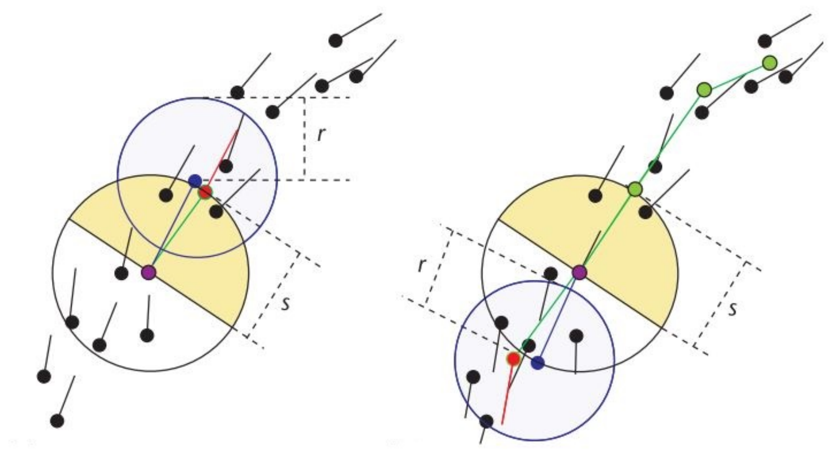


(b) Deriving breaklines from surface model.

Figure 2.7: Extracting a breakline from a surface model by Brzank et al. [2005]

project those points to the most likely valley-ridge lines. Third, by growing the polylines along the directions of the maximal principal curvatures they obtain the 3D valley-ridge lines (see Figure 2.8).

For different object, breakline detection are used in many field. For example,



Left: tracing in one direction of the seed point; Right: tracing in the opposite direction.
The purple point: the starting point; the blue point: the new sample point; The red point: the optimized new sample point.
r: the radius of the neighbor region of the newly generated sample point; s: a user-defined step size.

Figure 2.8: Connecting potential valley-ridge points along the directions of the maximal principal curvatures ([Pang et al., 2012]).

Rutzinger et al. [2006] outlines an object-based extraction framework and demonstrate it on the extraction of moraines. A preset definition of the phenomena distinctiveness dictates the search window and curvature related parameters. Features are extracted by skelatonization of high curvature regions.
Baruch and Filin [2008] detects receding shorelines by local extreme curvature from a laser point cloud. It uses a multi-scale approach that allows detecting different ridge realizations.

2.1.5 Summary

A breakline is a structure line with local extreme curvature in the continues surface, but a point cloud is discrete data and can not cover the whole space (discontinues in surface). From the existing methods, there are three ways turning discrete points to a continuous surface, which are actually interpolation methods:

1. Turning the point cloud into raster image;
2. Fitting a surface to the point cloud based on the shape of an object;
3. Finding local neighbours of a point, and using mathematical statistics.

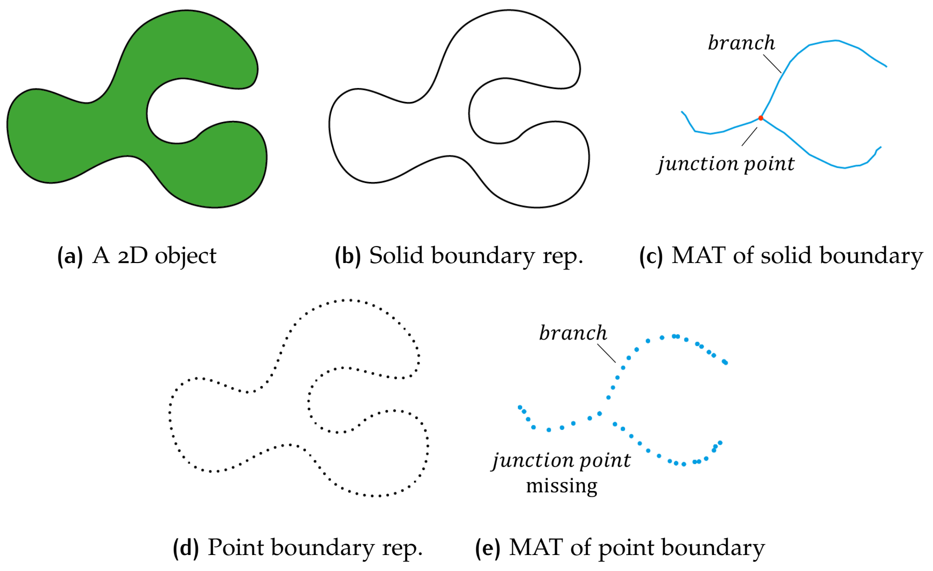
Turning into raster to generate breaklines can benefit from plenty of image processing methods and operators. Fitting a surface can result a smooth breakline. In those two methods, individual points must be compliant to the collective on a local scale and lose their individual property. It might lose accuracy or extreme values in the transforming (to raster or surface) process.
In general, all the methods contain two basic parts:

1. depending on the input, a method defines geometry or physical properties which can distinguish the discontinuity points/region in the surface (e.g. height difference in raster, curvature in contour line and mesh, rain flow accumulation etc.). Based on those geometry, the method extracts topographic characteristic points what might in the breaklines.
2. connect the characteristic points to get breaklines.

Examples of this process are shown in Figure 2.1 and Figure 2.2.

2.2 MEDIAL-AXIS TRANSFORM

The medial-axis transform (MAT) was first proposed by Blum [1967] in 2D at the beginning, and Peters [2018] applied the MAT to geographical point clouds. It is an alternative way to describe an object's shape, and to characterize it by a skeleton, which is different from the common boundary representation (see Figure 2.9).



(a) A 2D object **(b)** Solid boundary rep. **(c)** MAT of solid boundary

(d) Point boundary rep. **(e)** MAT of point boundary

Figure 2.9: Boundary representation and interior MAT

2.2.1 MAT definition

MAT is based on the definition of "Medial balls", which is a ball that fits completely inside the object's boundary, and does not contain any other ball that would fit inside it. The centre of the MAT ball is called a medial atom. And it has an important characteristic: a Medial ball touches the boundary in at least two points, called its feature points. Therefore, the distances between the medial atom and its feature points are equal. The MAT is defined as a set of medial atoms of the object.
Figure 2.9 shows a 2D object and its boundary representation (boundary rep), which is a solid closed polyline. And the interior MAT is also a set of solid polylines. One polyline is called a medial branch, and the point where one medial branch intersects with another one is called junction point, and it presents the topology relationship between different parts of the object.
If we turn the 2D boundary polyline into a set of 2D points, the interior MAT will also turn into a set of 2D points (see Figure 2.9). While the exact location of junction points might missing, the topology can still be estimated by where the medial balls overlap between different branches.

Moving from 2D points to 3D points, the 2D medial branch become a 3D "surface", called a medial sheet. Except for convex objects, objects have both an interior MAT and an exterior MAT (Figure 2.10). For an unclosed 3D surface, for example, the ground surface, the interior MAT is the points below the ground, and the exterior MAT is the points above the ground (Figure 2.11).

The greatest advantage of MAT method is supporting processing point cloud di-

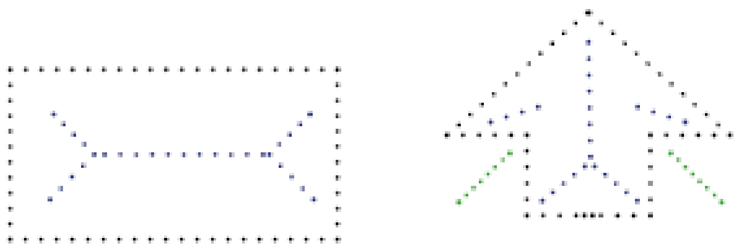


Figure 2.10: Interior MAT (blue) and exterior MAT (green) (black: objects surface)

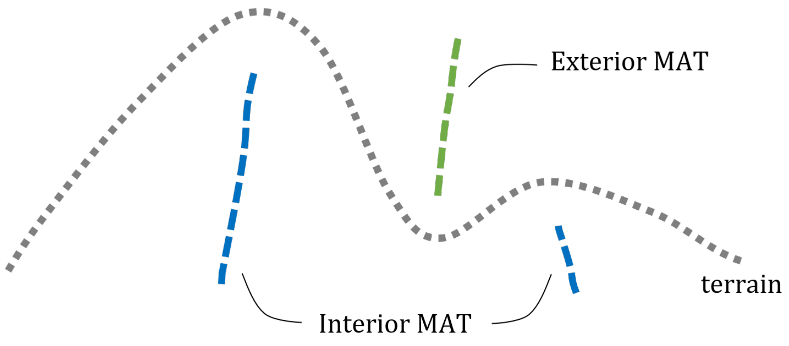


Figure 2.11: interior MAT and exterior MAT of ground surface

rectly, in other words, it is fully 3D. And there is no need to convert point cloud into raster or vector data. Therefore it will not lose information, and keep high accuracy.

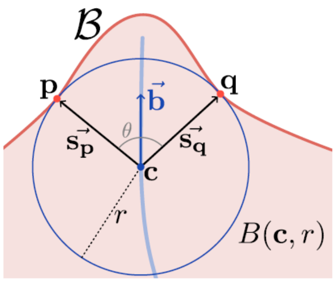
2.2.2 Medial geometry

In addition there are several geometry property for each Medial ball (shown in Figure 2.12):

- Spoke vector: the vector from central of Medial ball to a feature point;
- Separation angle: the angle θ between the two spoken vector;
- Media bisector: the angular bisector of separation angle θ

2.2.3 Shrinking-ball algorithm for medial atoms

In practice, the MAT can be approximated by the shrinking-ball algorithm, which requires an oriented point cloud as input. For one feature point, the centre of a medial ball must lie in the reverse direction of its normal vector, the centre of the ball can be calculated if the radius is known. With the decrease of the radius, the ball keeps shrinking until it touches another point and there is no other point inside



Symbol	Description
$B(\mathbf{c}, r)$	medial ball
$\mathbf{c}$	medial atom
r	radius
$\mathbf{p}, \mathbf{q}$	feature points
$\vec{\mathbf{s}}_{\mathbf{p}}, \vec{\mathbf{s}}_{\mathbf{q}}$	spoke vectors
θ	separation angle
$\vec{\mathbf{b}}$	medial bisector

Figure 2.12: Medial point and its geometry [Peters, 2018]

the ball. And this ball is a Medial ball, and the centre of the medial ball is the medial atom. The 2D shape result is shown as blue dots in Figure 2.13, and mountainous surface point cloud MAT result is in Figure 2.14. It is possible that a point may not touch any medial ball. Therefore, for the interior or the exterior MAT, the maximum possible number of medial atom is the same size of input point cloud.

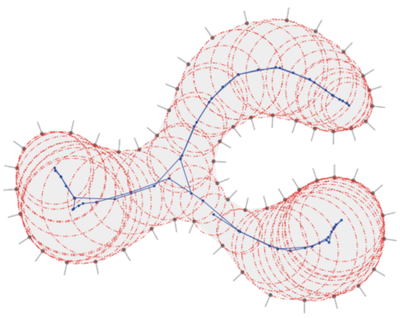
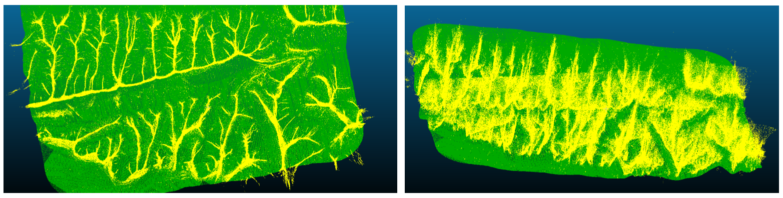


Figure 2.13: MAT point in 2D shape [Peters, 2018]



(a) Top view (b) Side view

Figure 2.14: MAT points for mountainous surface point cloud.

2.2.4 Medial segmentation and topology

Furthermore, the MAT processing also includes a segmentation method, whose function is to classify the medial atoms into different medial sheets by checking the similarity of the bisector. And the topological relationship between medial sheets is estimated by local neighbour analysis. This two kinds of geometry feature can be useful for extracting breaklines from a point cloud.
The segmentation process is implemented with a region growing algorithm, which starts at a random points as the initial seed point, using a validation function to check whether its local neighbour point is belongs to the same medial sheet as the initial point or not. And the points pass the validation will add to the seed point set. It iterates until all the seed points have been checked and there is no new seed point can be added. One can observe that for one medial sheet, the medial balls

have similar bisector. Therefore, the validation function will be true if the difference of bisectors is below a certain threshold. Also in one medial sheet, the medial balls are overlap with each other in a large areas (see Figure 2.15). So another validation function could be whether two medial balls overlap enough, more specifically, the sum of two balls' radius divided by the distance between the two medial atoms is larger than a threshold.

The medial sheets containing a very low number of medial atoms will be labelled

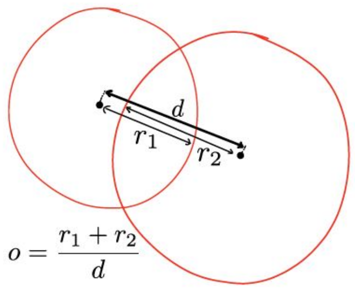


Figure 2.15: MAT segmentation: ball overlap [Peters, 2018]

as unclassified and eliminated from the segmentation result. This might because of ground noise, for example a large stone. Also if vegetation is captured a lot by the point cloud, especially for dense point clouds, whose normal vector will vary a lot in a small scale or within its neighbours. According to the shrinking ball algorithm, it might generate some medial atoms close to the ground surface with nonuniform bisector. Those MAT points are classified into ground noise.

As for the topology of medial sheets, for each pair of medial sheets, one can accumulate the neighbour points belonging to another sheet (and the idea is shown in Figure 2.16). If the accumulation number is large enough, these two sheets have a high probability being connected. In conclusion, the MAT method is able to im-

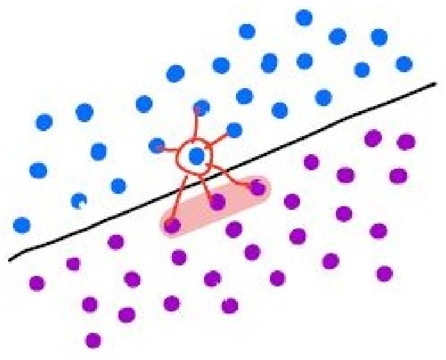


Figure 2.16: Accumulating neighbours [Peters, 2018]

plement on point cloud directly, and it provide medial geometry, segmentation and adjacent information, which are convenient to breakline generation. More important, from the top view of MAT result (in Figure 2.14), it has a "breakline-like" distribution, which means the MAT result indicates the breakline with some properties, which makes this method very suitable to solve the main research question.

3 BREAKLINE GENERATION WITH THE MAT

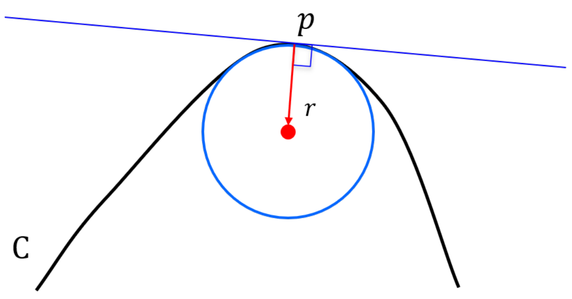
This chapter introduces the conceptual framework to extract breaklines based on the MAT method. The focus will mainly be on the theoratical aspects of this project. Section 3.1 explains the geometry link between the MAT and breaklines which is the fundamental idea of this project. This section also gives an overview of the workflow, briefly presents the steps. The next section explains how to estimate the orientation of a point cloud. Section 3.3 includes the MAT adjustment made in this project to solve the main research question. Section 3.4 explains how to extract points that might lie on a breakline. The following two sections present two methods to generate polylines as breakline from the extracted points. Section 3.5 presents how to find breakline with graph theory (minimum spanning tree and shortest path algorithm). Section 3.6 presents another method to find breakline by polynomial fitting. Section 3.7 connects the breaklines using the MAT adjacent relationship. Finally Section 3.8 provides an option to simplify or smooth the breaklines.

3.1 METHODOLOGY

As defined in Chapter 1, a breakline is a structured 3D polyline of the object surface which has high curvature. Curvature has different definitions in different areas of geometry. The well known Augustin-Louis Cauchy defines:

> the center of curvature of a curve C as the intersection point of two infinitely close normals to the curve, the radius of curvature as the distance from the point to C, and the curvature itself as the inverse of the radius of curvature.

Figure 3.1 shows a point p on a 2D curve C, and according to Cauchy's definition, the curvature of p is $\frac{1}{r}$. To be noticed, the tangent circle is also a 2D medial ball in the figure.
As for a point P on a 3D surface S, one can fit a ball touching the surface at point P with maximum radius r, and the curvature of this point P is still $\frac{1}{r}$.
This definition also links the breakline and the MAT together well. In Figure 3.2a,



curvature of point p is $1/r$

Figure 3.1: A 2D curve (blue circle is also a medial ball)

c_1, c_2, c_3 are the medial atoms of a terrain MAT, and m is the point on the breakline. Along the direction of c_1, c_2, c_3, the radius of the medial balls become smaller and smaller. The last medial ball c_3 is approximately tangent at point m and the radius of c_3 indicates the locally extreme curvature.
As further evidence, if the medial atoms are projected to the xy plane, the projected points are gathered around $m(x, y)$, which explains the reason why MAT result have a "breakline-like" distribution in Section 2.2 (see Figure 2.14).
In further observation of Figure 3.2b, one can fit a curve l passing the medial atoms. l intersects the ground surface at the ground point m with maximum curvature, which is a candidate point. The direction of l is approximately the direction of the bisector.
In addition, in terrain modelling, the interior and exterior MAT respectively corre-

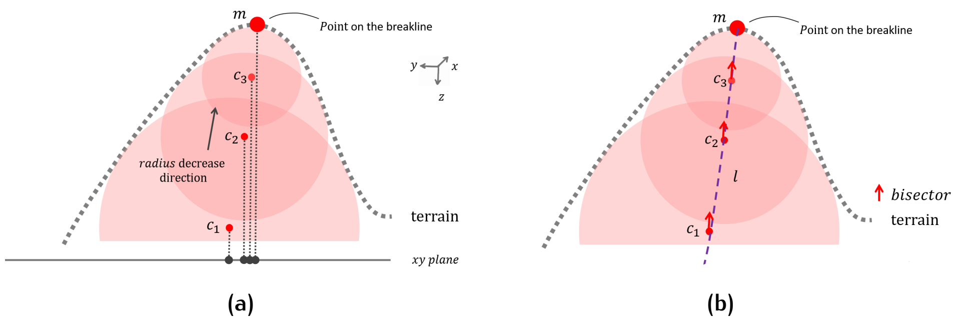


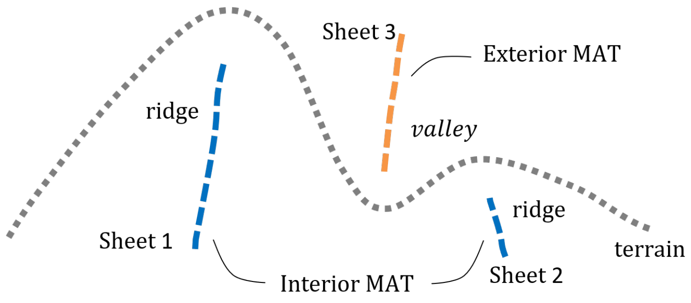
Figure 3.2: Medial ball and curvature (side view)

spond to the topographic ridge and valley features (see Figure 3.3), which are two main kinds of breaklines in a natural ground surface. The process to extract ridges and valleys are the same, and if not specified, all the explanation and images use the ridges and interior MAT result as example.
Furthermore, the medial atoms are segmented into medial sheets (explained in Section 2.2), and one medial sheet indicates one ridge or valley (see Figure 3.3).
In conclusion, those internal relations are the fundamental of this project.:

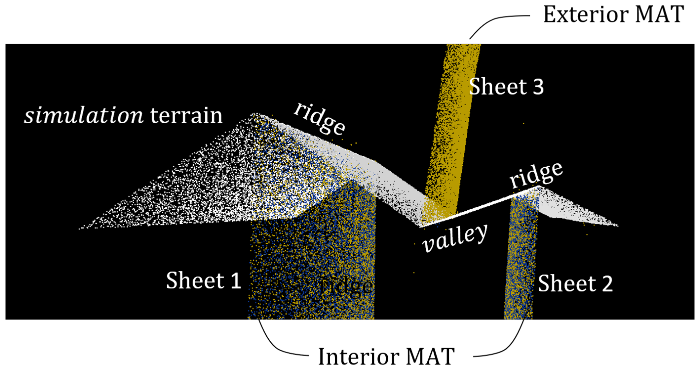
1. Radius and bisector of medial balls indicate the point on the breakline;
2. Interior and exterior MAT separate ridge and valley;
3. One medial sheet stands for one breakline.

With the help of MAT, this chapter proposes a new approach to generating breaklines from point cloud directly. Similar to the other methods (see section 2.1.5), the key part to generate breaklines consisting of two aspects: **(1)** finding candidate points that might lie on the breaklines, **(2)** connecting the points to form polylines to get breaklines.
Based on the fundamental idea, 7 steps are taken to get 3D breaklines directly from point cloud (shown in Figure 3.4):

1. Normal estimation for the point cloud;
   The normal vector of a point is a required input for the MAT ball shrinking algorithm.
2. MAT processing (preparing for step 3 and 5);
   In the second step, the plane area of the terrain is separated from the point cloud by MAT. This step proposes a new geometry and a new method for medial sheet segmentation. The adjacent relationship between medial sheets are separated into deep and surface adjacent.
3. Candidate point extract;
   In the third step, since a small radius of a medial ball corresponds to a high



(a) Concept.



(b) Simulation

Figure 3.3: Interior and exterior MAT indicate ridge and valley (side view)

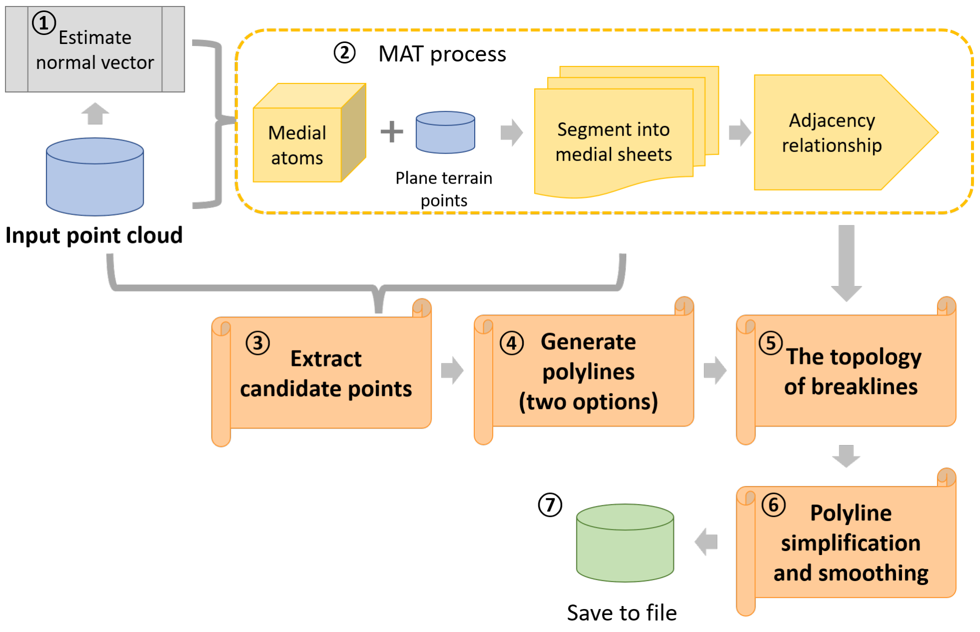

① Estimate normal vector
Input point cloud
② MAT process
Medial atoms
+
Plane terrain points
Segment into medial sheets
Adjacency relationship
③ Extract candidate points
④ Generate polylines (two options)
⑤ The topology of breaklines
⑥ Polyline simplification and smoothing
⑦
Save to file

Figure 3.4: Methodology flow chart

curvature, those medial balls are selected to calculate candidate point coordinates.

4. Polyline generation from candidate points;
   The fourth step provides two options to generate polylines.
   One option utilizes graph theory. It first connects the candidate points to its closet neighbour with the minimum spanning tree (MST) algorithm. Then it simplifies the MST into a polyline with the shortest path algorithm.
   The other option utilizes polynomial fitting. It fits a cubic polynomial function through the candidate points as a polyline.

5. The topology of breaklines;
   The fifth step estimates the junction point for adjacent breaklines with the MAT adjacent relationship between medial sheets.

6. Polyline simplification and smoothing.
   In the sixth step, the polyline can be simplified and smoothed. The simplification eliminates less important point in the polyline. For the smoothing four methods are compared to insert new point in the polyline.

7. Save polylines to a file.
   In the end, the result can be saved into a .obj file to share and open in other software.

3.2 NORMAL ESTIMATION

In a point cloud, the orientation of a point p is the normal vector of its local surface. In this study, the local plane is fitted by the k-nearest-neighbour of the point P (see Figure 3.5). Each point has two reverse orientations, one points above the ground, the other point under the ground. The first is for the exterior MAT and second for the interior MAT.

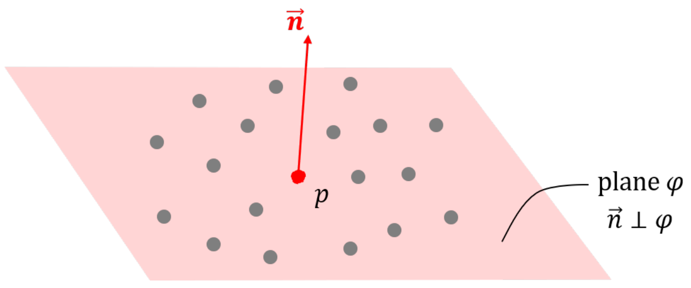


Figure 3.5: Normal estimate

3.3 MAT PROCESSING

This section extracts more information from the MAT to solve the main research question. First, it separates points in the plane area of the terrain from the point cloud by MAT. Second, it proposes a new metric and a new method to do sheet segmentation. Third, this section separate the medial adjacency into deep and surface adjacent.

3.3.1 Finding planar areas in the terrain with the MAT

In the MAT definition, a feature point touches a medial ball at a point p, but this does not always mean that the curvature of p is the reciprocal of the medial radius. As Figure 3.3b shows, the simulation terrain consists of three planar surfaces, the curvature should be 0.
Using medial ball's radius to find candidate point, a plane terrain area with a small medial ball may also be selected as a candidate point. This can cause problems.
In the MAT shrinking-ball algorithm (Section 2.2), some terrain points can not find a medial ball, and they are left as unshrunken points (see Figure 3.6). In the interior MAT, the unshrunken points cover planar areas in the terrain, and a few valley points, and vice versa for the exterior MAT. The unshrunken points can thus be used to remove the points that touch small medial balls in a planar areas.

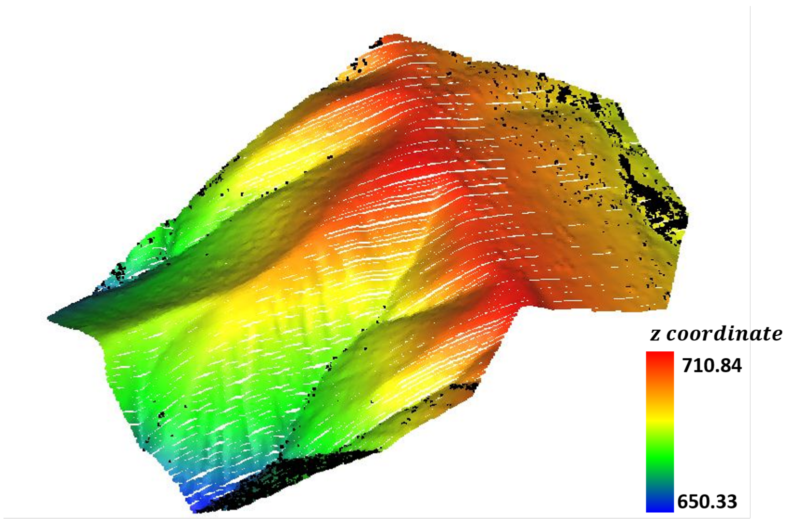


Figure 3.6: Terrain point cloud (colour) and in interior unshrunken points (black).

3.3.2 Medial direction and segmentation

Because one medial sheet corresponds to one breakline, a better sheet segmentation result will provide a better breakline result. In order to get a more accurate segmentation, this section introduces a new property called the medial direction, $\vec{n_c}$.
As shown in Figure 3.7, point c is the centre of the medial ball, and point p, q are the two feature points. Points p, c, q lie in the same profile/section plane. The medial direction, $\vec{n_c}$ is the normal vector of plane pcq, and it can be calculated by the normalization of the cross product of two spoke vectors:

$$\vec{n_c}' = \vec{s_p} \times \vec{s_q} \tag{3.1}$$

$$\vec{n_c} = \frac{\vec{n_c}'}{\left\|\vec{n_c}'\right\|} \tag{3.2}$$

Depending on the order of spoke vectors in cross product, $\vec{n_c}$ can have two directions: one points to the inside of plane pcq ($\vec{s_p} \times \vec{s_q}$), and the other points to the outside of plane pcq ($\vec{s_q} \times \vec{s_p}$). This project always use the spoke vector that equals to the reverse normal vector of point cloud as the first multiplier.
Since all the points in plane pcq have the same normal vector. For the medial ball close to the point on the breakline, the point is approximately on the same plane.

Because the line between the medial ball and the curvature, $\vec{n}_c$ can be approximately seen as the directions of maximal principal curvatures of the point ($\vec{v}$) on the breakline.

