Geographical point cloud modelling with the 3D medial axis transform

Peters, Ravi

DOI
10.4233/uuid:f3a5f5af-ea54-40ba-8702-e193a087f243