In Figure 3.8, each sheets has a different medial direction. Therefore instead of

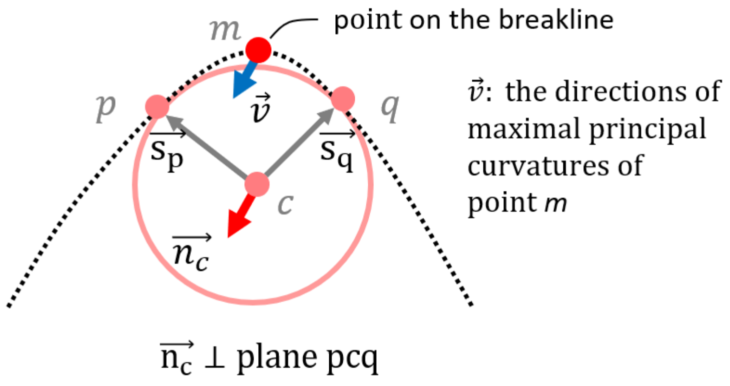
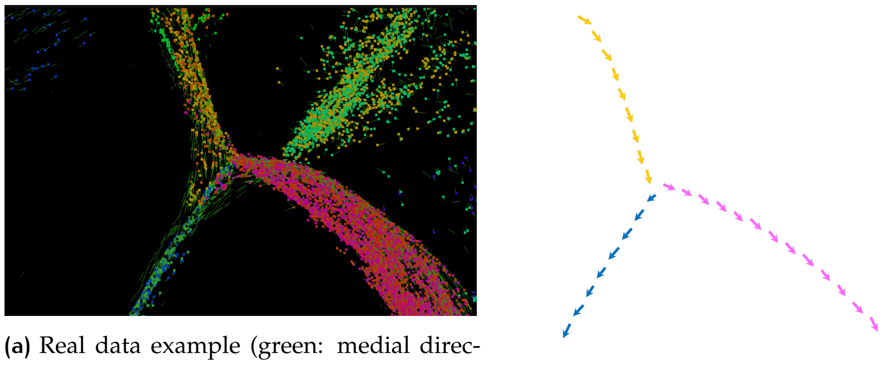

m
point on the breakline
p
q
$\vec{v}$
$\vec{s_p}$
$\vec{s_q}$
c
$\vec{n_c}$
$\vec{v}$: the directions of maximal principal curvatures of point *m*
$\vec{n_c} \perp$ plane pcq

Figure 3.7: Medial direction $\vec{n}_c$

using bisector difference as validation function (explained in Section 2.2), one can define a new segmentation validation condition by medial direction similarity :
If the difference between the medial direction of two medial atoms is below a threshold, they are likely to belong to the same medial sheet.
But the change of medial direction is gradually, not a "jump" change. Hence when



(a) Real data example (green: medial direction, orange, yellow, pink, blue: medial atoms belonging to different sheets). **(b)** Theoretical example (arrows: medial direction).

Figure 3.8: Medial sheets and its direction.

a sheet bends into two different sheets, the differences of medial direction will stay below the threshold (for example Figure 3.8a) , and it might be classified into the same sheet which is not correct actually.
One solution is using a smaller threshold. But a very small threshold will split a large medial sheet into many small medial sheets (over-segmentation). If the number of points in one segment class is too small to fit a medial sheet, it will be labelled as unsegmented.
Another solution is to combine two validation functions. Take the bisector and the ball overlap validation function (explained in section 2.2) into account, there are three independent validation functions:

1. the difference between the medial direction of two medial atoms is below a threshold $\vec{n}_{ct}$.

2. the difference between the bisector of two medial atoms is below a threshold $\vec{b}_t$.

3. the sum of two balls' radius divided by the distance between the two medial atoms is larger than a threshold bo_t.

There are four different combinations {1,2}, {1,3},{2,3} and {1,2,3}. If both two or three conditions are true, two medial atoms are likely to belong to a same medial sheet.
In this way, the strict thresholds can be avoid, and the validation function has more constraints. Figure 3.9 shows a segmentation result.
In addition one more threshold is introduced:
c_n : a threshold for the minimum number of points to firm a medial sheet;
The size of a medial sheet represents the scale of the breakline. If the total number of points in a medial sheet is below c_n, than this sheet might be caused by a big stone or noise in the ground, and it will not be taken for the further processing. The value of c_n is related to the resolution of original point cloud and the minimum breakline length a user want to detect.

3.3.3 Separating adjacent relationship into surface and deep adjacent

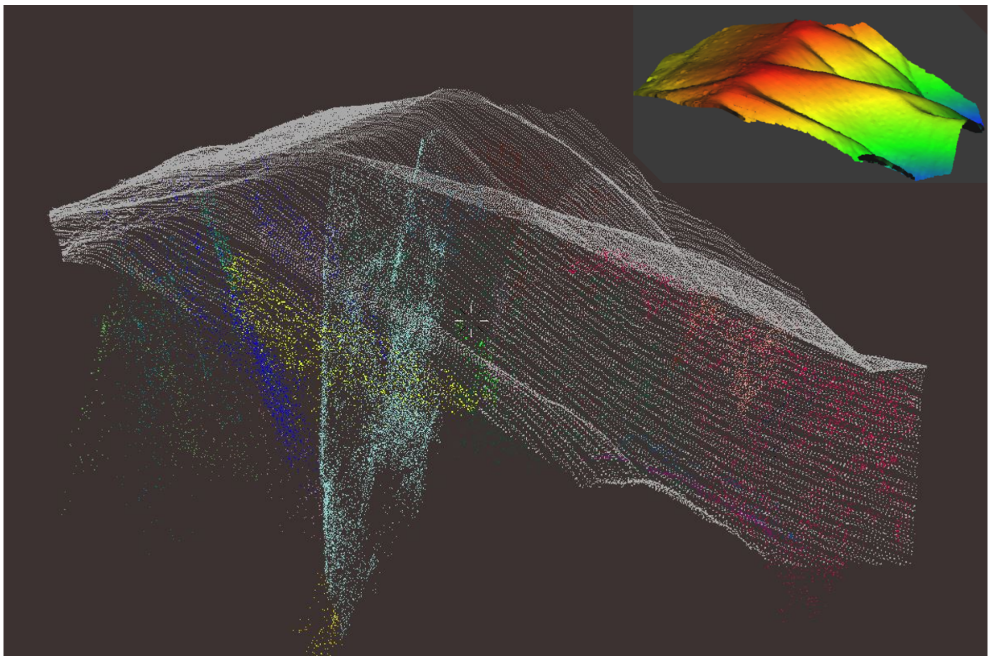
Observing carefully in the segmentation result, one can find two kinds of adjacency between medial sheets (see Figure 3.10). If two medial sheets are adjacent close to the terrain surface, they are surface adjacent. While if two medial sheets are adjacent far from the terrain surface, they are deep adjacent. Those two kinds of adjacency relationship can be distinguished by the shortest distance of the neighbour points (explained in Section 2.2.4) to ground surface.
Figure 3.11 demonstrates the possible causes of the deep adjacency. The deep adjacency shows the internal relation of the mountain, such as the mountain folded relationship or indicating the denudation process. It might be interesting for geology research. But the breakline is the ground surface geometry, and this project only need the surface adjacency.

3.4 CANDIDATE POINTS EXTRACTION

According to the fundamental idea explained in Section 3.1, a candidate point on a terrain surface is a point with a local maximum curvature. In the context of the MAT, a greater curvature corresponds to a smaller Medial ball. Therefore, the medial balls with a small radius are selected in each medial sheet, and the candidate points are calculated based on those medial balls (see Figure 3.12). Within this section, both finding medial balls and extracting candidate points are based on the segmentation result from the last section, more specifically, they are applied on each medial sheet.

3.4.1 Finding edge balls

From the observation of one medial sheet, medial balls closer to the terrain surface will have a smaller radius, and these balls are edge balls (see Figure 3.13 and Figure 3.2b). For the interior MAT (corresponding to ridge) these are the top edge balls, and for the exterior MAT (corresponding to valley) these are the bottom edge balls.
The small radius is relative, and different for every medial sheet. Even in the same medial sheet, the edge balls' radius is variable. That is why an edge ball can not be selected by filtering on medial radius. The edge balls need to be found from their location in the sheet.
This project tried two options to find the edge balls:


(a) Side view.

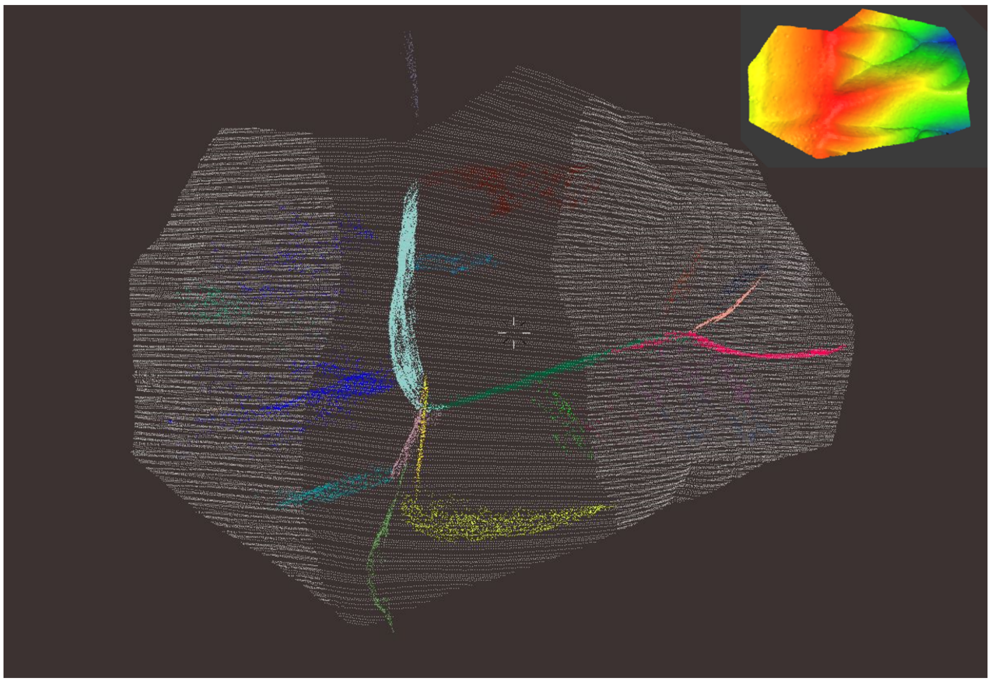
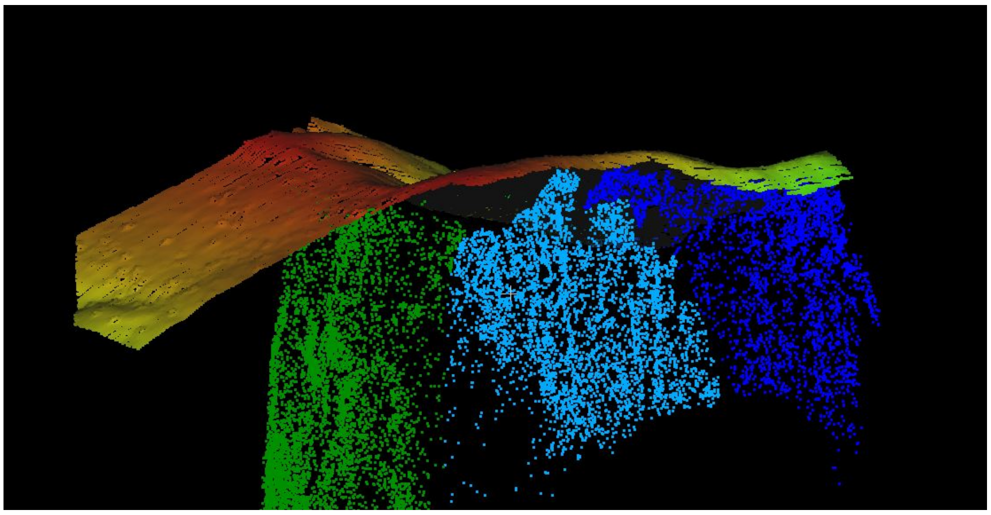
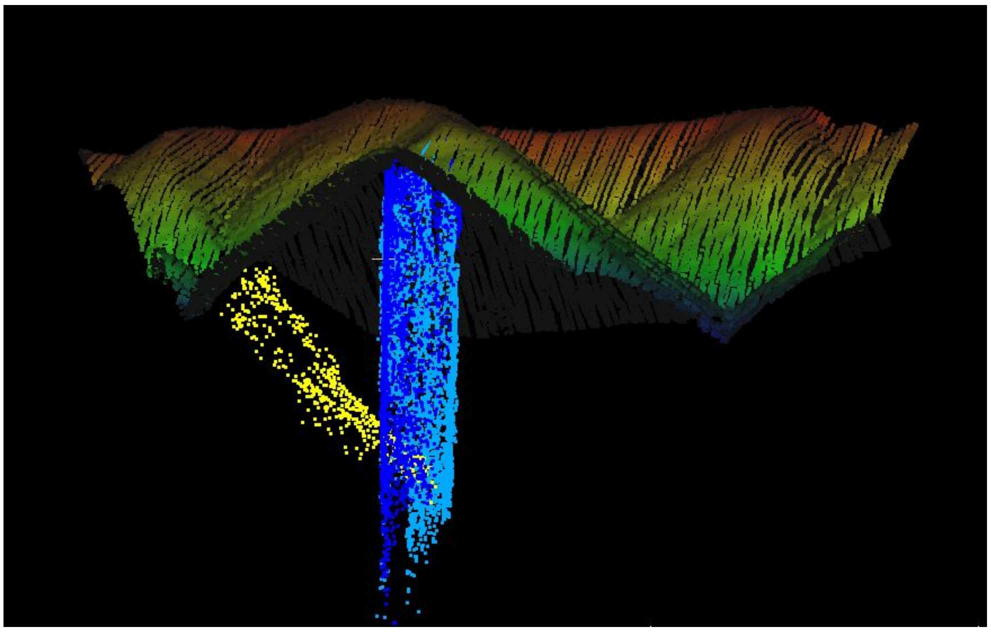

(b) Top view.

Figure 3.9: Segmentation result: medial sheets (colour) and terrain point cloud (white).

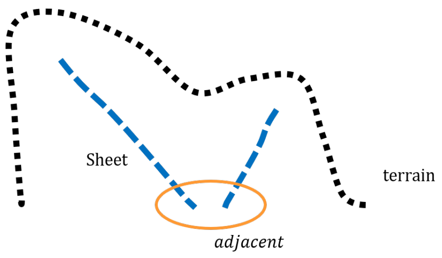


(a) Two pairs of medial sheets are adjacent close to the terrain surface.

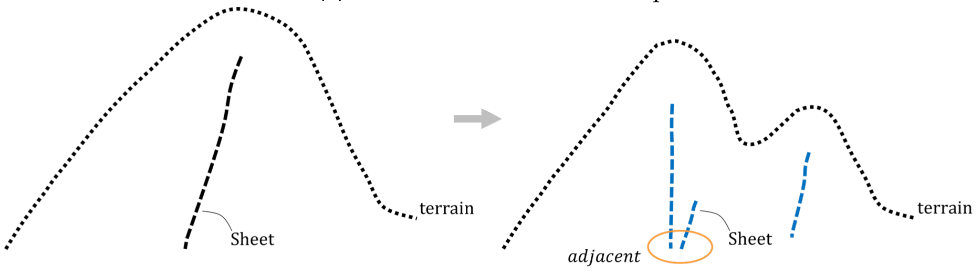


(b) Two medial sheets are adjacent far from the terrain surface.

Figure 3.10: Two kinds of adjacent between medial sheets.



(a) Mountain folded relationship.



(b) Residual of the denudation process.

Figure 3.11: The causes of the deep adjacent.

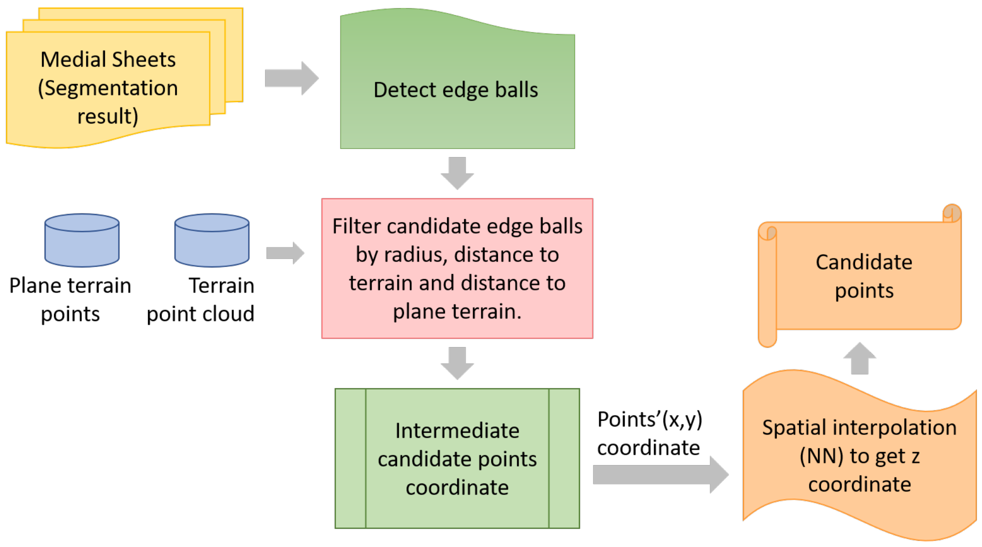

Medial Sheets (Segmentation result)
Detect edge balls
Plane terrain points
Terrain point cloud
Filter candidate edge balls by radius, distance to terrain and distance to plane terrain.
Candidate points
Intermediate candidate points coordinate
Points'(x,y) coordinate
Spatial interpolation (NN) to get z coordinate

Figure 3.12: Work flow for extract candidate points.

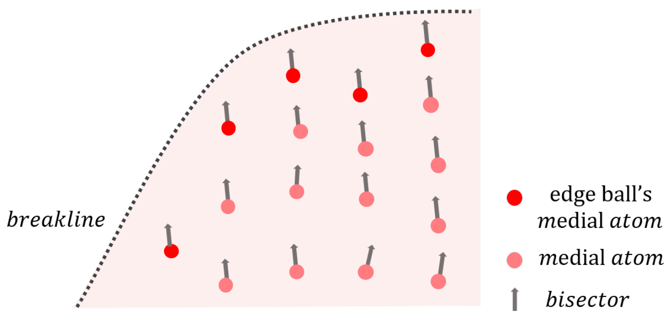


Figure 3.13: A medial sheet and its edge balls (side view)

1. Find traces of medial atoms approximately along the bisector $\vec{b}$ direction (see Figure 3.14) in one medial sheet. The last medial atom of one trace is the edge ball.

2. Evaluate separately for every medial ball whether it is an edge ball or not.

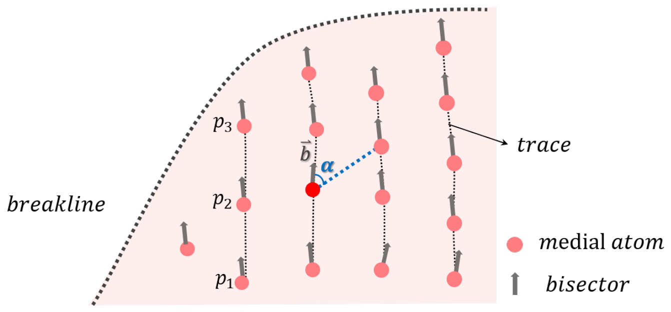


Figure 3.14: Profile section view for trace $p_1p_2p_3$ and deviation angle α

- **The first option: finding edge ball through traces**

In Figure 3.14, the medial bisector of a medial atom p_1 always approximately points to the next medial atom. The radius of p_2 is always smaller than p_1. Then p_2 might be able to find a next neighbour (p_3) in its bisector direction, until there is in no point in the current point's medial bisector direction. $p_1p_2p_3$ is a trace, and its end point is the required edge ball point.
Since p_2 is not exactly colinear with the bisector of p_1, a deviation angle α is introduced (shown in Figure 3.14). α is defined as the angle between the bisector of the current medial atom c and the vector starting at c ending at another medial atom c'.
In order to realize this idea, the FindTraces algorithm is implemented, see Algorithm 3.1.
A trace starts at a medial ball with a big radius, and along the trace is growing direction, the radius keeps decreasing. To make sure a trace start growing at a point with a large radius, a priority queue of medial atoms with a descending order on radius is used.
For a trace, the current last point is p, j is one of p's neighbour, the deviation angle of p, j is α_{pj}. With the user defined maximum deviation angle α_t, j is the next point in the trace if it satisfies the flowing three requirements:

1. j is not in other trace;

2. the radius of j is smaller than the radius of p;

3. $\alpha_{pj} < \alpha_t$.

If j meets all the requirement, it will be added to this trace and become the new current last point for the next iteration. If no neighbouring point meets all the requirements, the trace finishes at p.
In the FindTraces algorithm, for each point p, there might more than one point meeting those three requirements, and they are "valid points". There is a competition of "closest to bisector first" and "closest to point p first" to be the next point.
"Closest to bisector first" selects the point that is closest to the direction of the bisector of p as the next point among all the valid points. This is described by the deviation angle α, the smaller α the closer to the bisector. Algorithm 3.1 (line 16 - 20) implements this strategy.
But "closest to bisector first" may cause a problem. In Figure 3.15a, if both vp_1 and vp_2 are valid points, the further point vp_2 has a smaller deviation angle. With "close to bisector first" strategy, vp_2 is chosen as the next point, while the true next point vp_1 is missed . vp_1 will become an isolated point or be put into the wrong trace.

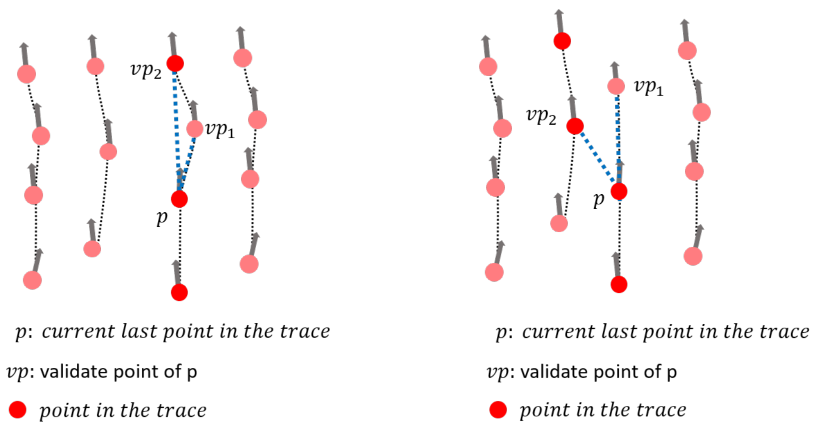
Algorithm 3.1: FindTraces (*MATdata*, *α_thresh*)

```
Input: $MATdata$: Medial ball $i$ belonging to one sheet consists of its
        geometry:
$c_i$: the $x, y, z$ coordinate of medial atom $i$,
$r_i$: the radius of of medial atom $i$,
$\vec{b_i}$: the bisector of medial atom $i$,
$\alpha_t$: the threshold for deviation angle.
Output: $AllTrace$: lists of medial atoms in one trace
1  $pq \leftarrow$ a priority queue of $c_i$ with the descending order of radius;
2  $kd \leftarrow$ a KD_tree of $c_i$;
3  $AllTrace \leftarrow empty$;
4  while $pq$ is not empty do
5      $p \leftarrow$ pop the first element $c_i$ with the largest radius in $pq$;
6      $next \leftarrow True$;
7      $a\_trace \leftarrow empty$;
8      while $next$ do
9          $a\_trace$ Add $p$;
10         $pq \setminus p$;
11         $next \leftarrow False$;
12         $Neighbours \leftarrow FindNeighbourPoints(p, kd)$;
13         $candidate \leftarrow Neighbours \cap pq$ & & $r_{Neighbours} < r_p$;
14         if $candidate$ is empty then
15             continue;
16         $\alpha \leftarrow$ a list of deviation angle for each pair of $(p, candidate_i)$;
17         $p_i \leftarrow$ the point with $min(\alpha)$;
18         if $min(\alpha) < \alpha_t$ then
19             $p \leftarrow p_i$;
20             $next \leftarrow True$;
21     $AllTrace$ Add $a\_trace$;
22 return $AllTrace$
```

"Closest to point p first" selects the point that is closest to the point p as next point among all the valid points. Then the 16th - 20th line of Algorithm 3.1 will become:

```
1 sorting the candidate by the distance to $p$ in ascending order;
2 for point $c$ in candidate do
3     $\alpha_{pc} \leftarrow$ the deviation angle of $p, c$;
4     if $\alpha_{pc} < \alpha_t$ then
5         $p \leftarrow p_i$;
6         next $\leftarrow$ True;
```

With the "close to point first" strategy, an unsuitable α_t will cause a problem. In Figure 3.15b, both vp_1 and vp_2 are valid points and vp_2 is closer to point p. A large α_t will result in vp_2 to be the next point, while the true next point is vp_1. The trace thus bends to wrong direction.
To get the best next point, this strategy needs a small α_t. But α_t is different for every medial sheet, and there is no good way to estimate α_t. Using a strict α_t might result in no valid points in some sheets.
In conclusion, to find a complete trace there needs to be a balance between the "closest to bisector first" strategy and the "closest to point first" strategy, which makes the problem even more complex. The traces found by "close to point first" for one sheet are shown in Figure 3.16. There are a lot of short traces and they end far away from the terrain surface. Still, no easy way to find a perfect complete trace can be found.


vp_2
vp_1
p
p: current last point in the trace
vp: validate point of p
● point in the trace
vp_1
vp_2
p
p: current last point in the trace
vp: validate point of p
● point in the trace

(a) Closest to bisector first

(b) Closest to point first

Figure 3.15: Problems in finding trace.

- **The second option: finding edge ball by evaluating every medial atom separately**

The second option is evaluating for every medial ball whether it is an edge ball or not. To realise the idea, an edge ball detection algorithm is shown in Algorithm 3.2. Different from Algorithm 3.1 that tries to find the next point in the trace, the EdgeBallDection algorithm does the opposite. It traverses every medial atom, if a point p does not have any point can be validate as "next point", than this point is an edge point. And for each neighbour n the validation criterion are:

1. the radius of n is smaller than the radius of p;

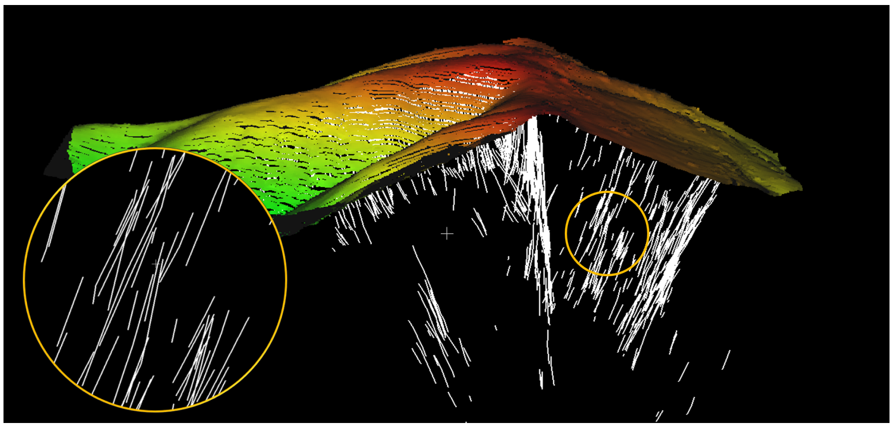
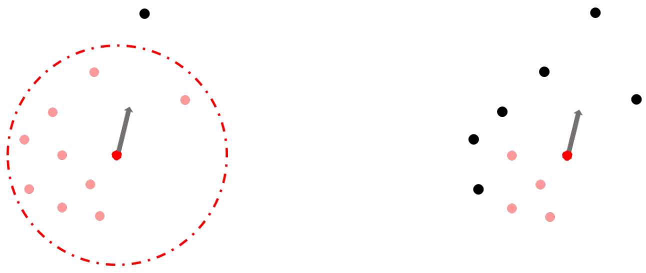


Figure 3.16: Profile section view for traces (white) and terrain point cloud (color stands for height).

2. α_{pn} is smaller than the maximum deviation angle threshold, α_thresh.

It starts at a random point of the medial atoms C of a sheet, i.e. c_i. First it finds the local neighbours of c_i, and if c_i does not have any neighbour, the point is probably far away from the medial sheet, and c_i will not be considered.
The algorithm sort the neighbours by the distance to c_i, and only keep the neighbours that satisfy criterion 1. Then it traverses each local neighbour n, and breaks when finding the "next point", more specifically, the deviation angle between c_i and n is below threshold t_a. If c_i does not have any "next point", in other words, is an edge ball point, c_i is the edge ball's centre, and will be added to the output.
There are two ways to find the neighbours of a point. One is searching the neighbours in a certain radius, t_r, and the other one is finding the k-nearest-neighbours. The difference between the two methods is summarised in Table 3.1. Both two options require the user to define a threshold. Different threshold values will affect the result and computation time. Both a small search radius and a small number of N-nearest-neighbours might result in no neighbours being found, and no edge balls. The "search radius" will cost more execution time, while it is capable to find neighbours in all directions. The "KNN" takes less execution time, but if there are plenty neighbours gathering in one direction this option might not able to find the next visit point (see Figure 3.17).
Table 3.2 provides a test on both two options. The total number of input Medial



(a) small radius neighbour search **(b)** 5-nearest-neighbour search

red: query point; pink: selected neighbours; black: unselected neighbours, gray: bisector.

Figure 3.17: radius neighbour search and k-nearest-neighbour search.

points is 336839, and besides the mentioned threshold, all the other processing is the same. Figure 3.18 compare the output of "20NN" and "search radius = 30m ". The output of "20NN" still has a sheet-like point distribution. Thus, in practice algorithm 3.2 incorporate the search radius method.
To avoid missing neighbours, the search radius should be a larger value. This

Algorithm 3.2: EdgeBallDetection (*MATdata*, *α_thresh*,*r*)

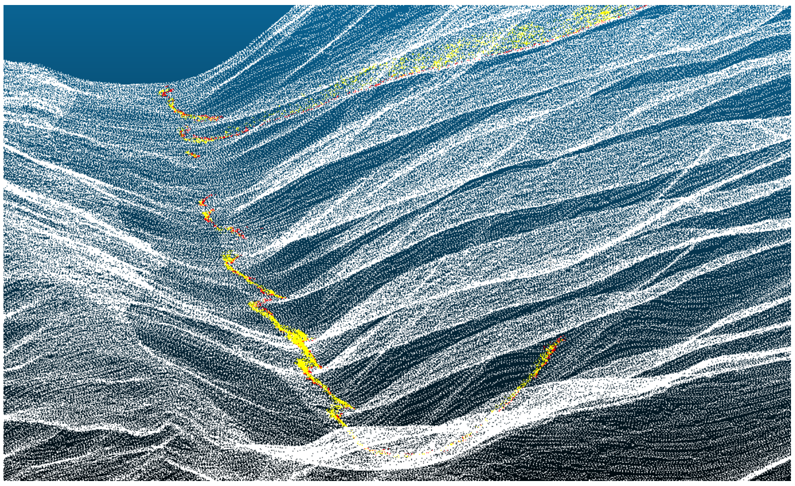
```
Input: $MATdata$: Medial ball $i$ belonging to one sheet consists of its
       geometry:
$C$: the $x, y, z$ coordinate of medial atom $i$,
$r_i$: the radius of of medial atom $i$,
$\vec{b_i}$: the bisector of medial atom $i$,
$\alpha_t$: the threshold for deviation angle,
$r$: Search radius
Output: $P_e$: a set of extreme balls' medial atoms
1  $kd \leftarrow aKDtreeofc_i$;
2  for each point $c_i$ in $C$ do
3      $Neighbours \leftarrow FindNeighbourPoints(c_i, kd, r)$;
4      if $Neighbours$ is empty then
5          continue;
6      else
7          $counter = 0$;
8          sort $Neighbours$ by ascending distance to $c_i$;
9          remove points with radius larger than $r\_ci$ in $Neighbours$;
10         for each point $n$ in $Neighbours$ do
11             $\alpha_n \leftarrow$ the deviation angle of $c_i$ and $n$;
12             if $\alpha_n < \alpha_t$ then
13                 break;
14             else
15                 $counter + +$;
16         if $counter ==$ size of $Neighbours$ then
17             $P_e$ Add $c_i$;
18 return $P_e$
```

	Radius	KNN
Required define threshold	radius [meter]	k [integer]
execution time	longer	shorter
small threshold results in	no neighbours	no neighbours
problems		can not deal with plenty neighbours gathering in one direction (see Figure 3.17)

Table 3.1: The differences of two methods for finding neighbours

	Number of edge balls (n_e)	n_e/n
10NN	261376	77.6%
20NN	188486	55.96%
Search radius = 30 m	61512	18.26%

Table 3.2: Number of edge balls with two methods ($n = 336839$ number of the input medial atoms)



red: search radius = 30m; yellow: 20 nearest neighbours; White: terrain point cloud.

Figure 3.18: Result of two methods.

returns more neighbours, but after sorting the neighbours by the closeness to the query point, the neighbour traversal will stop at the closest neighbour that meets the deviation angle threshold, and does not go through every neighbour. This reduces the running time. Moreover, the MAT atom density depends on the point cloud resolution, therefore using n times the resolution as search radius is logical.

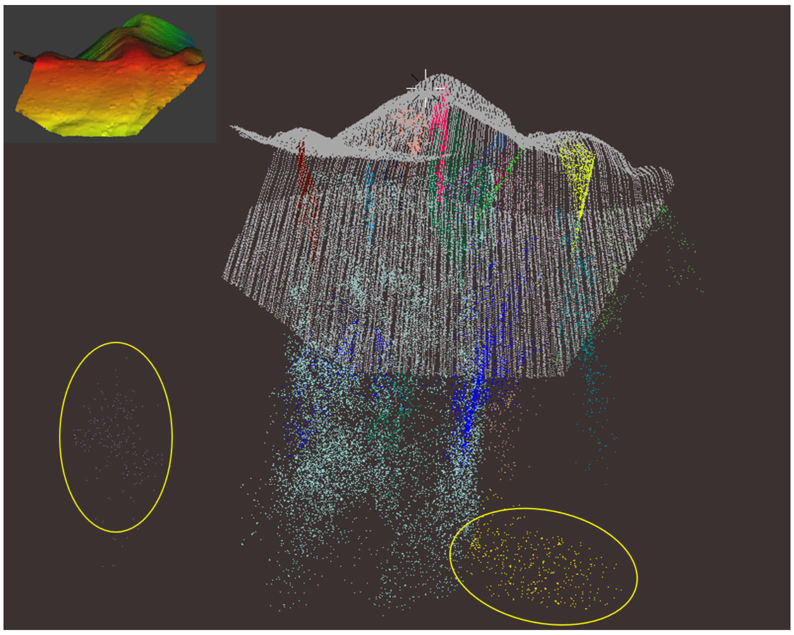
3.4.2 Filtering edge balls

To get better edge balls, some restricting conditions are used to filter edge balls.
First, since the radius and the curvature are reciprocal of each other, the maximum radius is used to remove edge balls with low curvature. Close to the ground, some ground noise will result in small radius medial balls, and they need to be filtered out too. Hence, the minimum radius is used. Therefore, the edge balls are limited by minimum and maximum radius.
Second, there are some points far away from the terrain surface point cloud. This might caused by the edge ball detection that is influenced by the shape of a medial sheet (see Figure 3.19), the points in the highlighted circle might also contribute to edge medial balls. They can be removed by the distance to the terrain surface.
Third, in planar area, the unshrunken points are used to remove edge balls by projecting the unshrunken points and edge ball atoms to the xy plane, if the distance between the edge ball atom and the closest unshrunken point is too small, the edge ball will be eliminated too.

3.4.3 Calculating candidate point coordinates from the remaining edge balls

This section is about calculating candidate point coordinates from the remaining edge balls (see Figure 3.20). According to Section 3.1, in an edge ball C, the candidate point m is the intersection of the bisector $\vec{b}$ and the terrain surface.

As explained before, the bisectors close to the ground are influenced by small ground objects (or noise), and the edge balls' bisectors varies more than others. To reduce the noise, the weighted average bisector $\vec{b_{avg}}$ is used.
For the medial atom of an edge ball p



colour points: medial atoms belonging to different sheets; White:terrain point cloud.

Figure 3.19: Medial sheet

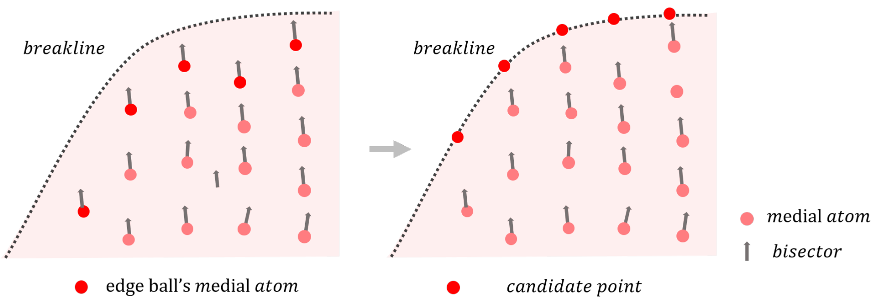


Figure 3.20: Edge balls to candidate points

The k-nearest-neighbor medial atoms of p in the same medial sheet are found

$$weight_i = \frac{r_i}{\sum_{i=1}^{k} r_i} \tag{3.3}$$

$$\vec{b_{avg}} = \sum_{i=1}^{k} \vec{b_i} \times weight_i \tag{3.4}$$

And for further calculation, $\vec{b_{avg}}$ needs normalization, with

$$\vec{b_{avg}} = (x_b, y_b, z_b) \tag{3.5}$$

the normalized the bisector $\vec{b_n}$ is found as

$$\vec{b_n} = \frac{(x_b, y_b, z_b)}{\sqrt{{x_b}^2 + {y_b}^2 + {z_b}^2}} \tag{3.6}$$

The comparison of whether using the weighted average bisector $\vec{b_{avg}}$ or not is in Figure 3.21. It is clear in the image that the candidate points are gathering to a better smooth line with $\vec{b_{avg}}$.

There are four methods to calculate the coordinates of candidate point m (see

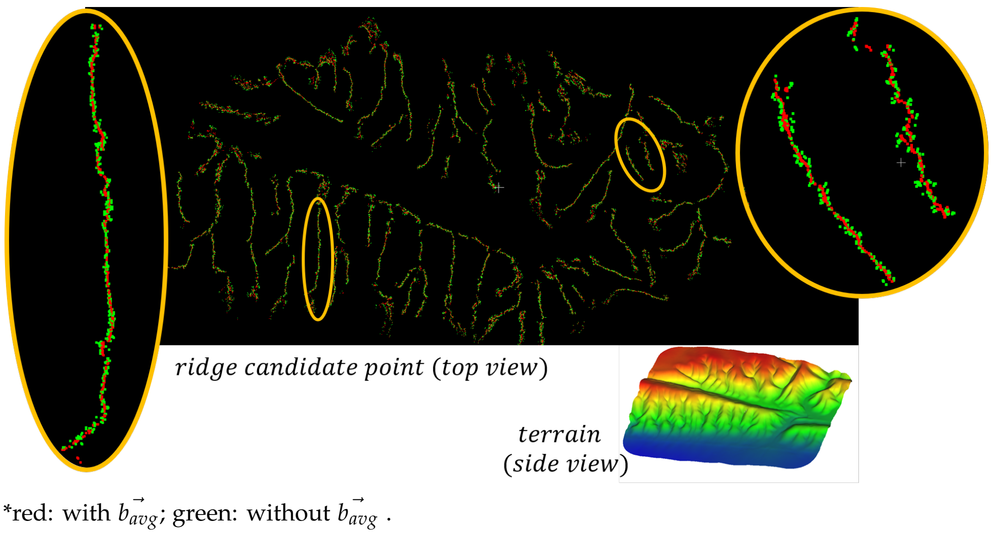

ridge candidate point (top view)
terrain (side view)
*red: with $\vec{b_{avg}}$; green: without $\vec{b_{avg}}$.

Figure 3.21: Result of using and not using weighted average bisector $\vec{b_{avg}}$.

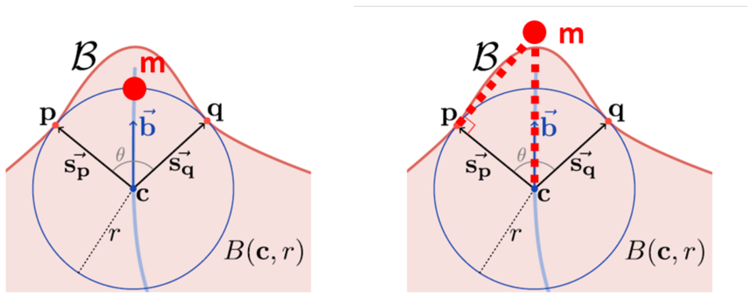
Figure 3.22):

1. The intersection of the bisector and the medial circle (Figure 3.22a).
   If the coordinate of c is (x_c, y_c, z_c), the radius is r, and the normalized weighted average bisector is $\vec{b_n}$,
   Candidate point: $m = c + r \cdot \vec{b_n}$

2. The intersection of the bisector and right-angle side l_{pm} (Figure 3.22b):
   In triangle Δmpc, $\angle mpc = 90°$,
   the separation angle $\angle pcq = \theta$, $l_{pc} = r$, $\angle pcm = \frac{1}{2}\angle pcq = \frac{\theta}{2}$
   $l_{mc} = \frac{l_{pc}}{cos(\angle pcm)} = \frac{r}{cos(\frac{\theta}{2})}$
   Candidate point: $m = c + l_{mc} \cdot \vec{b_n}$

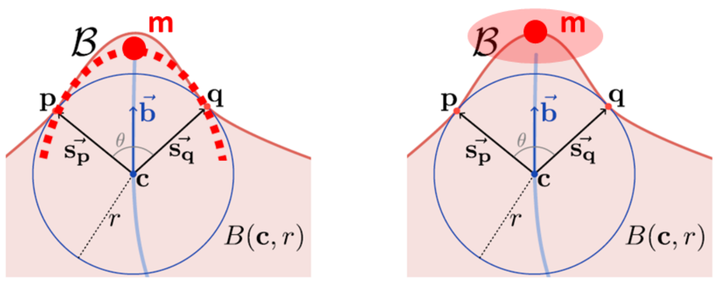
3. Using splines fitting the profile/section of plane pcq, the intersection of bisector and the spline will be the candidate point (see Figure 3.22c).

4. Find the k-nearest-neighbour points in the ground point cloud, and use spatial interpolation to get the z coordinate of candidate point m (Figure 3.22d). Candidate point $m = (x, y, z)$, the coordinate of c is (x_c, y_c, z_c), and the normalized weighted average bisector $\vec{b_n} = (\vec{b_{nx}}, \vec{b_{ny}}, \vec{b_{nz}})$;
   $x = c + r \cdot \vec{b_{nx}}$
   $y = c + r \cdot \vec{b_{ny}}$
   $z =$ NN spatial Interpolation at (x, y)



(a) Intersection of bisector and the circle

(b) Intersection of bisector and right-angle side



(c) Splines fitting

(d) Spatial interpolation

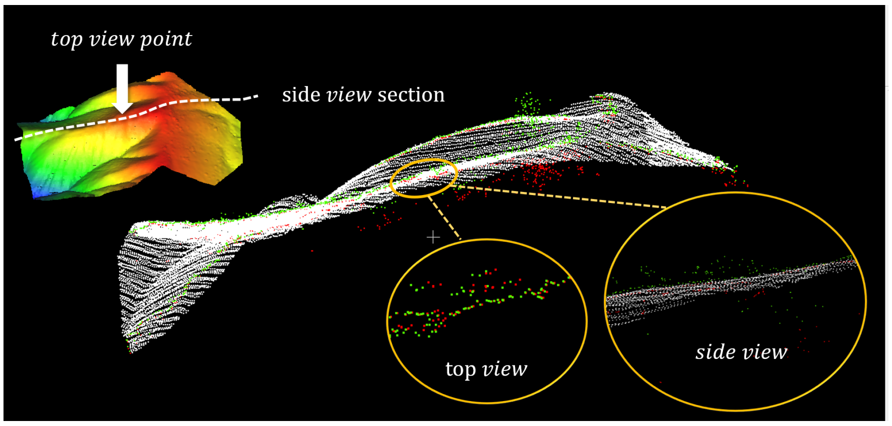
Figure 3.22: Four methods to calculate candidate points coordinates (candidate point: m).

The resulting candidate points of first two methods is in Figure 3.23. Both Figure 3.22 and Figure 3.23 shows that in side view, the intersection of the bisector and the circle is a little bit below the surface, while the intersection of bisector and right-angle side is higher than the surface. As to the top view, those two results are almost overlapping, which means the (x, y) coordinate of candidate points is stable in the two methods.

The difference of the (x, y) coordinate between the first and the second method is small, but the z coordinate is not accurate. This project takes the (x, y) coordinate with the first method (intersection of bisector and the circle) first. Then implement the Nearest Neighbour interpolation [1] at (x, y) to get z coordinate. In this way the candidate point's coordinate $(x, y, z)$is found.

This project also tried spline fitting to calculate candidate points coordinate (see Figure 3.22c). But to use spline fitting, additional points are required to control the curve. Those points can not be found, so spline is not the solution for candidate point coordinate calculation.

1 https://en.wikipedia.org/wiki/Nearest-neighbor_interpolation



red: intersection of bisector and the circle; green: intersection of bisector and right-angle side; White: terrain point cloud.

Figure 3.23: candidate points with first two methods.

3.5 POLYLINE GENERATION FROM CANDIDATE POINTS USING THE GRAPH THEORY

In this section, the main task is generating polylines from the candidate points as the breakline using graph theory.
This section first connects the candidate points to its closest neighbours with the minimum spanning tree (MST) algorithm. Then it simplifies the MST into a simple polyline with the shortest path algorithm.
As defined in Chapter 1, a breakline of an object's surface is a polyline. A polyline is a connected series of line segments, more exactly, it is a curve specified by a set of ordered points $P(p_1, p_2, \ldots, p_n)$ called vertices. The start and the end points of a polyline are called free vertices.

3.5.1 Connecting candidate points with the minimum spanning tree

As Section 3.1 explained, the candidate points from the same sheet should be the vertices from one polyline (or breakline). Section 3.4 shows that the extracted candidate points distribute like a polyline. To get this polyline, the points need to be connected to the nearest neighbours first.
The connection can be formed by the following process:
In a set of candidate points P from the same medial sheet:

1. Initialize a set of points $G = \varnothing$ and a set of line segments $l = \varnothing$;
2. starting at a random candidate point p_0, $P \setminus p_0$ (remove p_0 from P), $G \cup p_0$ (add p_0 to G);
3. For every point g_i in G, find the closest point p_i in P and calculate the distance d_i for each pair of (d_i, p_i), and find the minimum distance d_{min} and its corresponding points $\hat{d}$ and $\hat{p}$, $P \setminus \hat{p}$, $G \cup \hat{p}$, $l \cup (\hat{g}, \hat{p})$;
4. Repeat the third step until $P = \varnothing$; which means every point has been added into G, and the connection described by line segments is saved in l.

This is a description of the Minimum spanning tree algorithm.
In an undirected graph $G = (V, E)$, V is a set of vertices, (v_i, v_j) represents the edge connecting the vertices v_i and v_j, and E is a set containing all the edges, in other words, $(v_i, v_j) \in E$, and $w(v_i, v_j)$ is the weight of edge (v_i, v_j).

The Minimum spanning tree T is a graph, containing all the vertices and a subset of edges in G ($T = (V, \hat{E})$), and has the minimum sum of the edge weights $\sum_{(v_i,v_j)\in\hat{E}} w(v_i, v_j)$.
The graph G can be initialised by adding every candidate point as a vertices, adding each pair of candidate points as an edge, and the distance as the weight of the edge. So for a set of n points, the created graph G contains n vertices and $n(n-1)/2$ edges. The out put T is the minimum distance spanning tree. The concept is shown in Figure 3.24. The result of this step is in Figure 3.25.

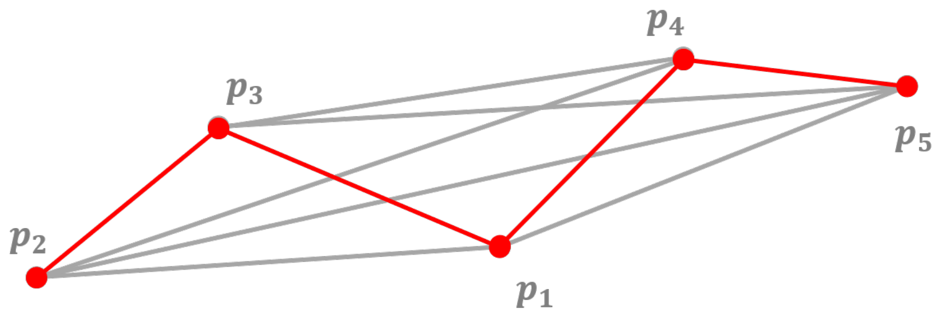


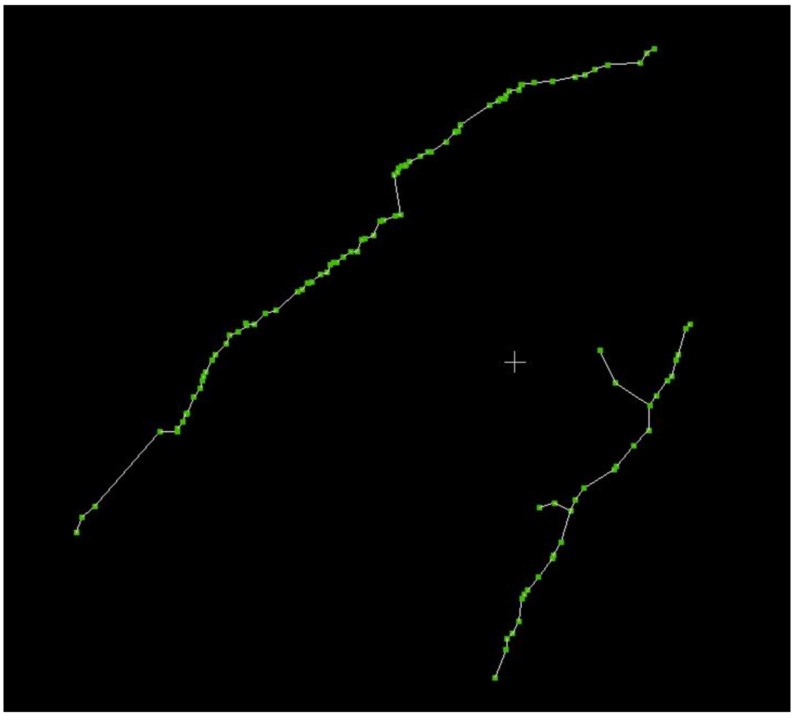
Figure 3.24: Connect candidate points by MST concept

3.5.2 Simplification of the MST into one polyline with the shortest path algorithm

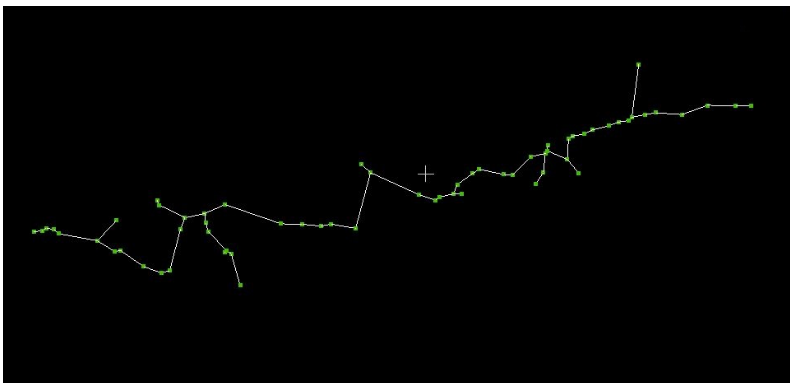
This section describes how to get a polyline from the minimum spanning tree graph (T), which stores a set of unordered line segments, by the shortest path algorithm. The target polyline is the longest path in T (see Figure 3.26). Notice that, not all the candidate points are taken into the output polyline. In Figure 3.26, only the red points are taken into the output polyline, and the black points are eliminated from the output.
There is no cycle in T, otherwise a vertex in the cycle can have two paths connecting to another vertex in the cycle, which is in violation of the minimum sum distance requirement. Therefore, there is only one path from one vertex to another vertex in MST.
This project tried three methods to find the start and end vertexes:

1. **largest Euclidean distance**: It calculates the Euclidean distance for each pair of candidate points. The point pair with the largest Euclidean distance are the start and end vertices in Graph T. The output polyline is the the path between them, and it can be found with the shortest path algorithm.
2. **longest accumulation distance**: It finds the path for each pair of vertices in Graph T. The path with the largest accumulation weight, in other words the path with the longest accumulation distance, is the required polyline.
3. **maximum number of vertices**: It finds the path for each pairs of vertices in Graph T. The path with the maximum number of vertices, in other words the path consisting of the most candidate points, is the required polyline.

In a set of well extracted candidate points, for example Figure 3.26, those three methods have similar output. Figure 3.27 shows the output polylines for the three methods, and the difference appear around the free vertices of a polyline. For this dataset, the ”largest Euclidean distance” and the ”maximum number of vertices” provide better results. But it might be different for other inputs. Because the Euclidean distance for each pair of vertices has already been calculated in previous step (Section 3.5.1), to save the computation time, the default option is the ”largest Euclidean distance”.



(a) .



(b) .

Figure 3.25: Minimum spanning tree result (green: candidate points, white: MST edge)

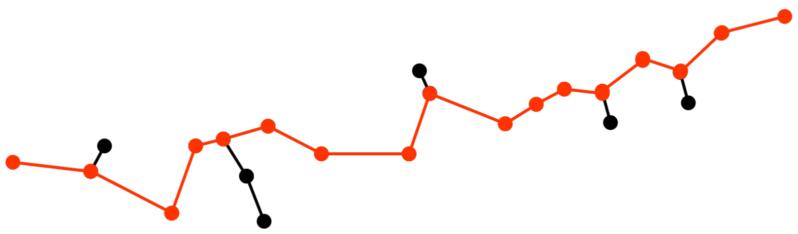
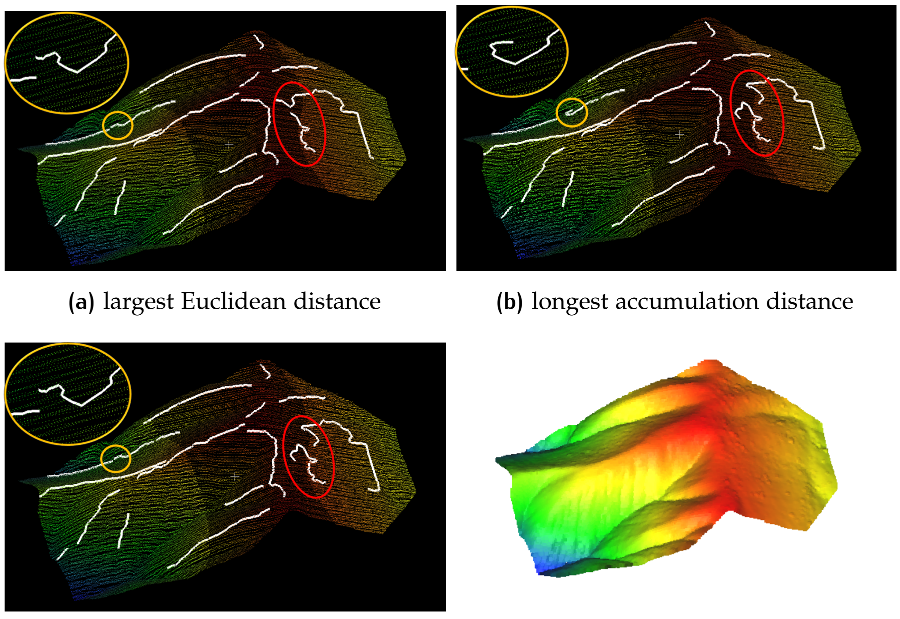


Figure 3.26: From MST line segments to polyline (point: candidate point; line: MST; red line: extracted polyline)



(a) largest Euclidean distance

(b) longest accumulation distance

(c) maximum number of vertices

(d) Point cloud (colour stands for height)

Figure 3.27: The result of Simplify MST into one polyline with three methods.

3.6 POLYLINE GENERATION FROM CANDIDATE POINTS USING A POLYNOMIAL FIT

This section provide another option to obtain breaklines from candidate points. It fits a polynomial function for all candidate points of a sheet, and exports a smooth polyline as breakline.
Polynomial fitting for breakline generation is actually a 2D problem. Projecting the polyline onto the xy plane, and fitting a polynomial function in the xy plane. The z coordinate could be calculated by spatial interpolation at (x, y) in the point cloud to make sure the breakline is always on the terrain surface.
This project uses the cubic polynomial function:

$$y = f(x) = t_3 \cdot x^3 + t_2 \cdot x^2 + t_1 \cdot x + t_0 \tag{3.7}$$

The "polynomial fit" treats the breakline as a function, therefore for one x value, the function $f(x)$ maps to exactly one y value, in other words, one x can not correspond to more than one y.
The candidate points in Figure 3.28 are not x-monotone, therefore it can not directly use its (x, y) coordinate for polynomial fit. Hence, a coordinate transformation is required.

The transformed coordinate x axis can be largest eigen value's corresponding eigen vector. The y axis is another eigen value's corresponding eigen vector. Than the transformed coordinate is settled. But is does not work for all breaklines (for example the highlight breaklines in Figure 3.27).
There are two common fitting methods: the least square fit and the maximum likelihood fitting. The sum of the distance between each candidate point and the fitting polyline is the root mean square error (RMSE) (see Figure 3.29). This project takes the least square fit because calculating the RMSE is easier during the process.

For ill distributed candidate points (see Figure 3.30), the RMSE is large. It shows the candidate points are inadequate to generate a breakline. Therefore, it will be removed from the resulting breaklines.

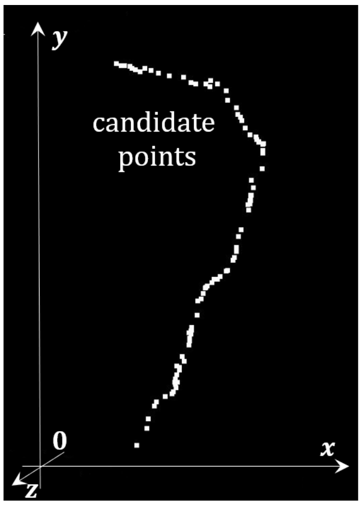


Figure 3.28: Candidate points for polynomial fitting

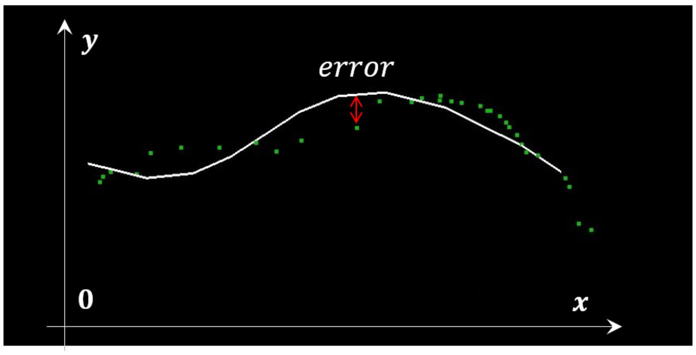


Figure 3.29: The root mean square error

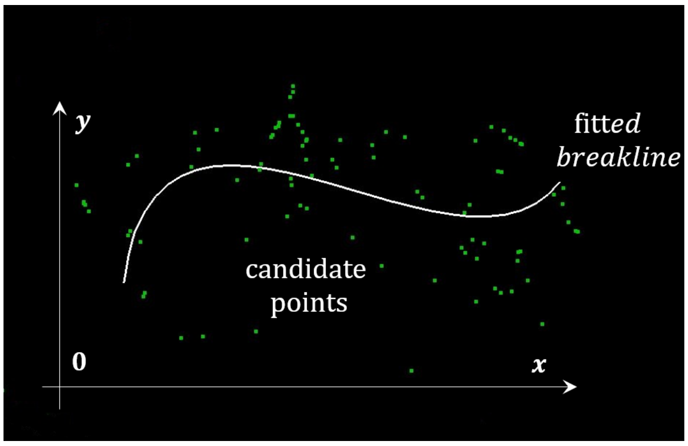


Figure 3.30: The large root mean square error

3.7 THE TOPOLOGY OF BREAKLINES

Based on the MAT adjacency graph, this section finds the topology relation between different polylines.
Two polylines can only have four kinds of relationships: isolated, touching, crossing or overlapping (see Figure 3.31). Within the scope of extracting breaklines for mountains, the cross relation will turn into touching, and overlap will be treated as one breakline (see Figure 3.32). Therefore two ridges or two valleys can not cross or overlap each other, they only have two relations: isolated or touching.
The breakline adjacency relationship is the same in the corresponding medial sheet.

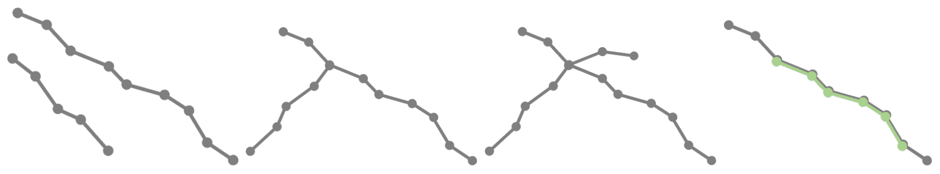
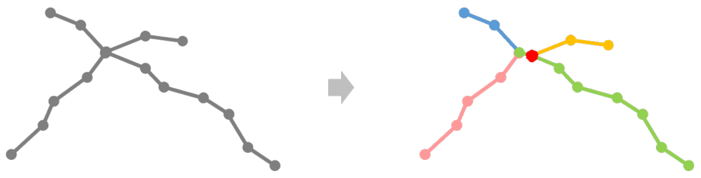

(a) Isolated (b) Touching (c) Crossing (d) Overlapping

Figure 3.31: Four kinds of relationships between two polylines


(a) cross turns into touching in breakline

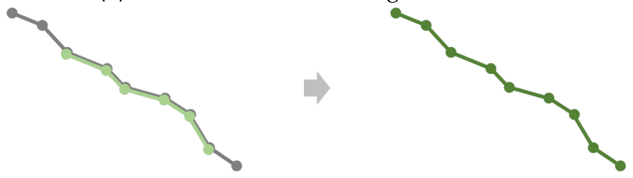
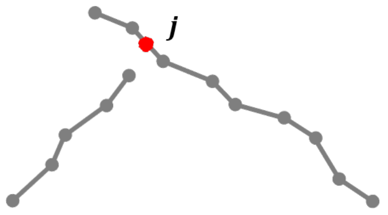
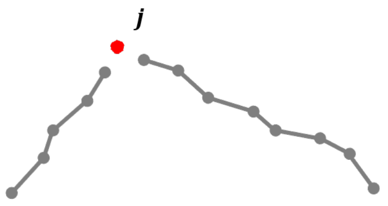

(b) overlap turns into one breakline

Figure 3.32: There is no cross or overlap between two breakline

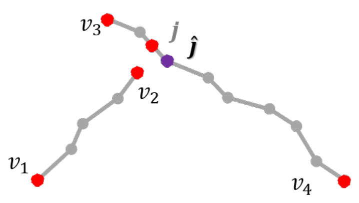
Because a breakline is the geometry for the object's surface, only the surface adjacency (see Section 3.3.3) is considered. The vertices where two breaklines touch or cross are called junction vertices. As Section 2.2 explained, they need to be estimated.
For two isolated breaklines, they are generated separately from different sheets and graphs, hence there is no edge or path connecting them. Isolated lines do not need further processing.
As for two touching breaklines, the junction vertices need to be located. There can be two situations (Figure 3.33a and Figure 3.33b), they both touch where two polylines are nearest.
For the four free vertices v_1, v_2, v_3 and v_4, the closest point in another polyline can be found vc_1, vc_2, vc_3 and vc_4. The nearest vertices pair $(v_i, vc_i), i \in 1, 2, 3, 4$ should be added to the polyline, and vc_i is the junction vertex. But vc_i is only an estimated point, and it may be a little shifted from the true junction point j (see Figure 3.33c and Figure 3.33d).



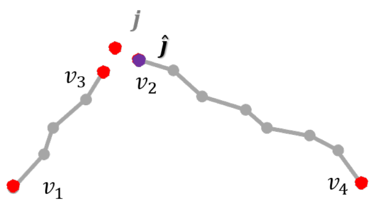
(a) j in the polyline



(b) j outside the polyline



(c) $\hat{j}$ for j in the polyline



(d) $\hat{j}$ for j outside the polyline

j: true junction point; v_1, v_2, v_3, v_4: free vertices from two polyline; vc_1, vc_2, vc_3, vc_4 closest vertices of free vertices; $\hat{j}$: estimated junction point .

Figure 3.33: Three methods to calculate candidate points coordinates.

3.8 POLYLINE SIMPLIFICATION AND SMOOTHING

Some parts of the resulting breaklines are zigzagging (especially when one zooms in, such as in Figure 3.34) and contain many points. Some applications might require a smooth polyline or with few points. This section provides a method to simplify the polyline and compares several options to smooth the polyline. Depending on the user's requirement, the breaklines can be simplified or smoothed, or both. Or a polyline can be simplified first, than smoothed, and simplified again.

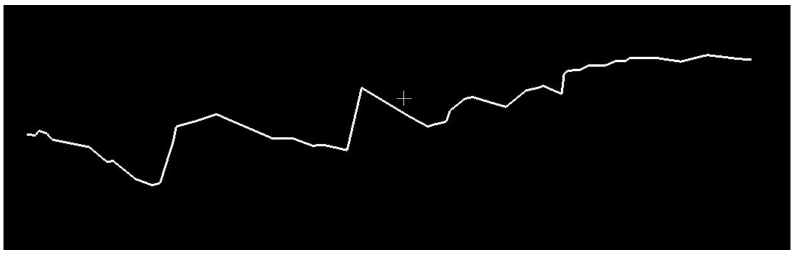
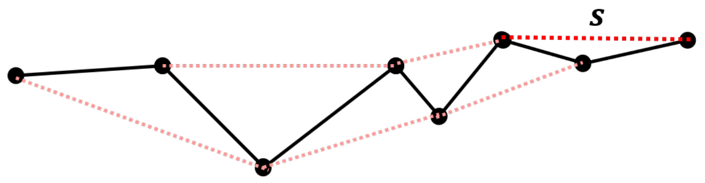


Figure 3.34: zigzagging

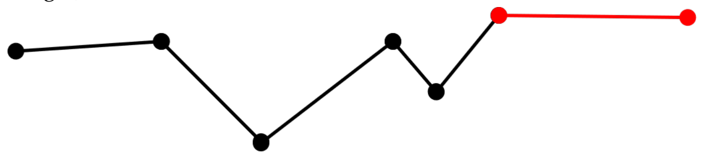
3.8.1 Polyline simplification

The purpose of line simplification is using as little as possible points to represent the polyline's shape.
This project use Visvalingam's algorithm to simplify the 3D polyline[2]. Compered with Douglas–Peucker, the well known line simplify algorithm, Visvalingam algorithm may be more effective and has a remarkably intuitive explanation: it progressively removes points with the least-perceptible change. See Figure 3.35, each pair of adjacent line segments forms a triangle, and the area of every triangle can be calculated. It removes the point with the smallest triangle area, then updates the polyline and the triangle area until all the triangle areas are larger than a threshold. The result is shown in figure 3.36.

2 https://hull-repository.worktribe.com/output/459275



(a) Calculating the area of every triangle (s is the smallest triangle).



(b) Remove point with the smallest triangle.

Figure 3.35: 3D polyline simplify.

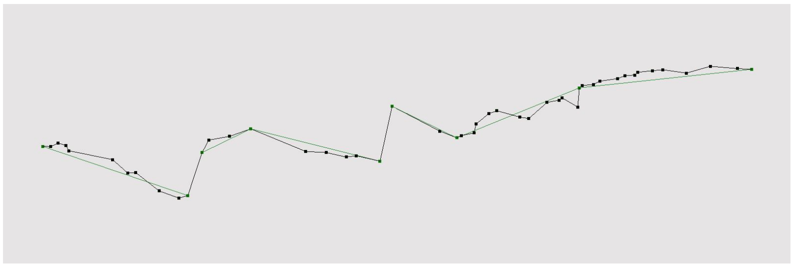


Figure 3.36: 3D polyline simplification result (before black; after green)

3.8.2 Polyline smoothing

The purpose of line smoothing is reducing sharp angles by adding new points. Here lists the comparison of three polyline smoothing method for this project:

1. b-spline
   B-spline smoothing only passes the start point and end point of a polyline [3], it might not pass other point in between.

2. cubic spline
   The cubic spline method passes all the points in the polyline. It fits a cubic polynomial function for each pair of neighbour points in the polyline. Forcing every piecewise function to have the same gradient at the breakline points except at the start and the end points in the polyline.

3. polynomial fit
   Similar to Section 3.6, the polynomial fit might not pass all the points from the polyline. It fits a polynomial function for all the points in the polyline.

This project tests the b-spline (see Figure 3.37). The output polyline is rather smooth and contains much more points, but it has strange curve at sharp corners.
This project also tests the polynomial fit. The difference of polyline smoothing and polyline generation with a polynomial fit is in the input points. Polyline smoothing takes the vertices in the polyline as input, while polyline generation takes the candidate points as input. The vertices in the polyline are a subset of candidate points. Figure 3.38 shows that with "polynomial fit", the smoothed breakline might very different from the input polyline.
This project also proposes another method for line smoothing. The main idea is

3 there are three cases for b-spline: open, clamped and close. This project takes "clamped" to force the smoothed polyline pass the start and end points.

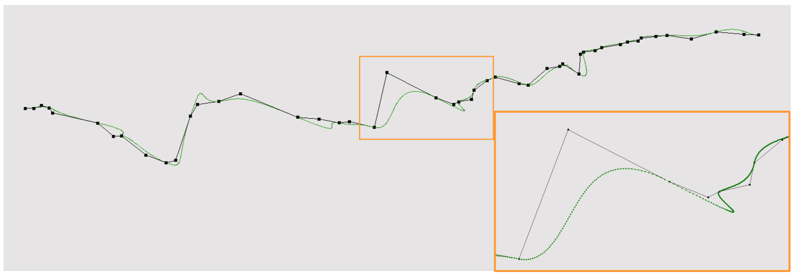


Figure 3.37: 3D polyline smoothing: b spline (before black; after green)

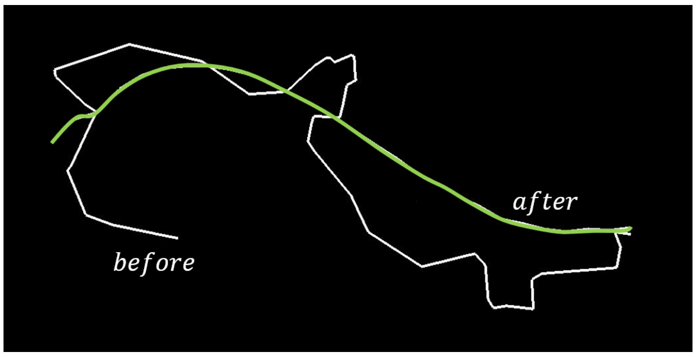
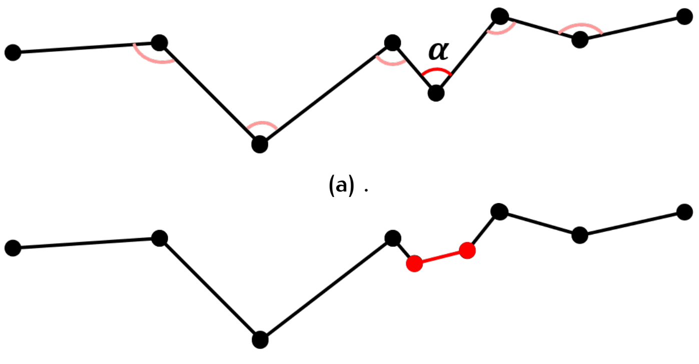


Figure 3.38: polynomial fit for polyline smooth

removing sharp angles. It calculate the angle of each pair of adjacent line segment in a polyline (see Figure 3.39). For the point with the sharpest angle α, two new point are inserted in the middle of the two adjacent line segments separately, and after that the old point is removed. This process is repeated until all the angles are larger than an angle threshold. The full procedure is given in Algorithm 3.3, in each iteration, one point is modified and one point is inserted to remove the sharp angle.



(a) .

(b) .

Figure 3.39: 3D polyline smooth.

The result is given in Figure 3.40. Because the derivative at point in the polyline is not continuous, the resulting polyline is not actually smooth. Within this method, the sharp angle is removed, the shape of polyline is kept and it does not insert too much points like b-spline fitting.

Each polyline smoothing method has advantages and disadvantages. A combination of them might generate a better result, for example first smoothing a line by removing sharp angles, then b-spline smoothing, finally line simplification.

Notice that the polyline simplification and smoothing is not always necessary. The candidate point extraction process performs a kind of spatial interpolation in the xy plane. A well extracted candidate point is already distributed smoothly. Ill

Algorithm 3.3: lineSmooth (*pts*, *α*)

```
Input: pts: a sequence of points of the input polyline,
α: the threshold for sharp angle
Output: pts: a sequence of points of the smoothed polyline
1  $angList \leftarrow$ the angle of two adjacent line segments in the polyline;
2  $minAng \leftarrow$ the minimum angle in $angList$;
3  while $minAng < \alpha$ do
4      $Ang\_idx \leftarrow$ the index of minAng in $angList$;
5      $pt\_idx = Ang\_idx + 1$;
6      $pt \leftarrow pts[pt\_idx]$;
7      $pt\_pre \leftarrow pts[pt\_idx - 1]$;
8      $pt\_aft \leftarrow pts[pt\_idx + 1]$;
9      $pt_1 \leftarrow (pt\_pre + pt)/2$;
10     $pt_2 \leftarrow (pt + pt\_aft)/2$;
11     $pts[pt\_idx] = pt_1$;
12     $pts.insert(insertIdx = pt\_idx + 1, insertPt = pt_2)$;
13     $angList[Ang\_idx] \leftarrow$ the angle between line segment $(pt_1, pt\_pre)$ and
        $(pt_1, pt_2)$;
14     $ang \leftarrow$ the angle between line segment $(pt_1, pt_2)$ and $(pt_2, pt\_aft)$;
15     $angList.insert(insertIdx = Ang\_idx + 1, insertAng = ang)$;
16     $minAng \leftarrow$ the minimum angle in $angList$;
17 return pts
```

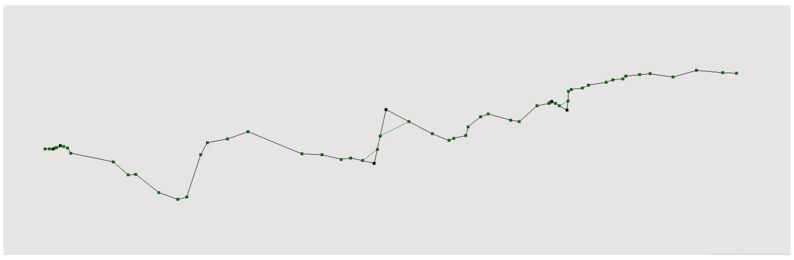


Figure 3.40: 3D polyline smoothing result (before black; after green)

conditioned candidate points are caused by ground noise or incomplete sheet segmentation results mostly. A better segmentation method would thus also help.

4 IMPLEMENTATION AND EXPERIMENTS

This chapter mainly explains the implementation details and some experiments to generate breaklines. Section 4.1 introduces the tools and open source libraries used in the project. Section 4.2 shows the software architecture and presents a summary of all the involved user defined parameters in the breakline generation process. Section 4.3 describes the experimental data used for this project. Finally Section 4.4 to Section 4.7 prsent experiments of some important intermediate results for several steps that are part of the breakline generation process.

4.1 TOOLS AND LIBRARIES

This project uses some existing tools and open source libraries to accomplish the task. For the development, the object-oriented programming language C++ is used. The most important library is geoflow [1]. All the steps to extract breaklines are implemented with this library. The main functions that are integrated in geoflow for this project are:

- normal vector estimation of a point cloud;
- the ball shrinking algorithm to extract medial atoms;
- Principal Component Analysis (PCA) for calculating eigenvalue and eigenvector of a point cloud;
- Visvalingam polyline simplification algorithm;
- 3D geometry viewer;
- some other useful 3D geometry operators, such as vector operators.

Furthermore, some other libraries used for this project are:

- boost: Minimum Spanning Tree [2] and shortest path [3];
- eigen: polynomials module [4];
- spline: b-spline interpolation [5];
- CGAL: spatial geometry operators [6].

In addition, Cloud Compare is used for manually drawing breaklines as reference data for validation. Qgis is also used for some spatial analyses in validation step. AcrMap is used for comparing result.
In the end, the project is on github [7].

1 https://github.com/tudelft3d/geoflow
2 https://www.boost.org/doc/libs/1_51_0/libs/graph/doc/kruskal_min_spanning_tree.html
3 https://www.boost.org/doc/libs/1_34_0/libs/graph/doc/dijkstra_shortest_paths.html
4 https://eigen.tuxfamily.org/dox/unsupported/group__Polynomials__Module.html
5 https://github.com/chen0040/cpp-spline.git
6 https://www.cgal.org/
7 Sourcecode:https://github.com/qq2012/geoflow-nodes

4.2 THE IMPLEMENTED PROTOTYPE

This section shows the software architecture and presents a summary of all the involved user defined parameters in the process.

4.2.1 The software architecture

The architecture of the code is based on the workflow explained in Section 3.1. The working flow chart implemented in geoflow is in Figure 4.1.

The parameter "isInterior" of "MaGeometryNode" in MAT processing step (see

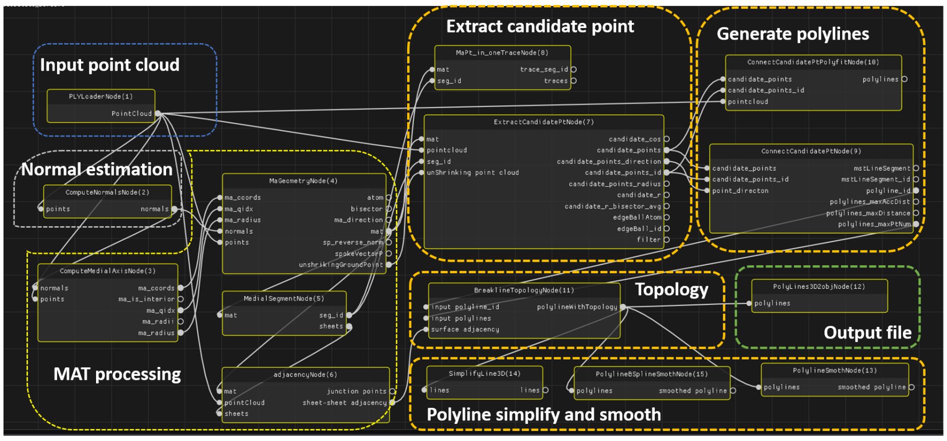


Figure 4.1: Working flow chart in geoflow_node.

Table 4.1 and Figure 4.3) controls if the output breaklines are ridges or valleys. If $isInterior = True$, the rest of flow chart works on the interior MAT, and generates ridges. While if $isInterior = False$, it works on the exterior MAT, and generates valleys.

4.2.2 Summary of thresholds

During the generation of breaklines, a number of user defined parameters are used, and they are listed in Table 4.1.

Some of the parameters such as $isInterior$ and $OnlySurfaceAdjacent$ determine the output of different steps.

The others parameters are user defined thresholds, and they are different from each input point cloud dataset. All of them depend on the point cloud resolution. Some of the thresholds like r prefer a larger value to get a more promising result, but will increase the execution time (explained in Section 3.4.1). c_n depends on the point cloud resolution, the degree of ground noise and the minimum length of breaklines the user want to detect. But the thresholds involved in the segmentation process don't have any default value, they need the user to find out the best values.

4.3 DATA AND RESEARCH AREA

In this project, the Lidar point clouds are download from OpenTopography [8]. Within the scope of this project, a mountain terrain with no trees is chosen as test data. The test data details are in Table 4.2 and Figure 4.2. As one can see, the second dataset

8 https://opentopography.org/

Variable and symbol	Type	Step	Description
$isInterior$	bool	MAT Processing	control the output breakline is ridge or valley
minimum point number: c_n	integer	MAT Processing: segmentation	the minimum number of medial atoms to firm a medial sheet
segmentation method	enumerate	MAT Processing: segmentation	choose the segmentation method*
ball overlap threshold: bo_t	scalar	MAT Processing: segmentation	evaluate if two medial atoms belong to the same medial sheet
bisector difference threshold: $\vec{b_t}$	degree	MAT Processing: segmentation	evaluate if two medial atoms belong to the same medial sheet
medial direction difference threshold: $\vec{n_{ct}}$	degree	MAT Processing: segmentation	evaluate if two medial atoms belong to the same medial sheet
$OnlySurfaceAdjacent$	bool	MAT processing: adjacent	control if the adjacency graph only exports the surface adjacency
search radius: r_1	meter	MAT processing: adjacent	find the neighbours of a medial atom within the radius
The minimum number of neighbour points: ac_n	integer	MAT processing: adjacent	determine if two medial sheets are adjacent.
The maximum distance of neighbour points to the terrain: dp_a	meter	MAT processing: adjacent	determine if two adjacent medial sheets are surface adjacent.
deviation angle threshold: α_t	degree	candidate point: edge ball detection	determine if a medial atom is an edge point
search radius: r_2	meter	candidate point: edge ball detection	find the neighbours of a medial atom within the radius
Maximum edge ball radius: r_x	meter	candidate point: filtering	remove candidate points with small curvature(big radius)
Minimum edge ball radius: r_n	meter	candidate point: filtering	remove edge balls resulting from ground noise
Minimum distance to unshrunken point: du_n	meter	candidate point: filtering	remove candidate points in planar areas
Minimum distance to point cloud: dp_n	meter	candidate point: filtering	remove edge ball atoms far away from terrain
bisector average factor: k	integer	candidate point:coordinate calculation	the number of knn neighbours used for calculatint the average bisector
Maximum Root Mean Square Error threshold: $RMSE_t$	scalar	Polynomial fitting	remove candidate points not distributed like a line

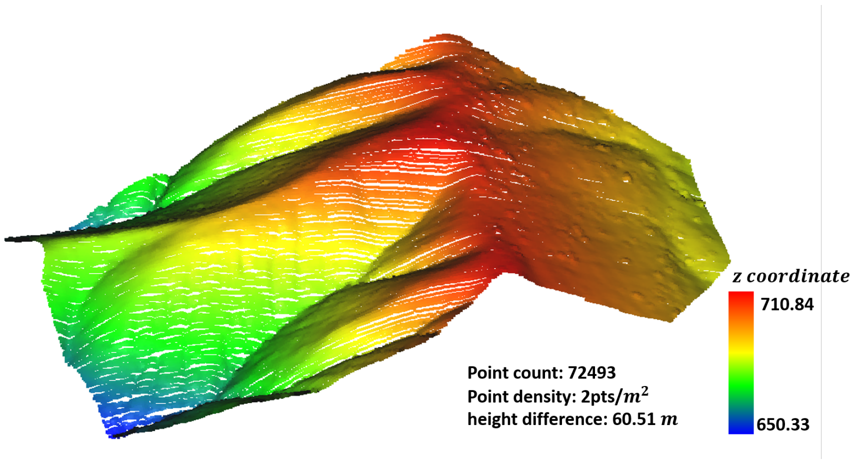
*segmentation method enumerate type = {"bisector", "ball overlap", "medial direction", combine "bisector" and "ball overlap", combine "bisector" and "medial direction", combine "ball overlap" and "medial direction", combine three} (see Section 3.3.2)

Table 4.1: Control parameters and useful thresholds

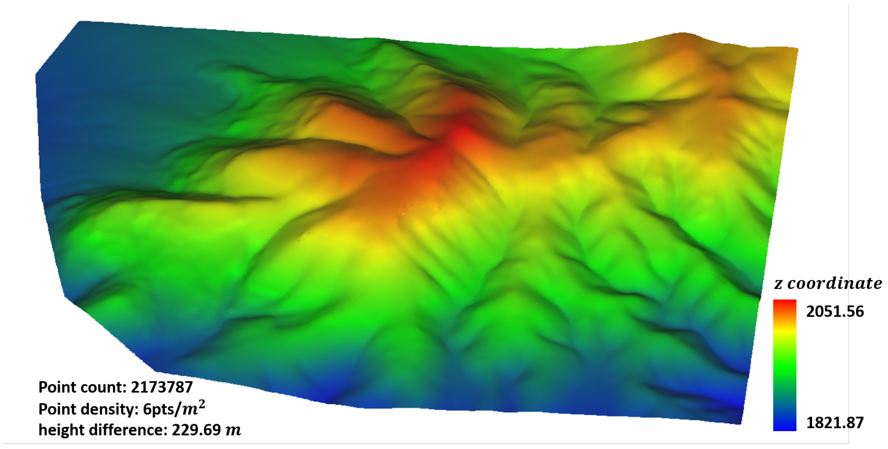
contains a large area and the point density, and the terrain height difference is bigger too.

	Format	Point count	Point density	Source
1	LAS	72493	$2 pts/m^2$	OpenTopography
2	LAS	2173787	$6.03 pts/m^2$	OpenTopography

Table 4.2: Overview of test data and their details



(a) Data 1.



(b) Data 2.

Figure 4.2: The two test datasets.

4.4 MEDIAL SEGMENTATION

Medial segmentation is a part of the MAT processing step (see Figure 4.3). The user can choose the segmentation method and the related thresholds (see Table 4.1).

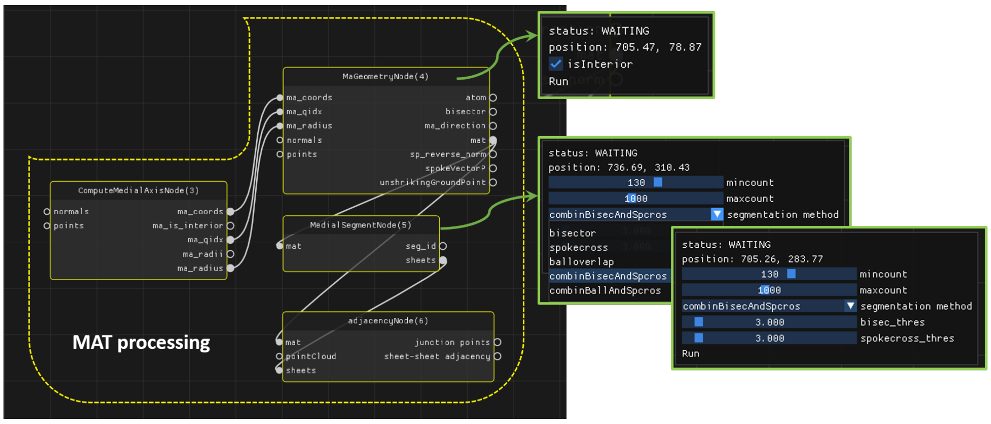
Since one medial sheet will generate one breakline (explained in Section 3.1), the medial sheets establish the basic shape of the generated breaklines. The medial segmentation is crucial to the result breakline.

The "minimum point number" c_n (see Table 4.1 and Section 3.3.2) is related to the smallest breakline that is generated, which is a constant value for one dataset.

The number of points in a medial sheet is influenced by the depth and length of the sheet (see Figure 4.4a), the density of input point cloud (explained in Section 2.2.3).

Data	minimum z coordinate	maximum z coordinate	height difference [meter]
1	650.33	710.84	60.51
2	1821.87	2051.56	229.69

Table 4.3: Height difference of two test data



```
status: WAITING
position: 705.47, 78.87
✓ isInterior
```

```
status: WAITING
position: 736.69, 310.43
130                     mincount
1000                    maxcount
combinBisecAndSpcros ▼  segmentation method
bisector
spokecross
balloverlap
combinBisecAndSpcros
combinBallAndSpcros
```

```
status: WAITING
position: 705.26, 283.77
130                     mincount
1000                    maxcount
combinBisecAndSpcros ▼  segmentation method
3.000                   bisec_thres
3.000                   spokecross_thres
```

Figure 4.3: Medial segmentation and its thresholds.

If a medial sheet contains too little atoms, the medial sheet corresponds to a small breakline, and it will be eliminated by c_n (see Figure 4.4b).

Generated breaklines with different c_n are shown in Figure 4.5. The figure shows that a too small c_n will take small breaklines into account, but a too large c_n will remove correct breaklines.

In order to investigate whether the small breaklines are expected or not, Figure 4.6 gives more information. The image shows an interesting case: a long breakline that is approximately in parallel with two other short breaklines. Checking the medial sheets that generate those three breaklines, it looks like a "tree". The two small sheets are the branches of the big main sheet. The bottom up view shows that the two small sheets indeed indicate the curvature change of the terrain surface. Those two small breaklines are expected output.

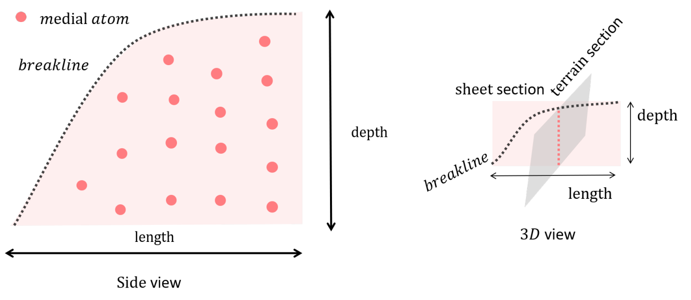
The "segmentation method" and its related thresholds play a decisive role in the segmentation result. There are two types of error in the segmentation:

1. One medial sheet is wrongly segmented into two or more sheets (see Figure 4.7). Two breaklines are extracted which should be one line actually.

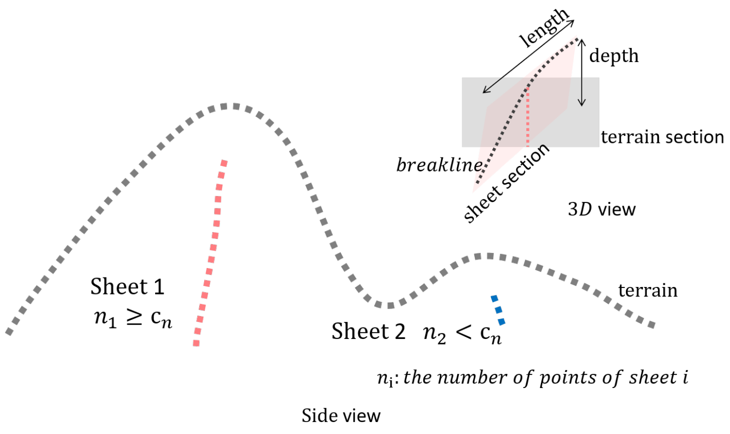
2. Two or more medial sheets are wrongly segmented as one sheet (see Figure 4.8).
   In Figure 4.8a, two adjacent sheets are merged into one. Polylines generated with the graph theory method (explained in Section 3.5,) will generate one polyline from a part of the sheet, and will cause a missing breakline of another part. Polylines generated with polynomial fitting (explained in Section 3.6) will result in one line which does not suit neither part of the medial sheet.
   In Figure 4.8b, two isolated but close sheets are merged as one. Depending on the extracted candidate points distribution, the extracted polyline is unexpected.

In practice, big segmentation errors can be avoided with careful "segmentation method" choice and thresholds setting. The error appears mostly at small breaklines (see Figure 4.9).

The shape of two adjacent medial sheets will also effect the output breakline (see

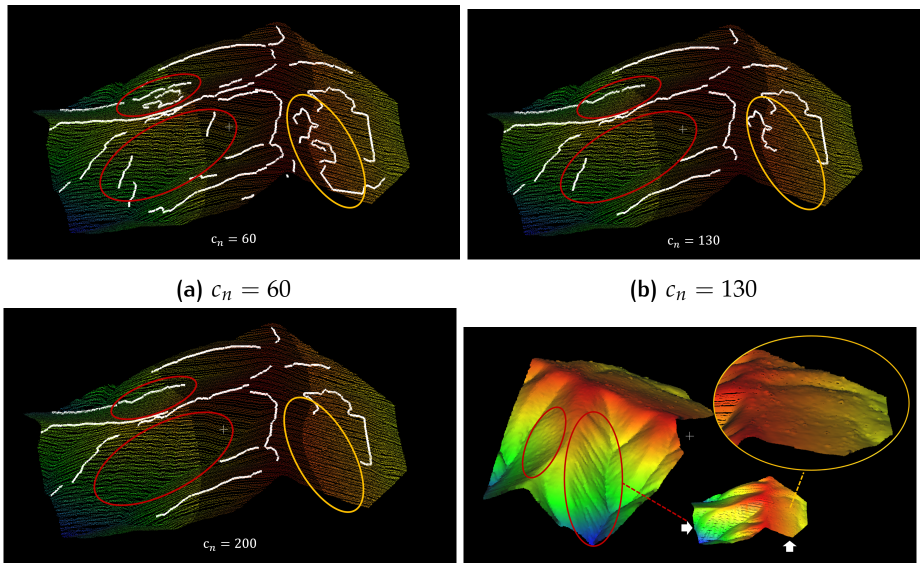


(a) The number of points in a medial sheet is influenced by the depth and length of the sheet.



(b) Very small features in the ground are eliminated by c_n.

Figure 4.4: Medial sheet and c_n.

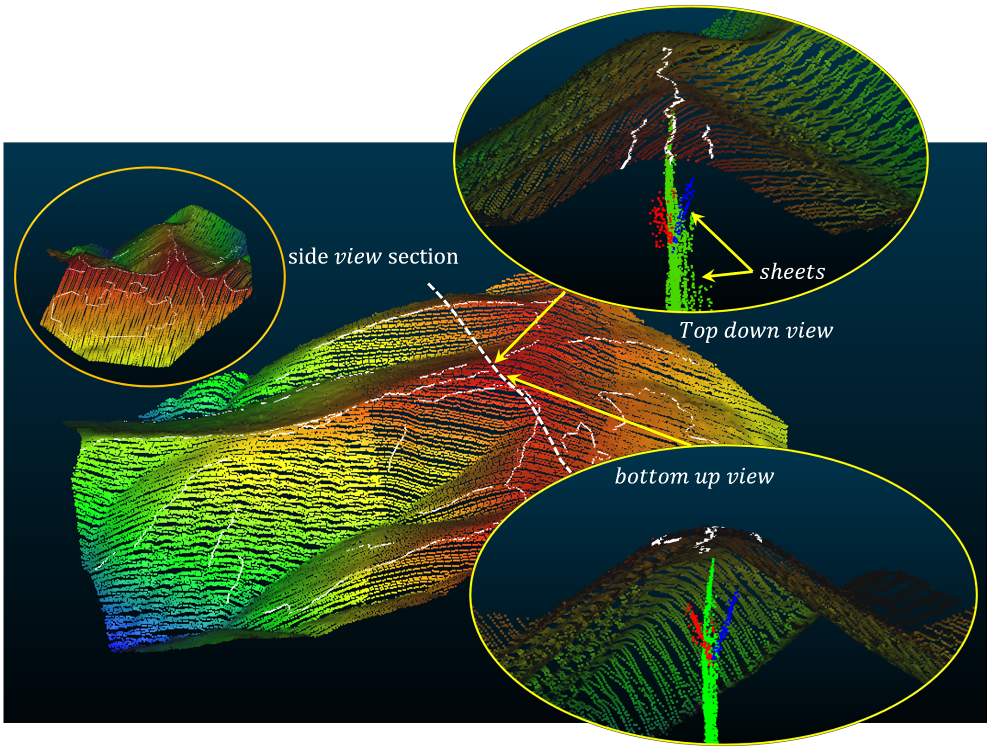


(a) $c_n = 60$ (b) $c_n = 130$

(c) $c_n = 200$ (d) Terrain point cloud

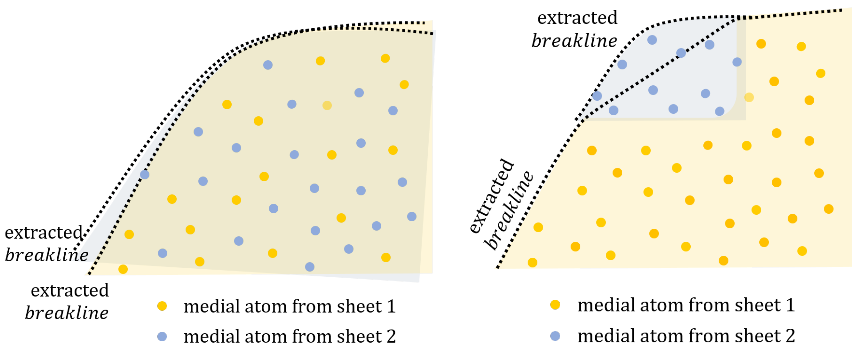
The polyline is generated with graph theory without topology, other parameters: "segmentation method" = "bisector", $\vec{b}_t = 3$, $\alpha_t = 15$, $r_2 = 45$)

Figure 4.5: different c_n with generated ridge for data 1.



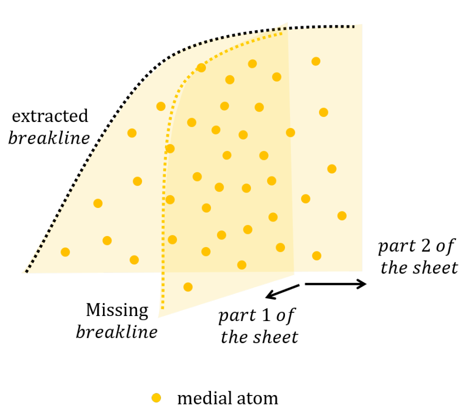
The ridge is generated with graph theory without topology, other parameters: "segmentation method" = "bisector", $\vec{b}_t = 3$, $c_n = 80$, $\alpha_t = 15$, $r_2 = 45$)

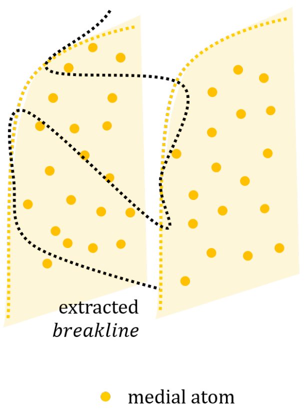
Figure 4.6: Three parallel breaklines.



(a) The whole sheet is separated as two. **(b)** A part of sheet is separated

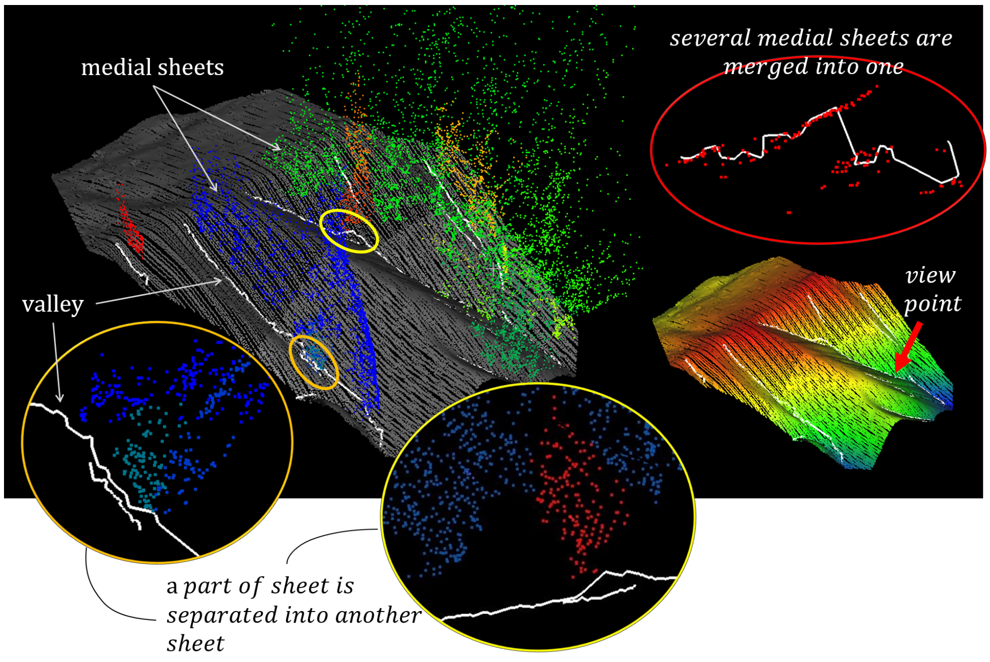
Figure 4.7: One medial sheet is wrongly segmented into two sheets (side view).





(a) Two adjacent sheets are merged into one. **(b)** Two isolated but close sheets are merged.

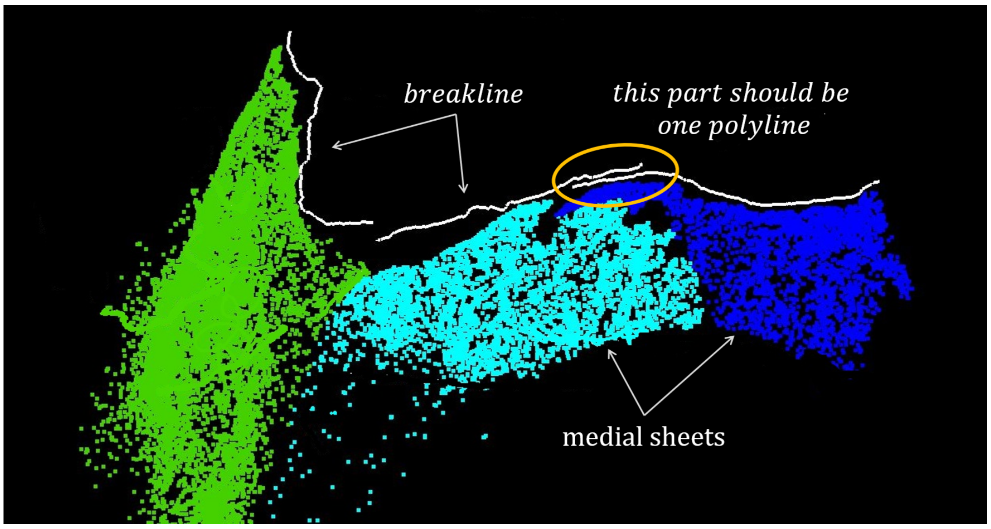
Figure 4.8: Two medial sheets are wrongly segmented as one sheet (side view).


medial sheets
several medial sheets are merged into one
valley
view point
a part of sheet is separated into another sheet

The valley is generated with graph theory without topology, other parameters: ”segmentation method” = ”bisector”, $\vec{b}_t = 3$, $c_n = 100$, $\alpha_t = 15$, $r_2 = 45$)

Figure 4.9: valley, exterior medial sheets and terrain point cloud.

Figure 4.10). The two polylines should be merged where they are parallel.



The breakline(ridge) is generated with graph theory without topology, other parameters: "segmentation method" = "bisector", $\vec{b}_t = 3$, $c_n = 130$, $\alpha_t = 15$, $r_2 = 45$)

Figure 4.10: Adjacent medial sheets and generated breakline.

4.5 CANDIDATE POINT EXTRACTION

This section consists of two parts. Section 4.5.1 discuss the performance of the "candidate point extraction" step. Section 4.5.2 shows another method to get candidate points, and points out the weakness of this method.

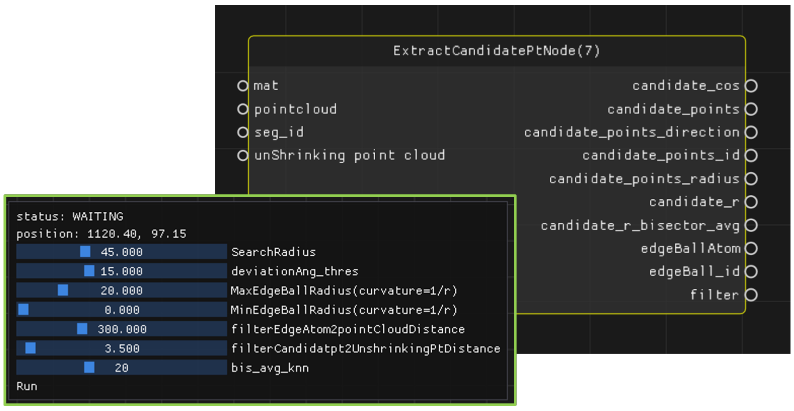
4.5.1 Experiments

The candidate point extraction step contains 7 parameters (also see Figure 4.11 and Table 4.1):

1. deviation angle threshold: α_t, default $\alpha_t = 15$;
2. search radius: r_2, default $r_2 = 45$;
3. Maximum edge ball radius: r_x;
4. Minimum edge ball radius: r_n, default $r_n = 0$;
5. Minimum distance to unshrunken point: du_n;
6. Minimum distance to point cloud: dp_n;
7. bisector average factor: k, default $k = 10$.

α_t and r_2 are involved in the edge ball detection part.
As explained in Section 3.4.1, r_2 prefers a large value, and it is related to the point cloud resolution. Figure 4.12 shows the detected edge balls with different r_2. In the two side views, it is obvious that smaller r_2 contains more unexpected points, more specifically, points not on the edge of a sheet. It is because a smaller search radius can not find neighbours outside the radius.

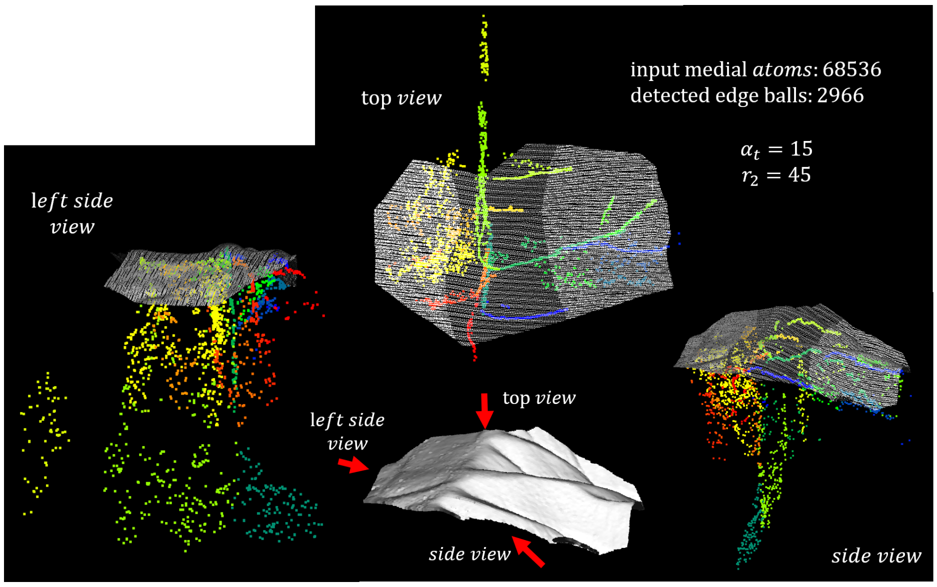
The unexpected point can be filtered out by r_x, r_n, du_n and dp_n (see Figure 4.13).



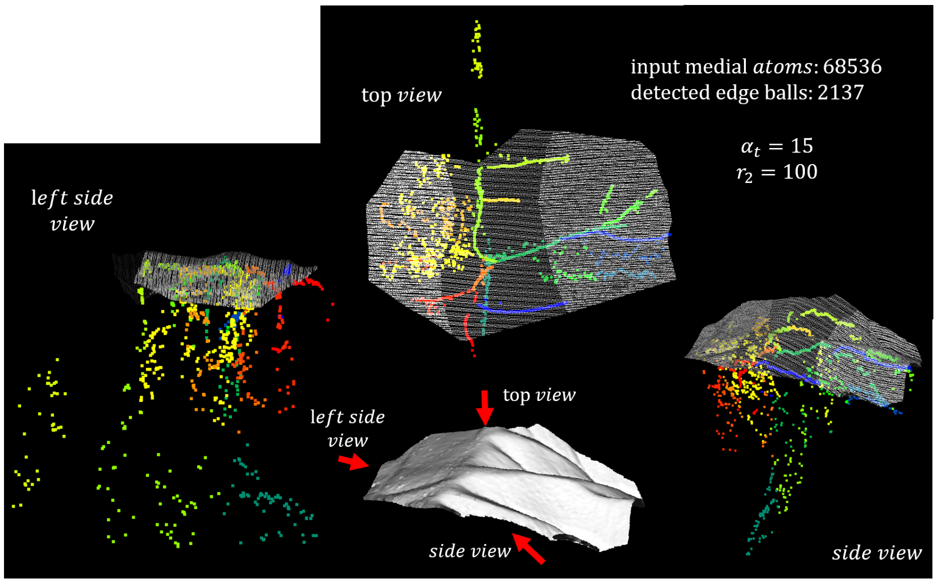
```
ExtractCandidatePtNode(7)
mat                                         candidate_cos
pointcloud                                  candidate_points
seg_id                                      candidate_points_direction
unShrinking point cloud                     candidate_points_id
                                            candidate_points_radius
                                            candidate_r
                                            candidate_r_bisector_avg
                                            edgeBallAtom
                                            edgeBall_id
                                            filter

status: WAITING
position: 1120.40, 97.15
45.000    SearchRadius
15.000    deviationAng_thres
20.000    MaxEdgeBallRadius(curvature=1/r)
0.000     MinEdgeBallRadius(curvature=1/r)
300.000   filterEdgeAtom2pointCloudDistance
3.500     filterCandidatpt2UnshrinkingPtDistance
20        bis_avg_knn
Run
```

Figure 4.11: Candidate point extraction and its thresholds.



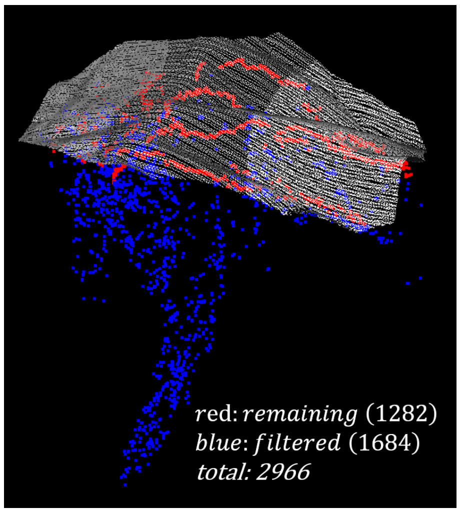
(a) $r_2 = 45$.



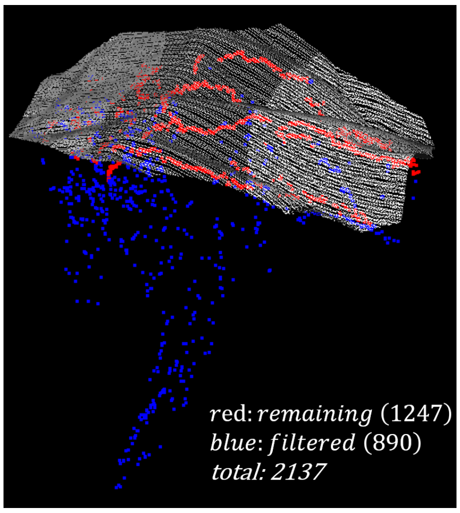
(b) $r_2 = 100$.

Figure 4.12: Detected edge ball.

The two filtered results get similar edge ball points. It proves that with the same α_t and filter parameters, the remaining edge ball atoms are similar, and the influence of different r_2 can be eliminated with filtering.
Thus a question might arise: is it possible to get edge balls by filtering medial atoms without edge ball detection? To answer this question, this section uses the same four thresholds as in Figure 4.13 to filter medial balls of each medial sheet. Figure 4.14 shows the result.
From the image without edge ball detection, the output points are distributed like a medial sheet. To be noticed, as Section 3.1 explained, only the edge ball's radius stands for the curvature, and other balls not. Except for that, is it possible to calculate candidate point coordinates from the "sheet like" points? This question shares common properties with Section 4.5.2, and will be answered there.



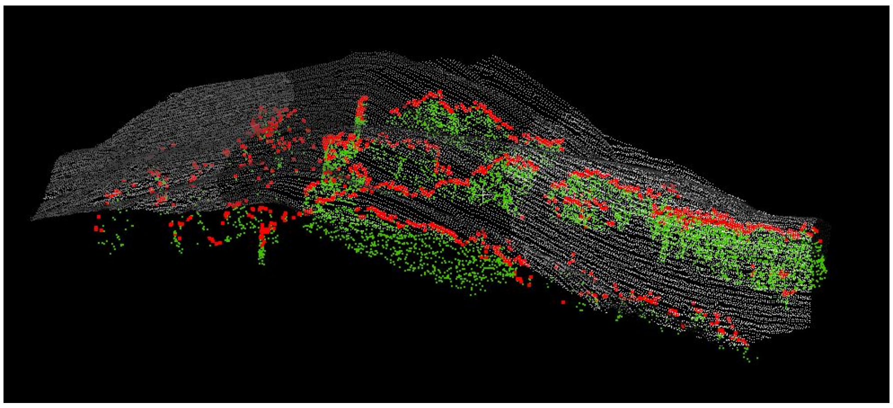
(a) $r_2 = 45$.



(b) $r_2 = 100$.

$\alpha_t = 15$, filter parameters: $r_x = 28$; $r_n = 0$; $du_n = 300$; $dp_n = 3.5$.

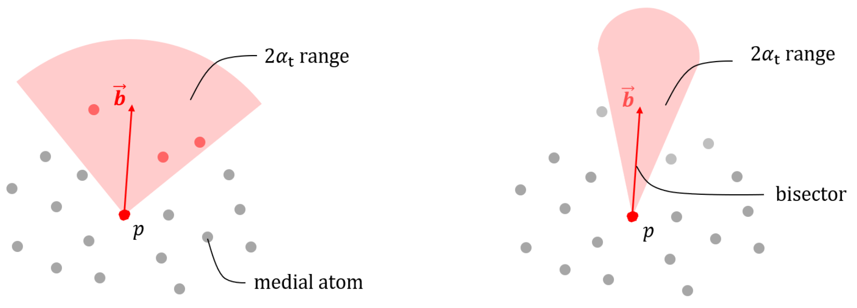
Figure 4.13: Remaining and filtered edge ball atoms with different r_2.



filter parameters: $r_x = 28$; $r_n = 0$; $du_n = 300$; $dp_n = 3.5$.

Figure 4.14: The medial atoms with edge ball detection (red), and without edge ball detection (green).

The value of α_t also influences the result. As Figure 4.15 shows, a more strict α_t will provide more edge medial balls.
The effect of different α_t is illustrated in Figure 4.16. It shows that the number of detected edge balls with different α_t of a terrain point cloud (total number of medial

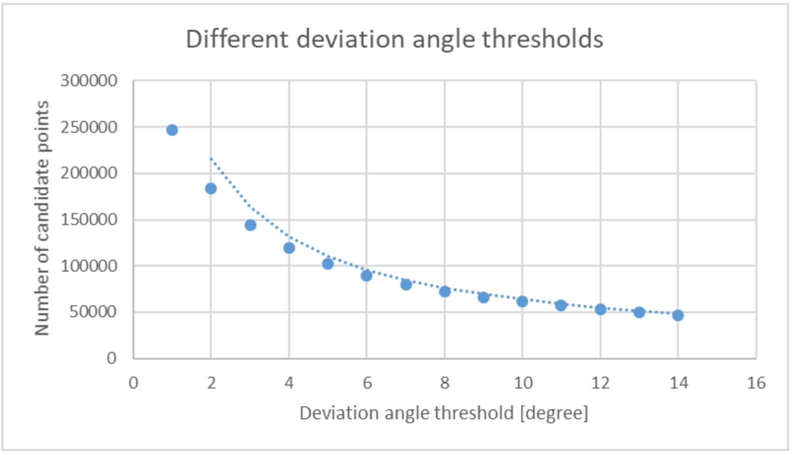


(a) p is not an edge ball with big α_t,　**(b)** p is an edge ball with small α_t

Figure 4.15: Big and small deviation angle thresholds α_t

atoms 336839). With an increased α_t, the number of detected edge medial balls are reduced. But very few edge balls might result in not enough candidate points for further processing.

Figure 4.17 shows the candidate points with different deviation angle threshold



the total input medial atoms size is 336839.

Figure 4.16: Effect of different deviation angle thresholds.

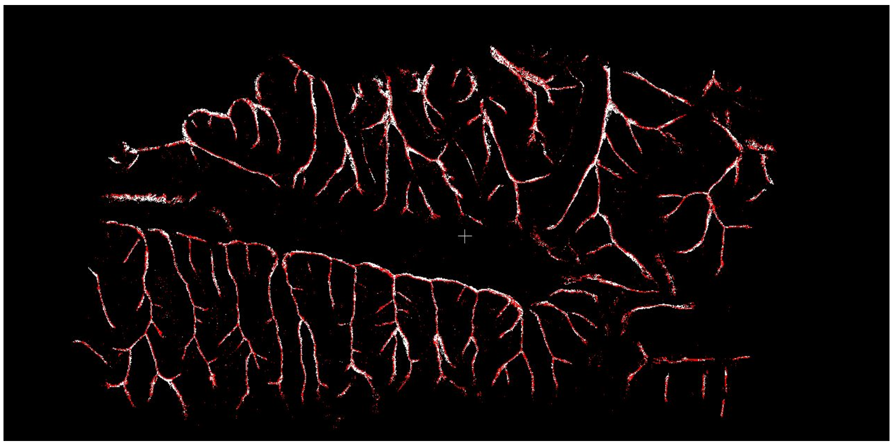
(without filtering candidate points far away from the terrain, as explained in Section 3.4.2). In Figure 4.17a and Figure 4.17b, the points from a larger threshold can not cover the points from a smaller threshold. It means the points from a larger threshold are more aligned to the breakline. In Figure 4.17c, the candidate points from the small threshold still looks like a "sheet".

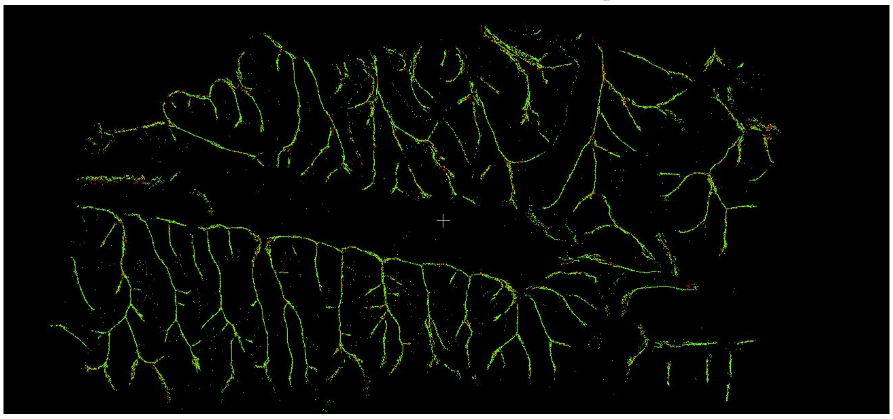
Therefore a result with more candidate points might also contain bad points that are less aligned with the true breakline. Hence, a larger deviation angle should be implemented. In general, α_t can take a default value for all data, but tuning the value for different input point clouds may yield a better result.

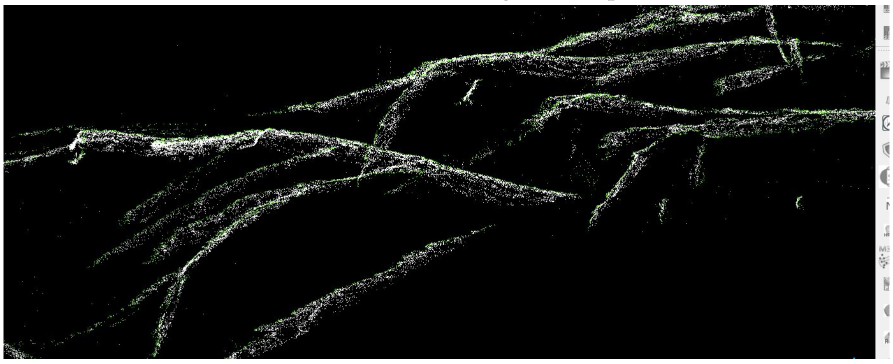
4.5.2 Calculating candidate point coordinates with all the medial atoms

This project also tried another method to extract candidate points: **For every medial sheet, taking all the medial atoms into the process to calculate the candidate point coordinates.**

In this way, every medial atom will produce one candidate point. The resulting candidate points are shown in Figure 4.18. The output points cover an area on the ground, and are not distributed like a line any more.


(a) $\alpha_t = 1$(white) and $\alpha_t = 10$(red)(Top view)

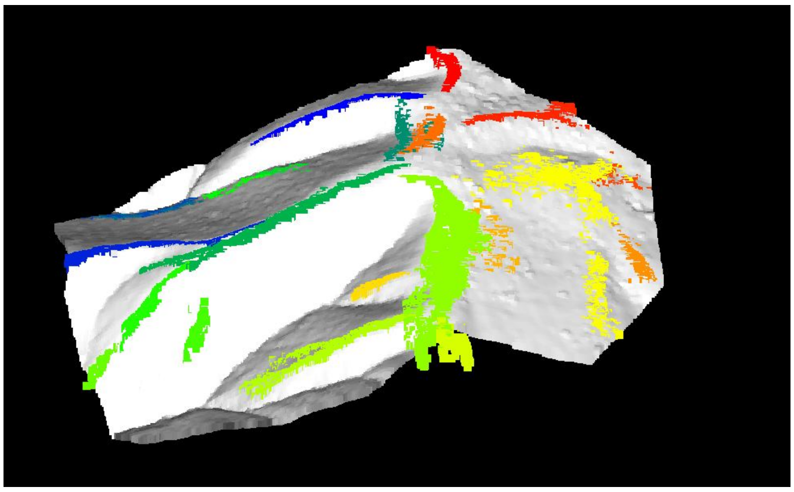

(b) $\alpha_t = 10$(red) and $\alpha_t = 14$(green) (Top view*)


(c) $\alpha_t = 1$(white) and $\alpha_t = 14$(green) (side view*)

*In the top view, if two points are in the same position, only one can be seen.

Figure 4.17: Candidate points with different deviation angle thresh (α_t).

The wide bands of points will cause problem for fitting a polyline (see Figure 4.19).

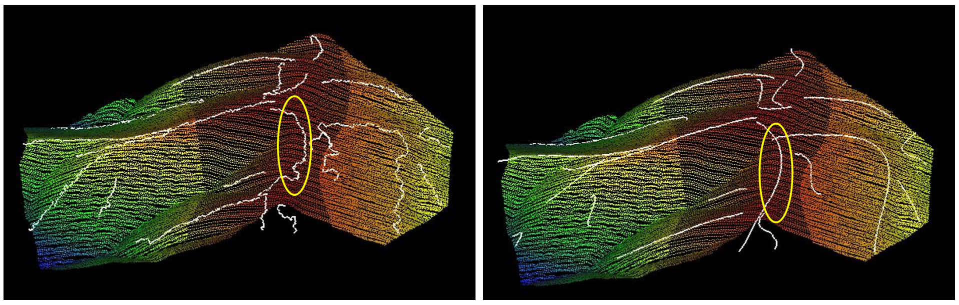


colors stand for points from different medial sheets.

Figure 4.18: Candidate points from all the medial atoms.

The graph theory result is much more zigzagging. The highlighted part does not correspond with the terrain in neither case.

As to the question in Section 4.5.1, are the "sheet like" distribution points (obtained



(a) Generating breakline with graph theory method **(b)** Generating breakline with polynomial fitting

Figure 4.19: Generating breakline from all atoms

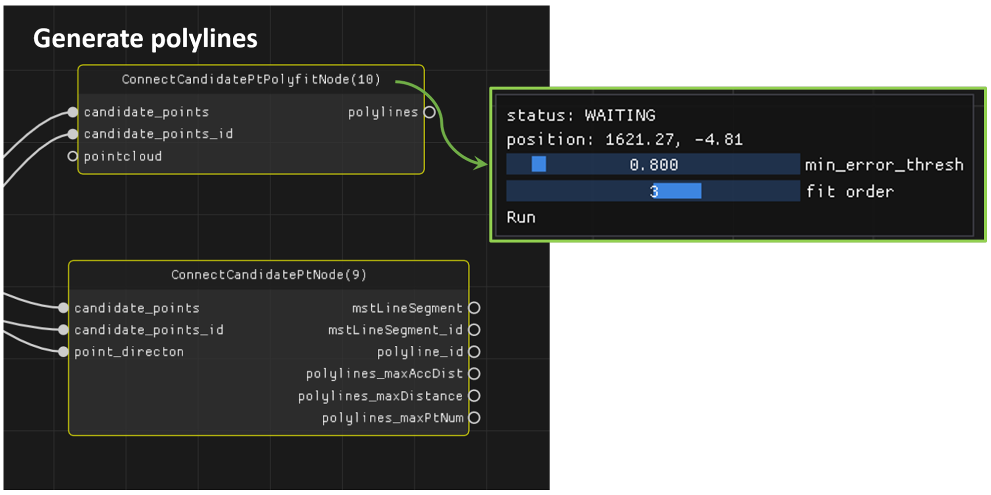
by filtering medial atoms for each sheet) suitable for calculating candidate points? Using all the medial atoms as input points to calculate the candidate point coordinates is an extreme case of the question. The input points disperse wider along z axis, the output candidate points are distributed wider band on the terrain surface. Whereas the input points disperse narrower along z axis, the output candidate points are distributed narrower on the terrain surface.

Thus the "sheet like" distribution points is inconvenient for candidate points. The extracted "edge ball" atoms can be seen as $depth = 0$, and the resulting candidate points are distributed like a line most. The edge ball detection step is thus necessary.

4.6 POLYLINE GENERATION

Generating breaklines with graph theory is automatic, and does not involve user defined parameters. The result of this method has already been displayed in Section 3.5, and it will not be repeated now.

This section focus on the performance of generating breaklines with the polynomial fit method. Section 4.6.1 compares the breaklines from polynomial fitting of different degrees.



```
status: WAITING
position: 1621.27, -4.81
0.800        min_error_thresh
3            fit order
```

Figure 4.20: Polyline generation and related thresholds.

4.6.1 Comparing polynomial fitting with different degrees

The polynomial function with different degree i is:

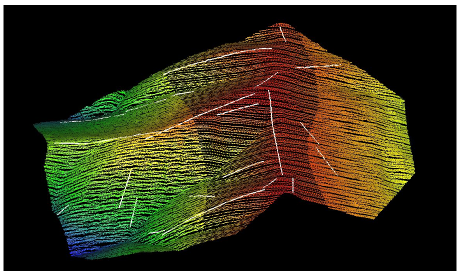
$$y = f(x) = \sum_{0}^{i} t_i \cdot x^i \tag{4.1}$$

The output breaklines from degree 1 to 5 degree are shown in Figure 4.21.
The quintic polynomial (degree = 5) might overshoot for some breakline. Thus it is not a good option.
The average RMSE for all polylines (including removed by $RMSE_t$) is decreasing with increasing degree . With the same $RMSE_t$, quartic and cubic polynomial keep one more breakline than quadratic and linear polynomials. It proves that a higher degree of polynomial fitting a better function. But a better function dose not means a more reliable breakline.
Quadratic and linear polynomial generate rough polylines, which are not suit the terrain surface very much.
The polylines from quartic and cubic polynomial fitting look better. They get larger curvature than the other two. Quartic consists of more curves than cubic, but it does not fitting the ground better. In addition, the natural mountain breaklines usually do not contain such strong curve.
Therefore cubic polynomial fitting is the best option.

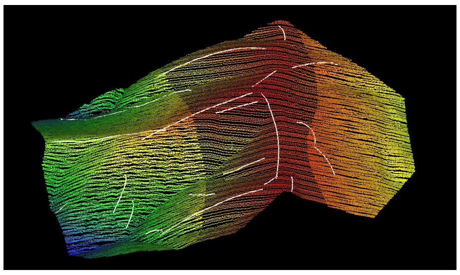
4.6.2 Generating polyline with medial direction

As Section 3.3.2 explained, the medial direction $\vec{n_c}$ of an edge ball can be approximately seen as the directions of maximal principal curvatures of the point ($\vec{v}$) on the breakline. As Section 2.1.4 Figure 2.8 explained, $\vec{n_c}$ will point to the lower direction of a breakline or the upper direction of the breakline.
In practice, this project does not take advantage from this property. But several ideas might improve the result and are worth trying:

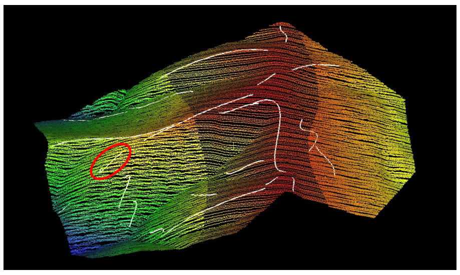
- Connecting the candidate points by using $\vec{n_c}$ (see Figure 4.22).
  Similar to Algorithm 3.1 (finding trace line by point's bisector) and the research of Pang et al. [2012] (see Section 2.1.4 and Figure 2.8). The polyline can



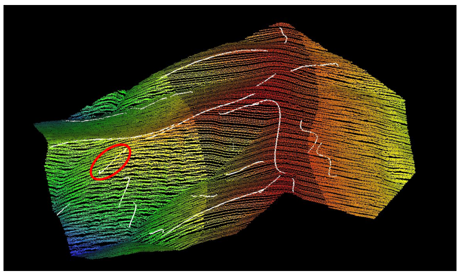
(a) $i = 1$; $avg(RMSE) = 0.354$



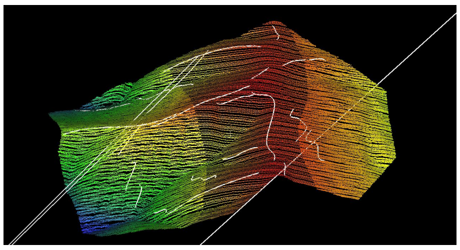
(b) $i = 2$; $avg(RMSE) = 0.291$



(c) $i = 3$; $avg(RMSE) = 0.258$



(d) $i = 4$; $avg(RMSE) = 0.248$



(e) $i = 5$.

i: the degree of polynomial function; $RMSE_t = 0.8$;
$avg(RMSE)$: the average RMSE for all polylines (including removed by $RMSE_t$).

Figure 4.21: Breaklines generated with different degree of polynomial function

be generated following $\vec{n_c}$. The output polyline might consist of less candidate points, but the polyline will be less zigzagging.

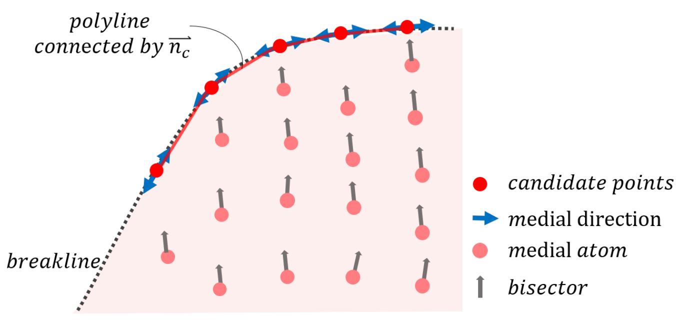


Figure 4.22: Profile section view of a medial sheet and $\vec{n_c}$.

- In Section 3.5.1 "Connecting candidate points with the minimum spanning tree", the weight of an edge (v_1, v_2) is confirmed by:

  $$weight = distance(v_1, v_2) \tag{4.2}$$

  Because the breakline should approximately flow the $\vec{n_c}$ or $-\vec{n_c}$. The edge direction that is similar to $\vec{n_c}$ should get heavier weight. Therefore the weight can be improved:

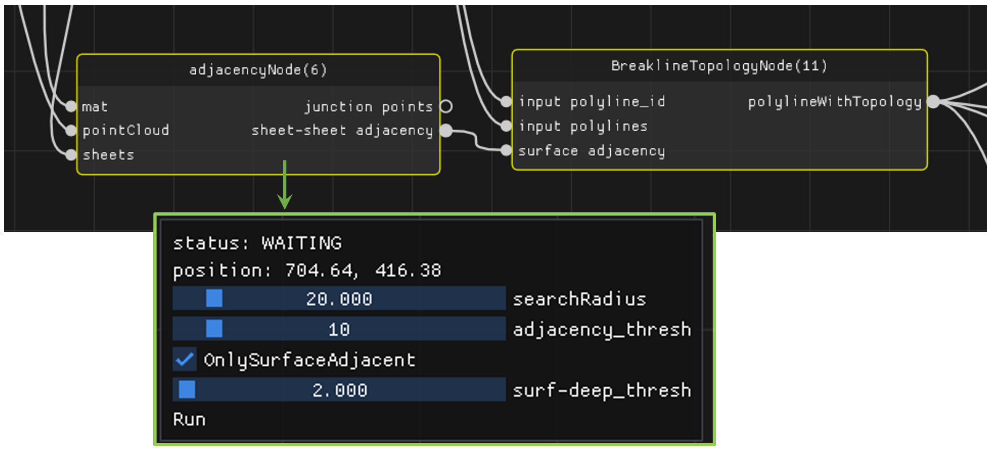
  $$weight = \alpha \cdot distance(v_1, v_2) + \beta \cdot Angle(\vec{n_c}, \vec{v_{12}}) \tag{4.3}$$

  where α, β are weighting parameters;
  $\vec{v_{12}}$ is the vector starting from vertex v_1 ending at v_2;
  $Angle(\vec{n_c}, \vec{v_{12}})$ is the angle between the two vectors.
  The advantage of this method is that it takes both the candidate points' spatial distribution and the directions of maximal principal curvatures ($\vec{n_c}$) of the candidate points into consideration. While the disadvantage is that Equation (4.3) includes two new parameters, thus a user needs to define more variables by himself.

4.7 SURFACE ADJACENCY AND THE TOPOLOGY OF BREAK-LINES

In order to get the surface adjacency relationship, three parameters are required (see Figure 4.23 and Table 4.1).
The search radius (r_1) and the minimum number of neighbour points (ac_n) are two related parameters. r_1 is to find the neighbours of every medial atom, and a larger r_1 will find more neighbours belonging to another medial sheet. The correlated ac_n should be larger to determine if two medial sheets are adjacent.
The maximum distance of neighbour points to the terrain (dp_a) determines if two adjacent medial sheets are surface adjacent, and it is related to point cloud resolution.
Figure 4.24 shows the result of connecting breaklines by surface adjacent medial sheets. Some connections are good estimations, and some problems of connecting to the closest points method are exposed:

1. **rough angle**
   Because two breaklines are connected with their closest point pairs, the added connecting line segment and the ordinary polyline create a rough angle.
   An idea to solve this problem is to fit a polynomial functions for two polyline



```
status: WAITING
position: 704.64, 416.38
20.000    searchRadius
10        adjacency_thresh
✓ OnlySurfaceAdjacent
2.000     surf-deep_thresh
```

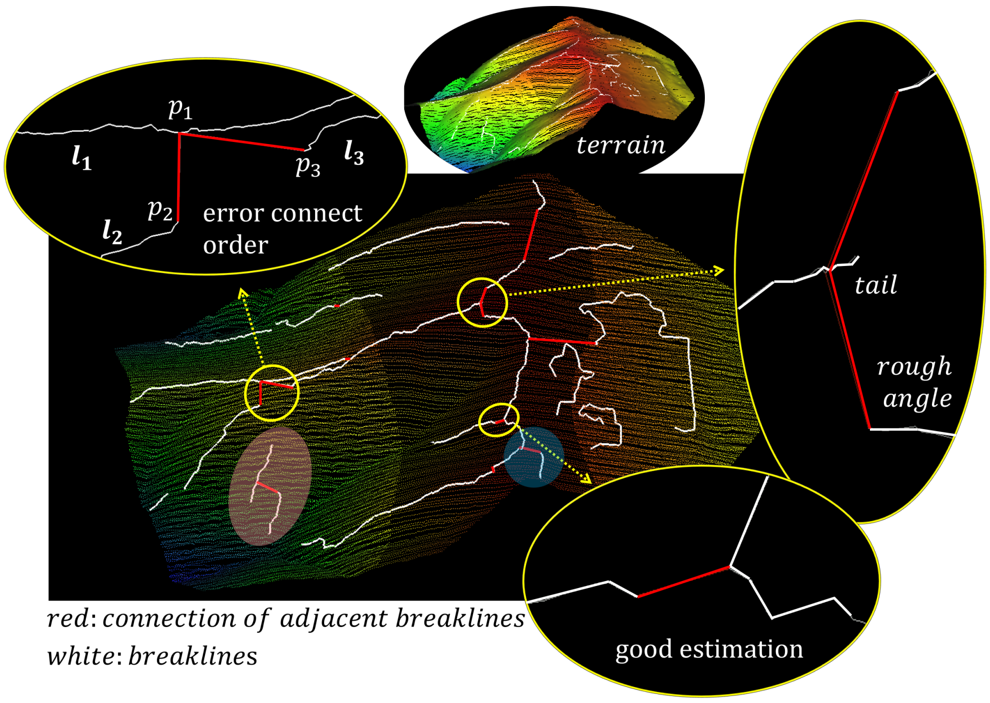
Figure 4.23: MAT adjacent, breakline topology and related thresholds.

in 2D, and the estimated junction point $\hat{j}$ could be the intersection of those two functions. This idea might solve the "rough angle", but will cause other problems:
(1) When two adjacent breakline are approximately parallel to each other (see the pink area in Figure 4.24), $\hat{j}$ will be far away from the two polylines, and might even cross other breaklines.
(2) The fitted function pair might not intersect (see the blue area in Figure 4.24).

2. **tail**
   For some pairs of adjacent polylines, the closest point is not the best estimation of junction point, but the free vertex [9] is. It will create a tail at the end of a polyline.
   A possible solution is adding a new user defended parameter, the minimum tail length (l_t). When the tail is shorter than l_t, the junction will be the closest free vertex instead of the closest point. But this involves a new hard threshold again.

3. **connection order error**
   In the image, (l_1, l_2) and (l_2, l_3) are two pairs of adjacent breaklines. l_2 is connected to l_1 first, then l_2 is updated by adding a new point p_1 after p_2. l_2 and l_3 are connected after that. Because l_2 was changed, and the point of l_2 closest to l_3 is no longer p_2 but p_1. With this process, the angle $p_2p_1p_3$ is created.
   But actually (l_2, l_3) should be connected first, and the angle will not appear.

9 start or the end point of a polyline, explained in Section 3.5



$r_1 = 20$; $ac_n = 10$; $dp_a = 150$; breakline is generated with graph theory method .

Figure 4.24: Connection of adjacent breaklines

5 RESULT VALIDATION AND COMPARISON

This chapter focuses on the extracted breakline and the breakline's quality. Section 5.1 demonstrate the ridges and valleys of two test datasets. Section 5.2 compares the extracted breakline with other three method: manually drawn (Section 5.2.1), ArcMap (Section 5.2.2) and CGAL package [1] (Section 5.2.3). Section 5.3 explains the validation method (section 5.3.1) first, and then shows the validation result of dataset 1 (Section 5.3.2).

5.1 EXTRACTED BREAKLINE OVERVIEW

Figure 5.1 and Figure 5.2 show the extracted ridges and valley for data 1 and data 2.

In general, this project is able to extract ridges and valleys in both two test data with the graph theory method and polynomial fitting method. Both two results with the graph theory method are more zigzagging, and the polyline simplify and smooth are implemented. The advantage of generating breakline with polynomial fitting method is removing unexpected breaklines with $RMSE_T$. Meanwhile, the disadvantage is the result breaklines are too smooth and contains more polyline crosses. Those are not suit the reality.

Data 2 takes a smaller segmentation threshold, and a smaller c_n, and keeps more detail (small) breaklines. The breaklines with graph theory from data 2 looks more complete and smooth. It is because data 2 contains a larger area, a larger point cloud, a higher point density, a bigger height difference (see Table 4.3).

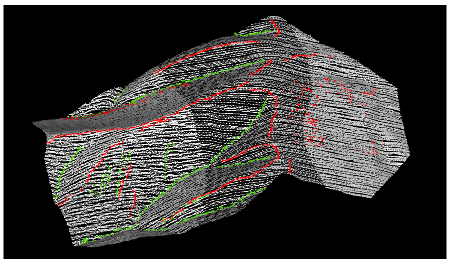
Examining the result breakline carefully, three kinds of problems can be found:

1. **original breakline shift**
   Observing the original breaklines (without smooth and simplify), one ridge is shift from the ground truth (see Figure 5.3). But in the corresponding medial sheet, the medial atoms don't shift. The zoom in image shows the medial atom and the bisector, and the bisector is variant a lot at that area. It cause the shifting. The bisector variation might because of the ground points containing noise (for example small vegetation or animals appearing etc).
   This kind of shifting is eliminated with the polyline smooth and simplify. But it also pay by lose points or accuracy of other polylines.

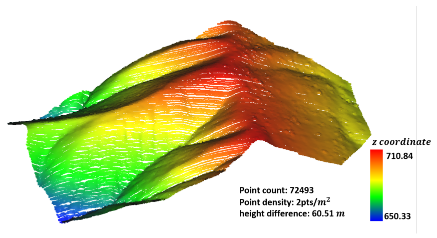
2. **incomplete small breaklines**
   There are some small ridges and valleys in the terrain, but only some of them will be extracted (see Figure 5.4). The related medial sheets containing little atoms will be filtered during the segmentation process (see Section 4.4). In addition, for those small area, to successfully separate medial sheets requires a more strict segmentation threshold. But the threshold is for the whole dataset, and it will influence other sheets (see Section 4.4).

3. **a ridge and a valley cross each other**
   The ridges or valleys do not cross each other. But in the result, a ridge and a valley might across. It appears twice in data 1 (see Figure 5.4). It might

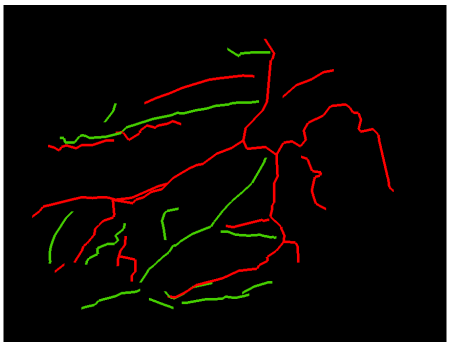
1 explained in Section 2.1.3



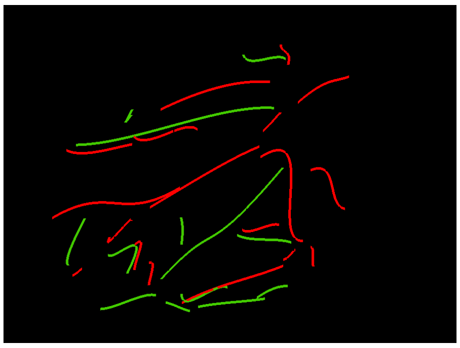
(a) candidate points



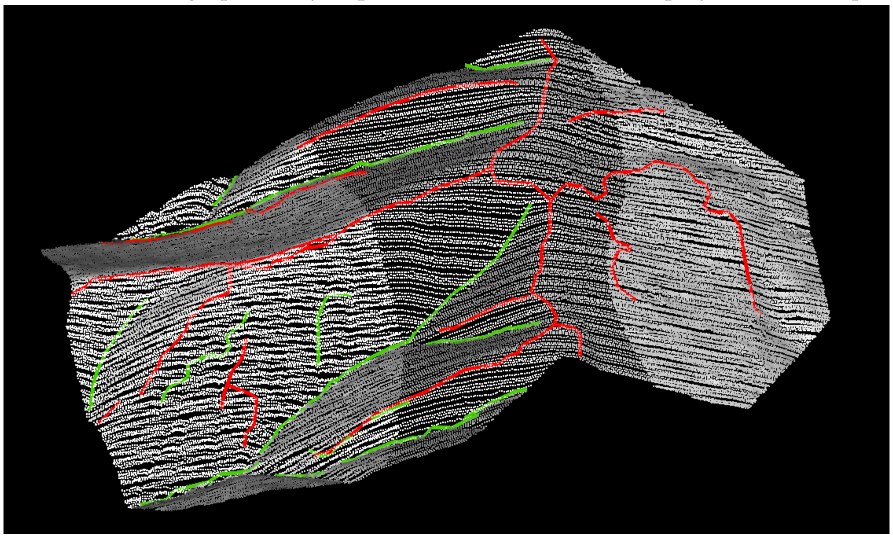
(b) terrain



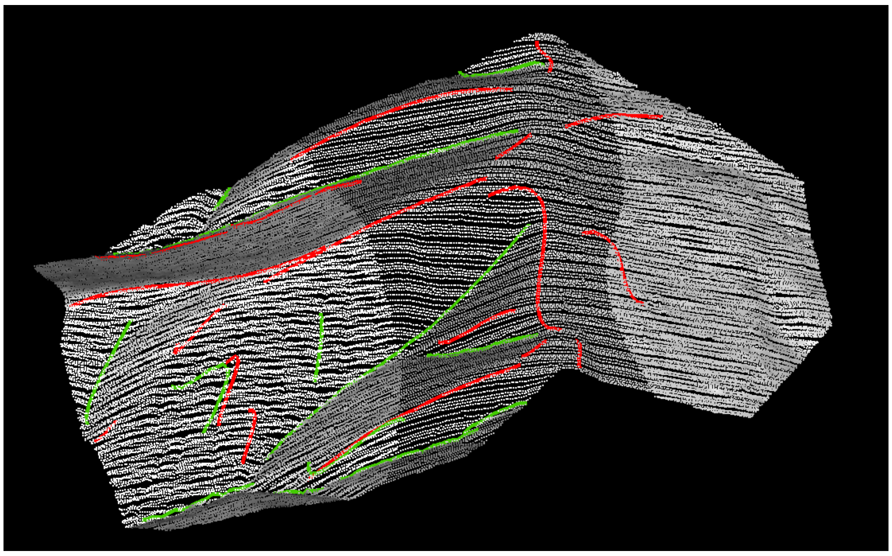
(c) breakline with graph theory (top view)



(d) breakline with polynomial fit (top)



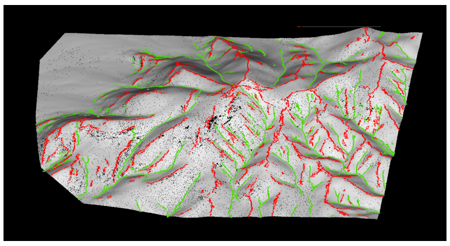
(e) breakline with graph theory (with topology, smooth and simplify)

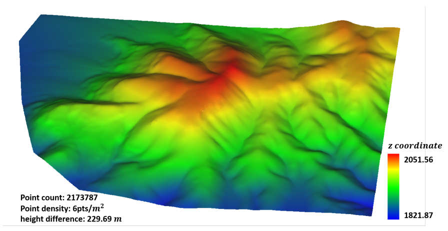


(f) breakline with polynomial fit

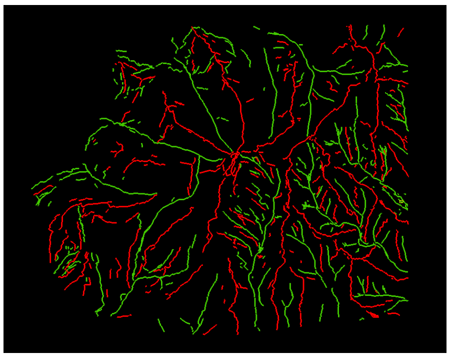
generate 20 ridges and 13 valley "segmentation method" = "bisector", $\vec{b}_t = 3$, $c_n = 110$, $\alpha_t = 15$, $r_2 = 45$)

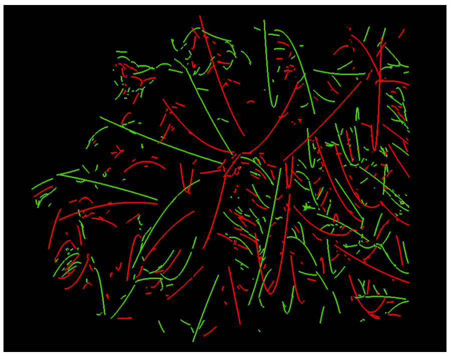
Figure 5.1: Ridge (red) and valley (green) of test data 1.

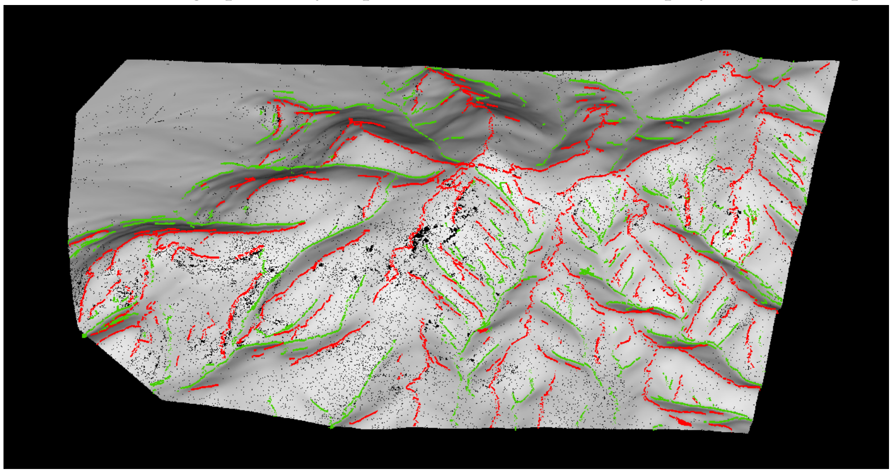

(a) candidate points

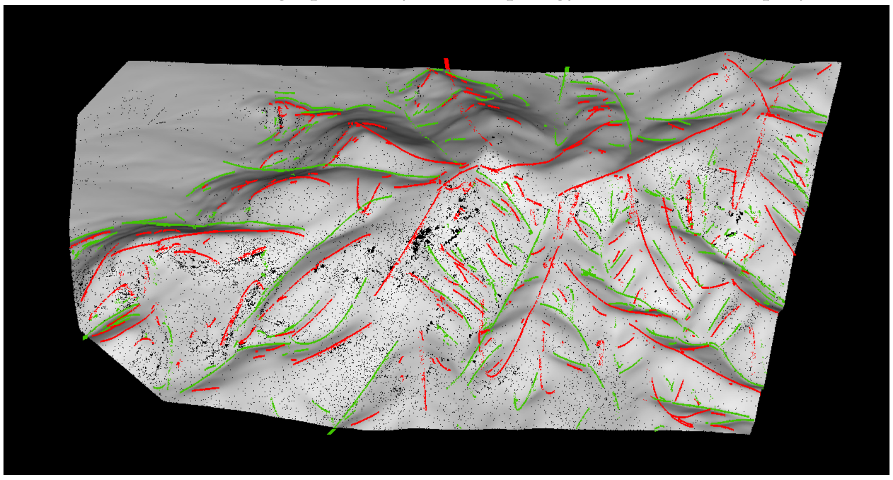


(b) terrain


(c) breakline with graph theory (top view)


(d) breakline with polynomial fit (top)


(e) breakline with graph theory (with topology, smooth and simplify)


(f) breakline with polynomial fit

generate 297 ridges and 324 valleys "segmentation method" = "bisector", $\vec{b}_t = 2.5$, $c_n = 70$, $\alpha_t = 20$, $r_2 = 100$)

Figure 5.2: Ridge (red) and valley (green) of test data 2.

because the **interior and exterior MAT does not separately completely**.
The medial sheet bellow the terrain point cloud, but belongs to the exterior MAT. But according to the MAT definition, the medial atoms below the terrain is the interior MAT. This miss classification sheet generates a "valley" which should be a ridge actually. The "valley" is overlapping with the the ridge generated from the correct classified interior MAT, the two breakline should be one ridge.

4. **The breakline indicated by the related medial sheet does not exist in the terrain** (see Figure 5.5).
Observing the medial sheets and the terrain carefully, there is no ridge but only valley in the area. In Figure 5.5, some of the point cloud normal vectors that are close to the breakline point to the exterior of the terrain, but some of the normal vectors point to the exterior of the terrain. According to the "shrinking ball algorithm" (Section 2.2.3), it will cause the "interior and exterior error". Similar to the previous item, the generated "ridge" should be a valley and merged with the an other valley.

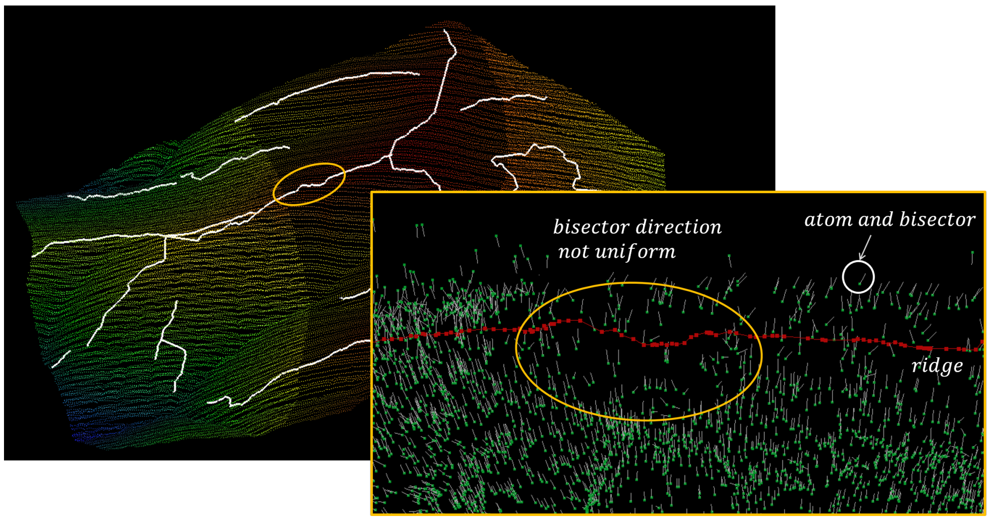


Figure 5.3: breakline shifting (example: ridge of data 1)

5.2 COMPARISON

Section 5.2.1 compares the extracted breakline with the manually drawn breakline for the small test data, data 1. Section 5.2.2 compares with the commercial software ArcMap. Section 5.2.3 shows the extracted breakline with CGAL package.

5.2.1 Comparison with manually drawn breakline

Figure 5.6 shows the manually drawn ridge and valley for test data 1. It uses the same point cloud as the breakline extraction process. An operator manually picks 3D points from the point cloud as breakline vertices with the help of an open source software "cloud compare". Picking original points as breakline vertices can make sure the vertices are exactly on the Terrain surface. The reliability of reference breakline is highly depending on the operator's intention, experience, ability, and perception. In addition, whether two breaklines should be adjacent or not is highly depends on the operator's individual understanding of the dataset, and might be different from operators.

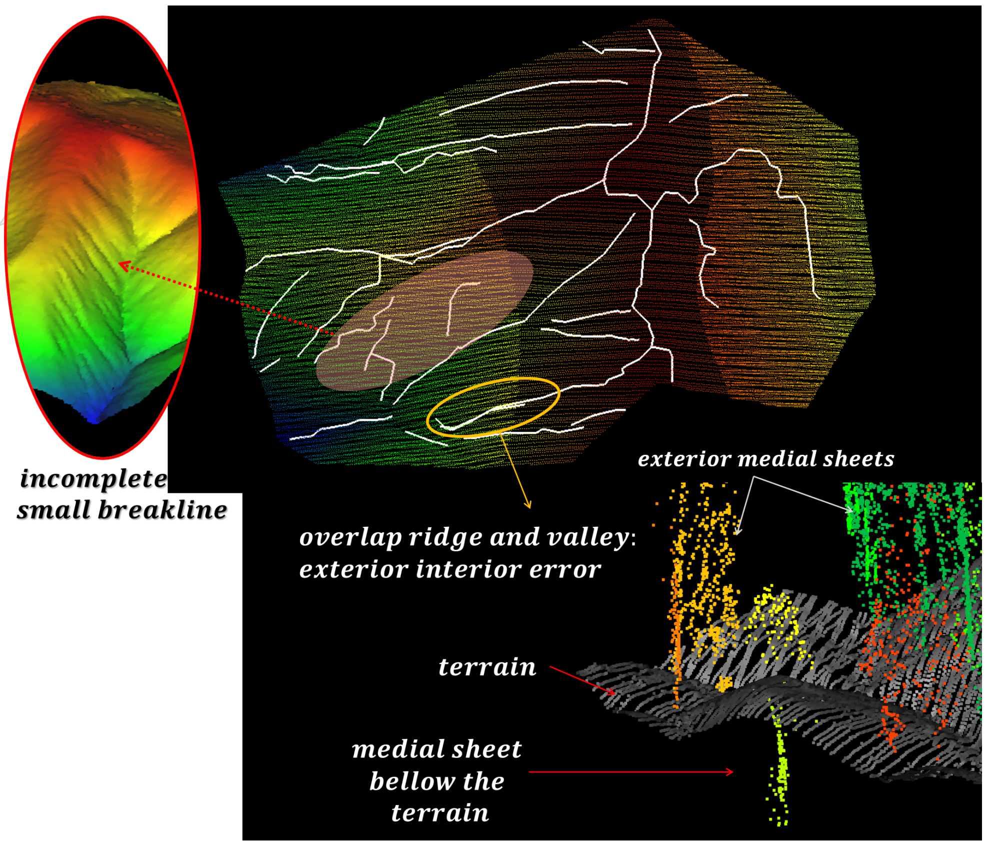


Figure 5.4: Overlap and incomplete breaklines (example: ridge and valley of data 1)

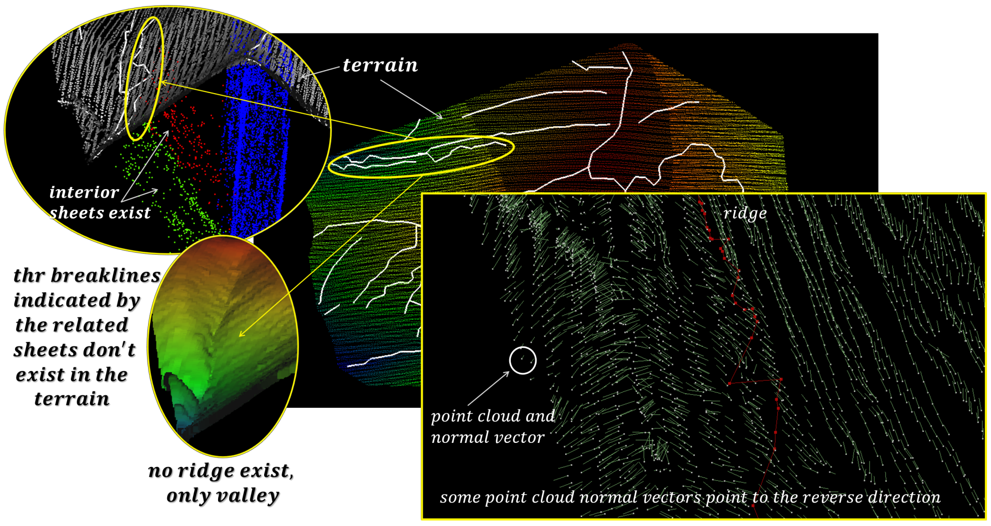


Figure 5.5: Overlap and incomplete breaklines (example: ridge and valley of data 1)

For further observation, the ridge and valley are shown separately in Figure 5.7.

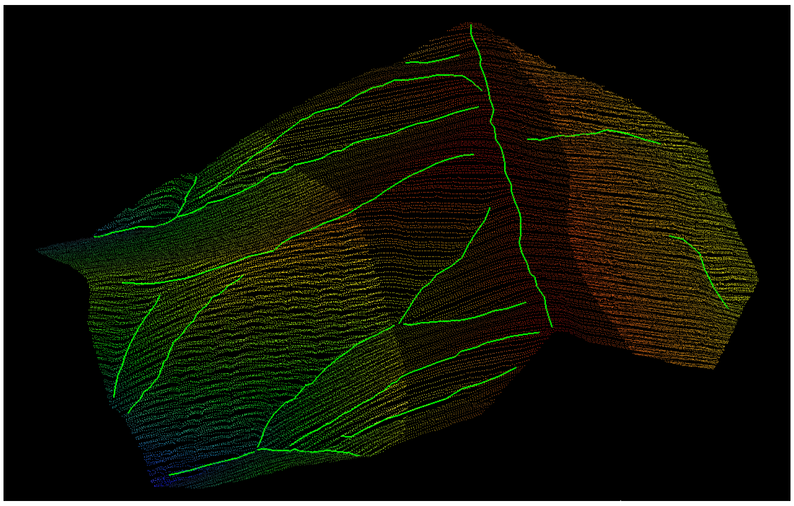


Figure 5.6: Manually drawn ridge and valley for test data 1

Except for the four problems mentioned in Section 5.1, some observations can be found:

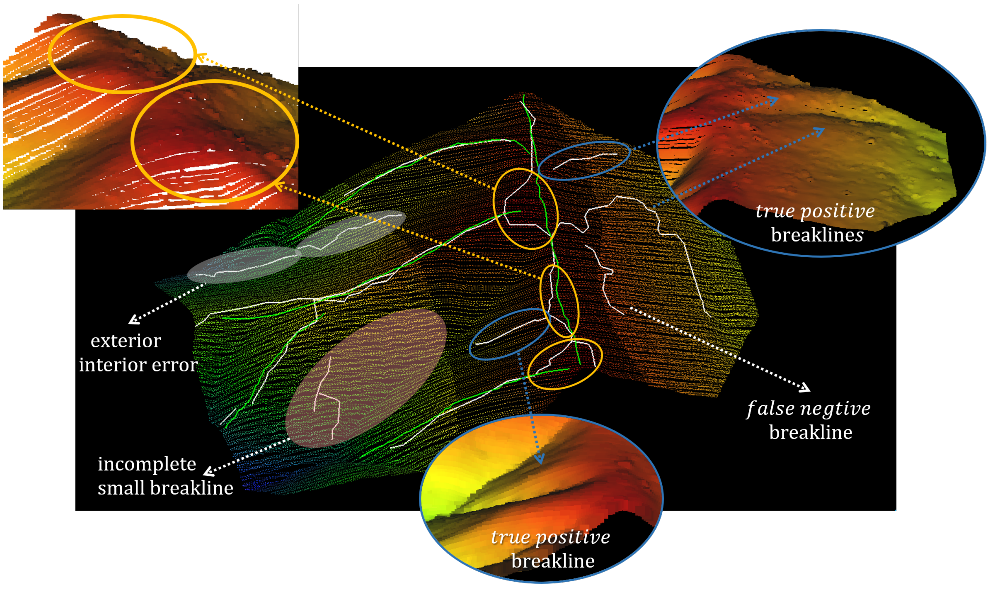
1. **different pattern around mountain peak**
   In Figure 5.7a, around the local peak (orange circles in the image), the manually drawn ridge has a different pattern to the extracted result. The manually drawn ridges pass and end at the local peak, while the extracted ridges will go around the peak.
   After more careful observation of the terrain point cloud, the maximum curvature is around the extracted polylines actually. In addition Section 4.4 Figure 4.6 also proved that the skeleton (medial sheet) of the terrain separates around the peak.
   This might proved that, the manually drawn ridge is effect by the operator, and the peak of a mountain might cause the optical illusions to cause inaccurate in manually drawn ridges in that area. The automatic extraction result in this project is less effected by the optical illusions, and extracting more reliable ridges around the peak.

2. **different topology**
   The two kinds of result breaklines has different topology.
   First, for a same pair of breaklines, the adjacency might be different between them. As this section explained at the beginning, the manually drawn adjacency depends on the operator. The adjacency from extracted breaklines also depends on the user defined parameters (see Section 4.7).Because around the beginning or the end of a breakline, the curvature tends to be more uniform, and the curvature becomes continuous gradually, and the adjacent uncertainty makes sense.
   Second, for one breaklines, it might exist in different forms. One breakline from the manually drawn data might separate into small fragments and becomes a part of horizontal breaklines in the extracted data, vice versa.
   For example, the zoom in "different topology" area in Figure 5.7b, there are three valleys in the manually drawn data, and two of them becomes one valley in the extracted data.

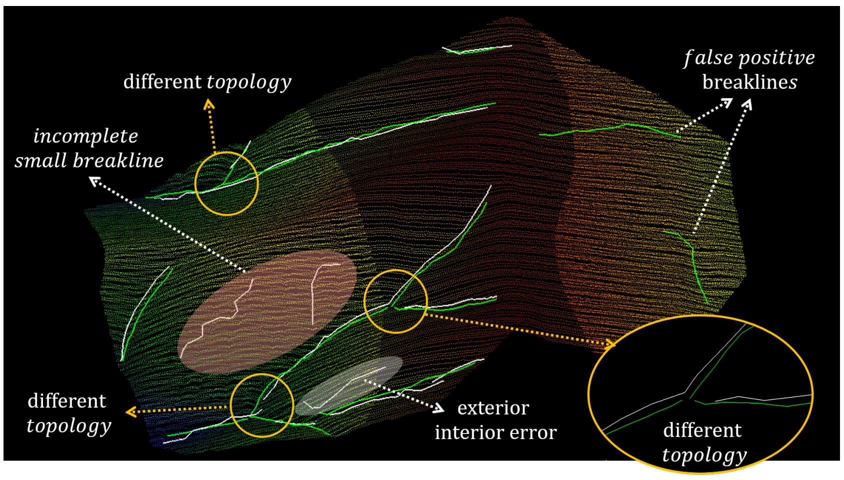
3. **true positive breaklines**
   If there is a breakline in the terrain, the extracted breakline is called "true positive breaklines". There are some true positive breaklines extracted with this project, but neglected by the manually drawn result. For data 1, it happens only for ridge where the breakline is inconspicuous (see Figure 5.7a).

4. **false positive breaklines**
   There are some breaklines founded by the manually drawn result, but neglected by this project. For data 1, it happens only in valley (see Figure 5.7b).

5. **false negative breaklines**
   If there is no breakline in the terrain, but this project extracted one in the related location, the extracted breakline is a "false alarm", and called "false negative" breakline. For data 1, there is one false negative in ridge extraction result (see Figure 5.7a).



(a) ridge



(b) valley

Figure 5.7: Comparison extracted breakline (white) with manually drawn breakline (green) of test data 1.

In conclusion, this project generate more reliable ridges around mountain peak. For test data 1, this project performs more sensitive when generating ridge. For one hand the method is able to find inconspicuous true positive ridges, which is easy to be neglected by the operator. This is the advantage of being sensitive. While in another hand, it also result in "false alarm", more specifically, generating false negative ridges. This is the disadvantage, also the "trade of" for the advantage aspect. For valleys, this method does not create false negative breaklines, but lead to false positive breaklines.

5.2.2 Comparison with ArcGis

At present, there is no commercial software can directly generate breakline for natural terrain. The Hydrology tool embedded in ArcMap can help user generate ridge and valley identify breakline pixels. It uses raster DEM as input, and calculates water flow direction and flow accumulation. The pixels Where the flow accumulation value is 0 is the ridge pixels. The result ridge points is in Figure 5.8.
The valley extraction uses a reverse DEM, other steps are the same as ridge extraction. The reverse DEM is calculated by using maximum height in the region minus the height for each DEM pixel.
Since the AcrMap does not have polylines, comparing with candidate points is fair.

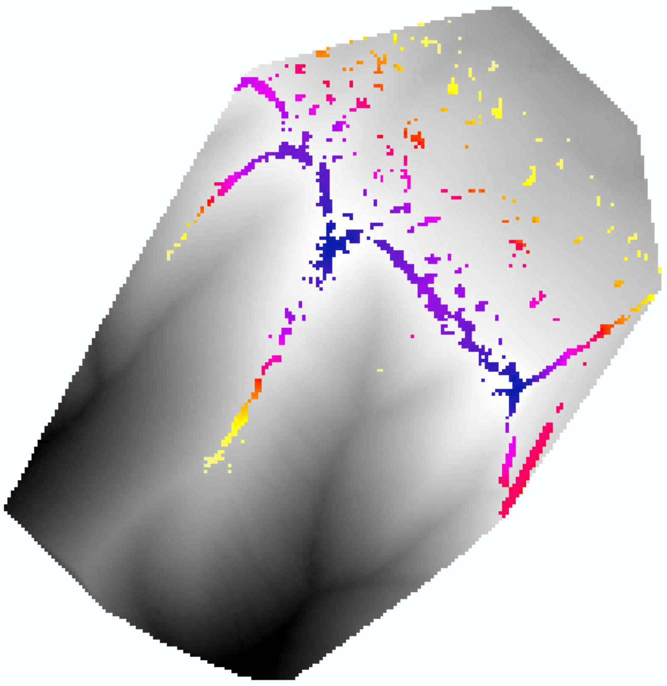


Figure 5.8: Generating ridge by ArcMap (gray: DEM raster; colour: ridge cells).

Figure 5.9 compares the ridge cells from ArcMap with the candidate points of data 1.
Both two results have the same pattern around the mountain peak.
AcrMap does not contain the "false negative ridge" (in the green circle of the image) caused by MAT "interior and exterior error".
But ArcMap result containing 3 wronging identify ridges (in right bottom yellow circle). It caused by the TIN interpolation method to generate raster from point cloud. AcrMap also missed some parts of ridges when they are vanishing (arrows in the figure), and also missed some small ridges (red area). Further more, the result is a set of cells, which does not convert into geometrical features and can not support further spatial topological analyses.

5.2.3 Comparison with CGAL

The CGAL package requires a TIN mesh as input. Data 1 is transformed to mesh with the Delaney triangulation (see Figure 5.10).
CGAL package also requires a lot of user defined parameters. The result is shwon in Figure 5.11.
The CGAL package generating 4055 breaklines which contains a lot of small line segments. A lot of breaklines are cross, which is unexpected.

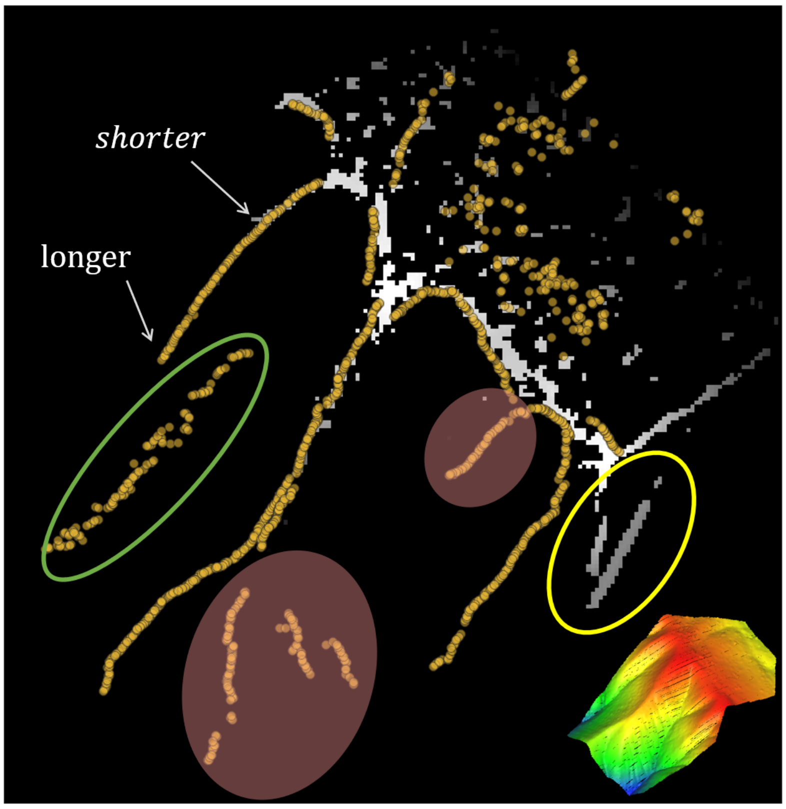


Figure 5.9: The ridge cells from ArcMap(white - grey tone) and the candidate points (yellow) of data 1

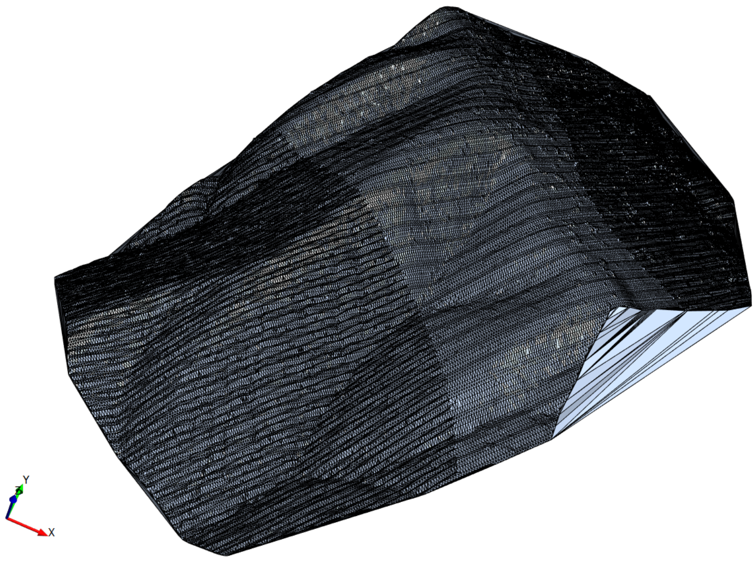


Figure 5.10: The TIN mesh generated from data 1.

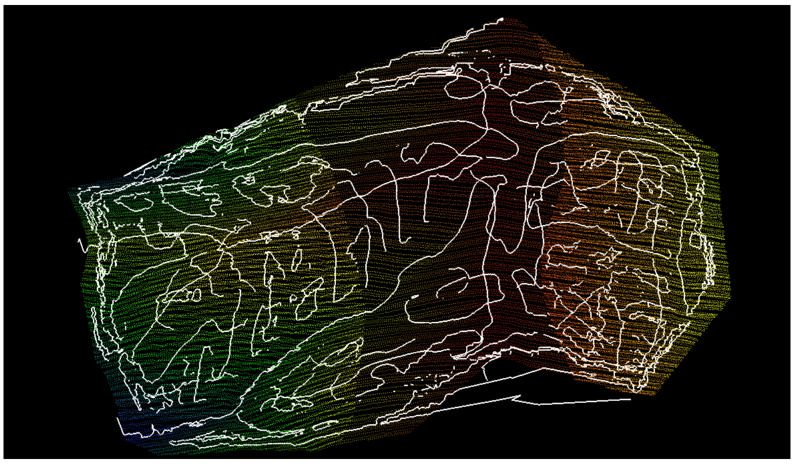


Figure 5.11: The resulting breaklines from CGAL.

It is because the CGAL package requires a smooth mesh as input and the result is data-driven.
While this project is able to extract reliable breaklines with the same point cloud that generates mesh for CGAL. Therefore, this project is less influenced by the input data.

5.3 VALIDATION AND ANALYSIS

This section contains three parts: the validation method, the validation result of dataset 1 and a short summary of the validation result.

5.3.1 Validation method

For the evaluation of the output breaklines, a set of manually drawn breaklines is used as a reference data. The breakline to be evaluation is the test data. Compared with the reference breakline, there are 3 situations demonstrating in Table 5.1.
The validation contain two parts:

		Reference breakline	
		breakline exist	breakline not exist
Test breakline	breakline detected	**TP** (true positive, correctly identified)	**FN** (false negative, incorrectly identified)
	breakline not detected	**FP** (false positive miss identified)	

Table 5.1: Overview of validation

1. if the breakline generated correctly?
   With in this context, the **TF**, **FN** and **FP** stand for the "true positive breakline", "false negative breakline" and "false positive breaklines". This part is manually identified.

2. how is the accuracy and precision of the "true positive breakline"?
   With in this context, the **TF**, **FN** and **FP** stand for the number of points of three different part of the breakline (see Figure 5.12).

In this study, for each manually drawn breakline l and its corresponding test breakline $\hat{l}$, one can create a buffer b and $\hat{b}$ with the same distance to separate the **TF**, **FN** and **FP** part of a breakline:

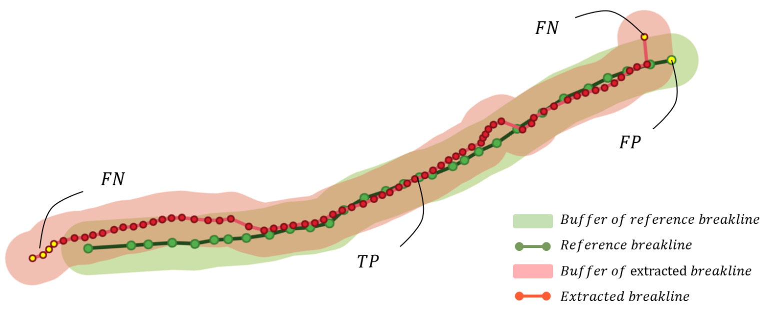


Figure 5.12: Validation

- The vertexes belonging to $\hat{l}$ but not in buffer b are the FN vertexes,
- and other vertexes inside or on buffer b are TP vertexes.
- the vertexes belonging to l, but not in buffer $\hat{b}$ are FP vertexes.

In addition, from both Table 5.1 and Figure 5.12, the number of test breakline vertexes NE is:

$$NE = TP + FN \tag{5.1}$$

Because the the number of vertexes of test breakline is usually ten times larger than the reference breakline, a scalar is used:

$$scalar = \frac{NE}{NR}; \tag{5.2}$$

where NR is the number of reference breakline vertexes.
The accuracy validation can be described in three parameters: correctness, quality and completeness. Those three parameters can be calculated by TP, FP and FN:

$$correctness = \frac{TP}{TP + FP \times scalar} \tag{5.3}$$

$$quality = \frac{TP}{TP + FN + FP \times scalar} \tag{5.4}$$

$$completeness = \frac{TP}{TP + FN}; \tag{5.5}$$

As to the precision can be described by polyline precision (p_l) and vertex precision (p_v). $d(v, r_l)$ is the distance from vertex v to the reference polyline r_l in $3D$. p_l and p_v can be calculated as:

$$p_l = \frac{\sum d(v, r_l)}{the\ length\ of\ l}; \qquad where\ vertex\ v \in TP; \tag{5.6}$$

$$p_v = \frac{\sum d(v, r_l)}{TP}; \qquad where\ vertex\ v \in TP; \tag{5.7}$$

Because the z coordinate of the extracted breakline is obtained by the nearest neighbour spatial interpolation, and the vertices of the reference breakline is from the original point cloud. Therefore both two kinds of breaklines are on the same terrain surface. Thus the buffer can be a 2D polygon. accuracy validation can be processed in 2D xy plane.
In the precision calculation, the distance between vertex v and polyline l is in 3D space, so the precision of z coordinate is also taken into consideration.

5.3.2 Validation result

This section shows the validation result of the breakline generated with graph theory adding the topology and smooth and simplify operation. A more detail validation should also consider the breaklines generated with graph theory adding the topology without smooth and simplify operation, also the breaklines generated with polynomial fitting.
In practice, the manually drawn breakline is influenced by the point cloud distribution (see Figure 5.13). Sometimes the best location of a breakline is not the point from the point cloud. Therefore the reference data is not perfect.

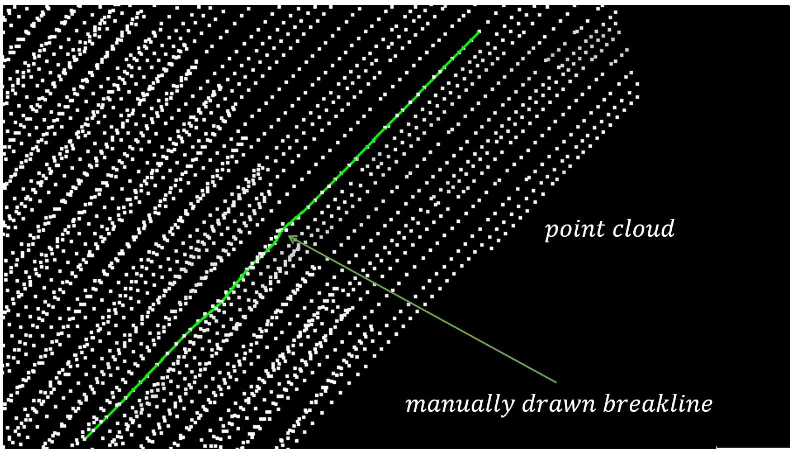


Figure 5.13: The manually drawn breakline is influenced by the point cloud distribution

- **PART 1**: if the breakline generated correctly

The Figure 5.14 shows the reference valleys and the test valleys for data 1. There are 13 test valleys and 18 reference valleys in the image.
The Figure 5.15 shows the reference ridges and the test ridges for data 1. There are

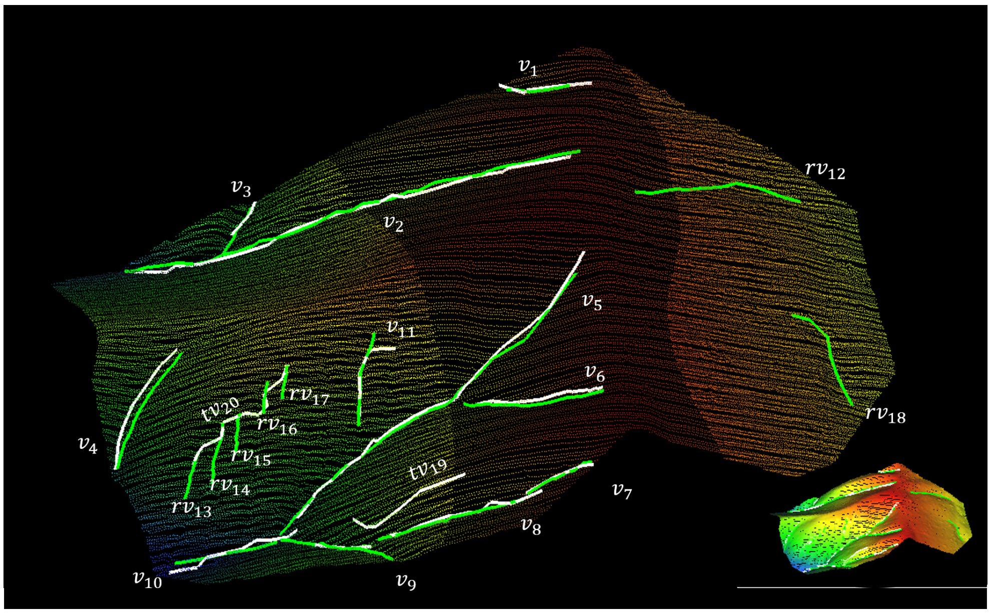


Figure 5.14: The reference valleys (green) and the test valleys (white) for data 1

19 test ridges and 19 reference ridges in the image.
The diagnose is list as flow:

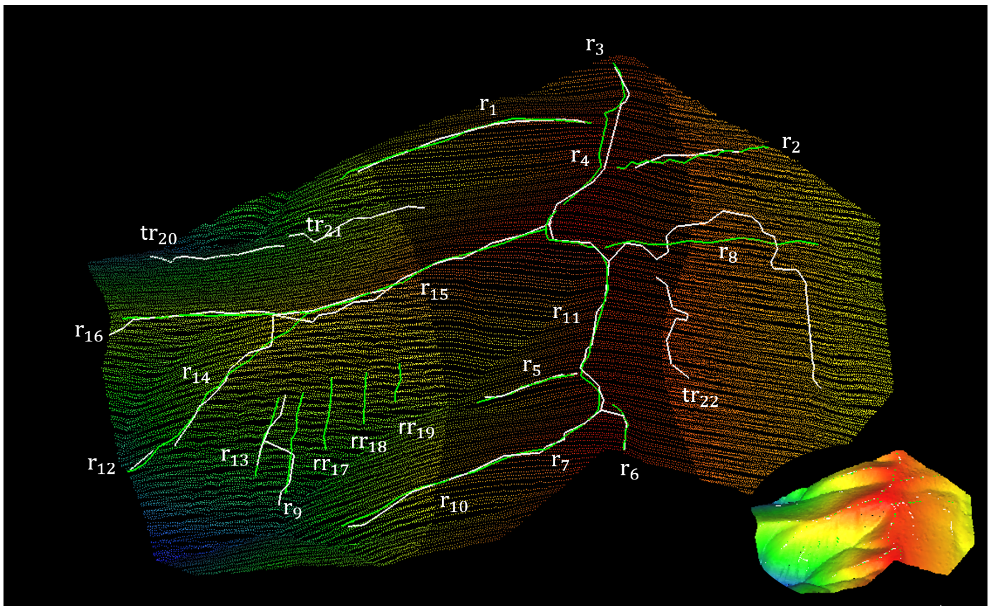


Figure 5.15: The reference ridges (green) and the test ridges (white) for data 1

1. **TP valleys**
   In the image, v_i $(i = 1, 2, 3, ..., 11)$ are eleven pairs of test valley and reference valley, and r_i $(i = 1, 2, 3, ..., 16)$ are 16 pairs of test ridge and reference ridge. They are true positive breaklines.

2. **TN valleys**
   Reference valley 12 (rv_{12}) and reference valley 18 (rv_{18}) are two true negative breaklines in valley. The missing valley happens in the area that the ground surface is more plane (also see Figure 4.5).
   Reference ridge 17, 18 and 19 (rr_{17}, rr_{18}, and rr_{19}) are true negative breaklines in ridge. It happens for the small ridges only.

3. **FN valleys**
   Test valley 19 (tv_{19}), test ridge 20, 21, 22 (tr_{20}, tr_{21}, and tr_{22}) are "false alarm". rv_{13}, rv_{14}, rv_{15}, rv_{16} and rv_{17} are five small valleys and they are merged as one test valley (tv_{20}).
   The cause of those false negative breaklines is explained earlier in Section 5.2.1 and Figure 5.5.

- **PART 2**: how is the accuracy and precision of the "true positive breakline"

The 11 pairs of true positive valley and 16 pairs of true positive ridge are taken into further analysis and the raw data is in Table 5.2 and Table 5.3, in addition, the validation result is in Table 5.4 and Table 5.5.

In the accuracy validation, the parameters are the larger the better accuracy, and in the precision validation, the parameters are the smaller the better.
In general, the average correctness is 97% and 96%, the average quality is 90.67% and 91.19%, and the average completeness is 93.84% and 93.92% for valley and ridge. Most of accuracy parameters are above 80%. The accuracy is high.
As to the precision, the average polyline precision is 0.3 and 0.23, and the average vertex precision is 1.14 meter and 1.12 meter. Comparing with the point density $2pts/m^2$ (see Table 4.2). 25 of 27 breaklines' polyline precision is better than 0.5. 16 of 27 breakline's vertex precision is better than 1.0; and 23 of 26 is better than 1.5. The precision is also good.

TP pair	TP	FN	FP	NE	NR	length of r_i	$\sum d(v, r_i)$
v_1	4	1	0	5	6	17.09	17.09
v_2	25	0	0	25	33	125.86	31.91
v_3	3	0	1	3	10	19.88	3.47
v_4	8	0	0	8	11	35.16	12.67
v_5	21	1	1	22	30	107.49	19.78
v_6	19	0	0	19	13	39.94	41.19
v_7	4	0	0	4	9	19.33	2.47
v_8	11	2	0	13	17	40.57	8.55
v_9	3	0	5	3	17	35.99	2.07
v_{10}	8	1	0	9	9	27.45	13.44
v_{11}	5	1	0	6	9	25.19	1.26

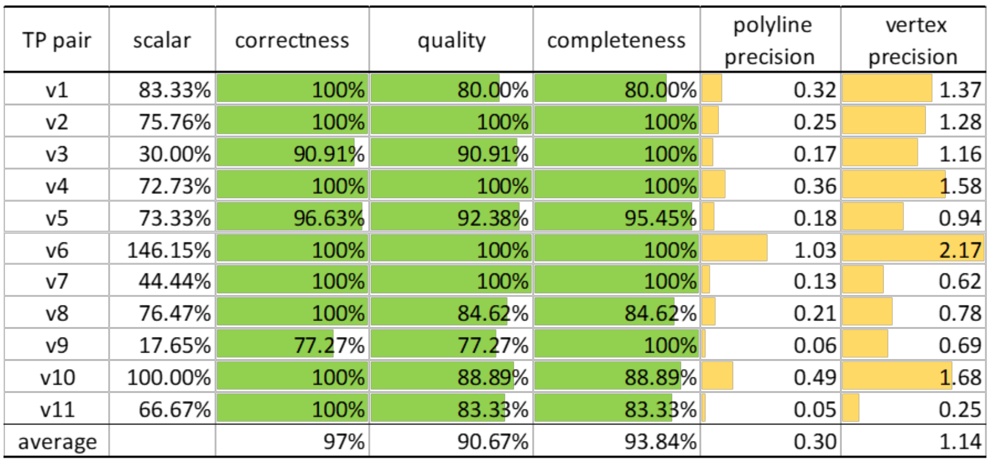
*r_i is the corresponding reference polyline

Table 5.2: Raw data for valley validation

TP pair	TP	FN	FP	NE	NR	length of r_i	$\sum d(v, r_i)$
r_1	10	0	0	10	10	72.39	4.61
r_2	5	0	3	5	23	49.46	4.69
r_3	6	0	0	6	7	15.91	4.20
r_4	4	0	0	4	14	38.27	5.75
r_5	6	0	0	6	11	30.12	2.96
r_6	4	0	0	4	9	16.38	4.35
r_7	3	0	0	3	9	11.90	2.33
r_8	17	6	1	23	12	63.64	78.47
r_9	5	1	3	6	9	28.29	7.42
r_{10}	11	0	0	11	21	70.49	9.31
r_{11}	16	0	0	16	25	77.18	13.70
r_{12}	2	0	0	2	5	11.97	0.83
r_{13}	5	0	0	5	6	22.51	4.20
r_{14}	11	1	1	12	15	55.17	10.92
r_{15}	8	5	0	13	13	48.79	8.83
r_{16}	12	1	0	13	17	70.90	11.14

*r_i is the corresponding reference polyline

Table 5.3: Raw data for ridge validation



TP pair	scalar	correctness	quality	completeness	polyline precision	vertex precision
v1	83.33%	100%	80.00%	80.00%	0.32	1.37
v2	75.76%	100%	100%	100%	0.25	1.28
v3	30.00%	90.91%	90.91%	100%	0.17	1.16
v4	72.73%	100%	100%	100%	0.36	1.58
v5	73.33%	96.63%	92.38%	95.45%	0.18	0.94
v6	146.15%	100%	100%	100%	1.03	2.17
v7	44.44%	100%	100%	100%	0.13	0.62
v8	76.47%	100%	84.62%	84.62%	0.21	0.78
v9	17.65%	77.27%	77.27%	100%	0.06	0.69
v10	100.00%	100%	88.89%	88.89%	0.49	1.68
v11	66.67%	100%	83.33%	83.33%	0.05	0.25
average		97%	90.67%	93.84%	0.30	1.14

Table 5.4: Valley validation result

TP pair	scalar	correctness	quality	completeness	polyline precision	vertex precision
r1	100%	100.00%	100.00%	100.00%	0.06	0.46
r2	21.74%	88.46%	88.46%	100.00%	0.09	0.94
r3	85.71%	100.00%	100.00%	100.00%	0.26	0.70
r4	28.57%	100.00%	100.00%	100.00%	0.15	1.44
r5	54.55%	100.00%	100.00%	100.00%	0.10	0.49
r6	44.44%	100.00%	100.00%	100.00%	0.27	1.09
r7	33.33%	100.00%	100.00%	100.00%	0.20	0.78
r8	191.67%	89.87%	68.23%	73.91%	1.23	4.62
r9	66.67%	71.43%	62.50%	83.33%	0.26	1.48
r10	52.38%	100.00%	100.00%	100.00%	0.13	0.85
r11	64.00%	100.00%	100.00%	100.00%	0.18	0.86
r12	40.00%	100.00%	100.00%	100.00%	0.07	0.42
r13	83.33%	100.00%	100.00%	100.00%	0.19	0.84
r14	80.00%	93.22%	85.94%	91.67%	0.20	0.99
r15	100%	100.00%	61.54%	61.54%	0.18	1.10
r16	76.47%	100.00%	92.31%	92.31%	0.16	0.93
average		96%	91.19%	93.92%	0.23	1.12

Table 5.5: Ridge validation result

In particular, r_8 is the worst breakline among all the ridges and valleys. From the image, the generated ridge 8 is far from the reference ridge.
The correctness and quality of v_9 is the worst in valley, both are 77.27%. Checking Figure 5.14, the test valley 9 missing a part of polyline, and it result in a larger number of FP vertexes. This also proved in Table 5.2. v_9 has the largest number of False Positive value (5) in the whole test data.
v_4, v_6 and v_7 achieve 100% for all three accuracy parameters. Checking in the image, they are correct and complete.
While for v_4, the whole polyline of test valley 4 is shifted a little from the reference data. It is shown in the vertex precision value (1.58), which is the third large value among all the test data.
Same as v_6, it has the best accuracy but the worst precision (1.03 and 2.17). In the zoom-in image (Figure 5.16), the extracted valley is a little bit far from the reference data. v_{11} provides the best precision in both polyline and vertex. In Figure 5.14, the

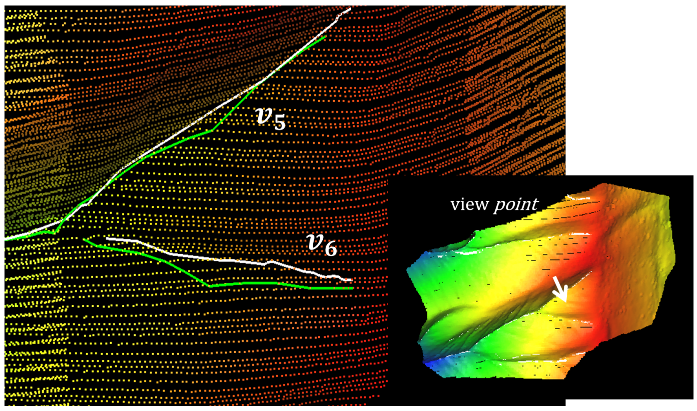


Figure 5.16: The reference valley 6 (green) and the test valley 6 (white)

true positive part of extracted valley 11 is all most overlap with the reference data.

5.3.3 Summary

The validation shows that most breaklines generated with this project is reliable (11 out of 13 valleys and 16 out of 19 ridges are TP). The reliable breaklines are high accurate (average more than 90%) and precise.
In addition, the accuracy and precision are not necessary and sufficient conditions to each other. A breakline with high accuracy might be bad in precision. While a low accuracy is likely to a poor precision.
This project implements the simplest spatial interpolation "Nearest Neighbour". A better spatial interpolation, such as TIN Linear Interpolation or Natural Neighbour Interpolation, will improve the precision.
Observing the TN and FN breaklines and bad accuracy and precision breaklines (r_8), they appear in the plane surface or happen for small breaklines or caused by the MAT interior exterior error (explained in Section 5.2.1).
To avoid the error in small breakline, the user can set a larger c_n to remove them. But a larger c_n takes the risk of removing TP breaklines. For example, to remove tv_{20} in Figure 5.14, the valleys smaller then tv_{20} might also be removed, such as v_3 and v_1. This is a trade of during the breakline generation process.
A solution is selecting a subset point cloud in the small breaklines area. Than generating breaklines for the subset. In this way, the segmentation method and threshold can be more suitable for this area. The out put breaklines needs to merge with the big dataset.
A better idea is estimate parameters for the input dataset, and might involved supervision classification or machine learning. It will help the whole process becoming fully automatic which is more convenient for the user.

6 CONCLUSIONS AND FUTURE WORK

Section 6.1 answer the main research question and summarises the main result of this project. Section 6.2 gives the answer to the sub research questions. Furthermore, Section 6.3 makes a conclusion of the limitation and trade of for the breakline generating process. Finally, Section 6.4 lists the promising ideas for the future work.

6.1 RESEARCH OVERVIEW

This project aims to generate breakline for mountains from the point cloud directly with the MAT.
The breakline is a structured line of the object surface which has high curvature. Meanwhile, the reciprocal of the medial edge ball's radius can represent the point with high curvature, which is the fundamental idea of this project.
To solve the main research question, 7 steps are taken to get 3D breaklines from the point cloud:

1. Normal estimation for the point cloud;
   The normal vector of a point is a required input for the MAT ball shrinking algorithm.

2. MAT processing (preparing for step 3 and 5);
   In the second step, the plane area of the terrain is separated from the point cloud by MAT. This step proposes a new geometry and a new method for medial sheet segmentation. The adjacent relationship between medial sheets are separated into deep and surface adjacent.

3. Candidate point extract;
   In the third step, since a small radius of a medial ball corresponds to a high curvature, those medial balls are selected to calculate candidate point coordinates.

4. Polyline generation from candidate points;
   The fourth step provides two options to generate polylines.
   One option utilizes graph theory. It first connects the candidate points to its closet neighbour with the minimum spanning tree (MST) algorithm. Then it simplifies the MST into a polyline with the shortest path algorithm.
   The other option utilizes polynomial fitting. It fits a cubic polynomial function through the candidate points as a polyline.

5. The topology of breaklines;
   The fifth step estimates the junction point for adjacent breaklines with the MAT adjacent relationship between medial sheets.

6. Polyline simplification and smoothing.
   In the sixth step, the polyline can be simplified and smoothed. The simplification eliminates less important point in the polyline. For the smoothing four methods are compared to insert new point in the polyline.

7. Save polylines to a file.
   In the end, the result can be saved into a .obj file to share and open in other software.

This method is able to extract breaklines for mountain regions. When the experiment data contains a larger area, a higher point density, a bigger height difference and less smooth surface, a set of smaller threshold is possible to use, and the generated breaklines contains more details (small breaklines).
Compared with the manually drawn breaklines. this project generate more reliable ridges around mountain peak. For test data 1, this project performs more sensitive when generating ridge. For one hand the method is able to find inconspicuous true positive ridges, which is easy to be neglected by the operator. This is the advantage of being sensitive. While in another hand, it also result in "false alarm", more specifically, generating false negative ridges. This is the disadvantage, also the "trade of" for the advantage aspect. For valleys, this method does not create false negative breaklines, but lead to false positive breaklines.
Compared with ArcMap which detects as breakline cells from a raster, this project generates 3D polylines which support further spatial topological analyses. The generated 3D polylines are more complete (or longer) the the ArcMap.
Compared with CGAL mesh method, this project is less limited by the input data. CGAL requires a smooth mesh input or the out put breaklines containing massive unexpected polylines. While this project is able to extract reliable breaklines with the same point cloud that generates mesh for CGAL.
According to the validation of data 1, that most breaklines generated with this project is reliable. The reliable breaklines are high accurate (average more than 90%) and precise. In addition, the accuracy and precision are not necessary and sufficient conditions to each other. A breakline with high accuracy might be bad in precision. While a low accuracy is likely to a poor precision.
Observing the TN and FN breaklines, they appear in the plane surface or happen for small breaklines or caused by the MAT interior exterior error (explained in Chapter 5).

6.2 RESEARCH QUESTIONS

In this section, the sub research questions proposed in Section 1.2 will be answered:

- Q: What is the link between Medial Axis Transform (MAT) and a breakline?
  A: Explained in Section 3.1, there are three link between them:
  1. The breakline's definition is 3D polylines with maximum local curvature in terrain surface. With the MAT method, the edge ball's reciprocal value of radius r ($\frac{1}{r}$) presents the curvature. So 3D points with maximum curvature can be located by this property, and those points are the vertexes in breaklines.
  2. Furthermore, the interior MAT presents the ridge and the exterior MAT presents the valley.
  3. One medial sheet indicates one breakline.
- Q: If MAT is suitable for generating a 3D breakline?
  A: The MAT is able to provide useful medial geometries to generate breaklines. The method this project proposed can successfully identify breaklines when terrain's height difference is large or in less planar area. This method is sensitive which is good at finding hidden breaklines neglected by operator, but also has false extraction or missing breaklines when the terrain surface is planar or the breakline is rather small. Furthermore the "exterior and interior error" will cause repeating (or overlapping) breaklines.

- Q: What are the requirements for the point cloud to extract breaklines?
  A: As Section 1.2 said, MAT can not find the structure of a point cloud contain trees in a small area. So the point cloud requires to be clean and do not contain trees. Or the trees needs to be filtered with ground filters before implementation.
  Also if the point cloud is incomplete, for example, having holes caused by lake or raver, this method can not extract breaklines successfully.

- Q:Is it possible to automatically generate a breakline without user defined parameters?
  A: For now this project is not fully automatic, and the segmentation process needs the user operation. The segmentation result is easy to visualise and compare in geoflow, so the proper parameters is not hard to find.

6.3 DISCUSSION

The main issue of this method is the trade off between detail (small breaklines) and accuracy. There are three major trades off in this project, and they are embodied in the user defined thresholds.

1. The resulting breakline is highly depends on the segmentation result. As Section 4.4 explained, a ill segmentation will cause many problems: two close or overlapping breaklines should be one, missing/incomplete breaklines or unexpected breaklines.
   The segmentation result is depends on the segmentation method and thresholds. They are hard values and used for the whole dataset. Some areas (such as the small breaklines in data 1, see Figure 5.4) require more specific thresholds. To take care of the entirety dataset, the local details (more specifically, the small medial sheets), are sacrificed. This is the first trade of in the method.

2. The sacrificed small medial sheets will cause errors in the resulting breaklines. Those error can be eliminated by a larger c_n. But a larger c_n takes the risk of removing True Positive breaklines. This is the second trade off in the method.

3. the polyline simplification and smoothing are capable to remove the "shift" of original breakline (see Section 5.1). But it is possible to lose accuracy at the same time. This is the third trade of.

6.4 FUTURE WORK

The solution for the main issue is automatic segmentation, then estimating the other thresholds for different sheets.
A idea for the automatic segmentation is implementing Support Vector Machine (SVM) which is a unsupervised classification method. One medial geometry is one feature vector in the SVM. The automatic segmentation should take care of the entirety dataset and the local details (the small breaklines) both.
For the thresholds estimation, the point cloud meta data, such point density and resolution, is necessary.
Besides the main issue, there are several ideas might improve the result:

1. the topology of the breaklines:
   (a) The junction points need better estimating to remove the tail and rough angle (see Section 4.7).
   (b) After removing the "tail", two breaklines are connected at "end-to-end" should be merged as one polyline.

2. Generating polyline with medial direction (See Section 4.6.2)
   The medial direction $\vec{n}_c$ of an edge ball can be approximately seen as the directions of maximal principal curvatures of the point ($\vec{v}$) on the breakline. $\vec{n}_c$ points to the lower direction of a breakline or the upper direction of the breakline. This property might help to generate a more smooth and reliable breakline.

3. Polyline smoothing by cubic spline method (see Section 3.8.2).

4. A better spatial interpolation will help for a better precision.
   This project implements the simplest spatial interpolation ”Nearest Neighbour”. A better spatial interpolation, such as TIN Linear Interpolation or Natural Neighbour Interpolation, will improve the precision.

In addition, the ”exterior and interior error” (see Section 5.1) in the MAT also need to take care of.

BIBLIOGRAPHY

Band, L. and Robinson, V. (1992). Intelligent land information system final report. *Toronoto University*.

Band, L. E. (1986). Topographic partition of watersheds with digital elevation models. *Water resources research*, 22(1):15–24.

Baruch, A. and Filin, S. (2008). Detection of subtle ridgelines from laser scanning data. *International Archives of Photogrammetry and Remote Sensing*, 37(Pt. B3A):111–118.

Blum, H. (1967). A transformation for extracting new descriptors of shape. *Models for Perception of Speech and Visual Forms, 1967*, pages 362–380.

Bridge, J. S. and Leeder, M. R. (1979). A simulation model of alluvial stratigraphy. *Sedimentology*, 26(5):617–644.

Briese, C. (2004). *Three-dimensional modelling of breaklines from airborne laser scanner data*. na.

Bryan, W. and Moore, J. G. (1977). Compositional variations of young basalts in the mid-atlantic ridge rift valley near lat 36 49′ n. *Geological Society of America Bulletin*, 88(4):556–570.

Brzank, A., Lohmann, P., and Heipke, C. (2005). Automated extraction of pair wise structure lines using airborne laserscanner data in coastal areas. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(3/W19):3641.

Cazals, F. and Pouget, M. (2005). *Topology driven algorithms for ridge extraction on meshes*. PhD thesis, INRIA.

Chang, Y.-C., Song, G.-S., and Hsu, S.-K. (1998). Automatic extraction of ridge and valley axes using the profile recognition and polygon-breaking algorithm. *Computers & Geosciences*, 24(1):83–93.

Chorowicz, J., Ichoku, C., Riazanoff, S., Kim, Y.-J., and Cervelle, B. (1992). A combined algorithm for automated drainage network extraction. *Water Resources Research*, 28(5):1293–1302.

Ebner, H., Reinhardt, W., and Hößler, R. (1988). Generation, management and utilization of high fidelity digital terrain models. *International Archives of Photogrammetry and Remote Sensing*, 27(B11):556–566.

Greysukh, V. (1967). The possibility of studying landforms by means of digital computers. *Soviet Geography*, 8(3):137–149.

Jenson, S. K. and Domingue, J. O. (1988). Extracting topographic structure from digital elevation data for geographic information system analysis. *Photogrammetric engineering and remote sensing*, 54(11):1593–1600.

Linde, Y., Buzo, A., and Gray, R. (1980). An algorithm for vector quantizer design. *IEEE Transactions on communications*, 28(1):84–95.

Martz, L. W. and Garbrecht, J. (1992). Numerical definition of drainage network and subcatchment areas from digital elevation models. *Computers & Geosciences*, 18(6):747–761.

O'Callaghan, J. F. and Mark, D. M. (1984). The extraction of drainage networks from digital elevation data. *Computer vision, graphics, and image processing*, 28(3):323–344.

Pang, X., Song, Z., and Xie, W. (2012). Extracting valley-ridge lines from point-cloud-based 3d fingerprint models. *IEEE computer graphics and applications*, 33(4):73–81.

Peters, R. (2018). Geographical point cloud modelling with the 3d medial axis transform.

Peucker, T. K. and Douglas, D. H. (1975). Detection of surface-specific points by local parallel processing of discrete terrain elevation data. *Computer Graphics and image processing*, 4(4):375–387.

Rutzinger, M., Höfle, B., Pfeifer, N., Geist, T., and Stötter, J. (2006). Object-based analysis of airborne laser scanning data for natural hazard purposes using open source components. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(4/C42).

Schwalbe, E., Maas, H.-G., and Seidel, F. (2005). 3d building model generation from airborne laser scanner data using 2d gis data and orthogonal point cloud projections. *Proceedings of ISPRS WG III/3, III/4*, 3:12–14.

Ugelmann, R. (2000). Automatic breakline detection from airborne laser range data.

Verma, V., Kumar, R., and Hsu, S. (2006). 3d building detection and modeling from aerial lidar data. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2213–2220. IEEE.

Vosselman, G., Dijkman, S., et al. (2001). 3d building model reconstruction from point clouds and ground plans. *International archives of photogrammetry remote sensing and spatial information sciences*, 34(3/W4):37–44.

Wang, J. and Shan, J. (2009). Segmentation of lidar point clouds for building extraction. In *American Society for Photogramm. Remote Sens. Annual Conference, Baltimore, MD*, pages 9–13.

Zhu, A.-X., Hudson, B., Burt, J., Lubich, K., and Simonson, D. (2001). Soil mapping using gis, expert knowledge, and fuzzy logic. *Soil Science Society of America Journal*, 65(5):1463–1472.

A | DIAGRAMS

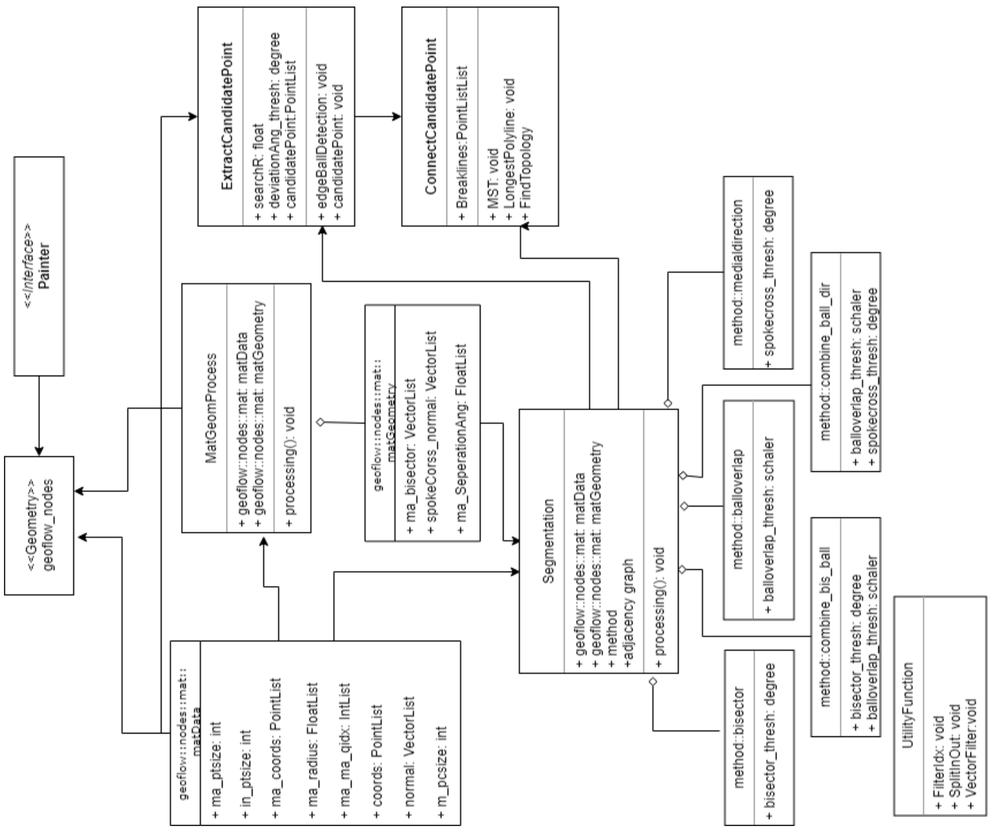


Figure A.1: The UML diagram of software architecture

COLOPHON

This document was typeset using LaTeX. The document layout was generated using the `arsclassica` package by Lorenzo Pantieri, which is an adaption of the original `classicthesis` package from André Miede.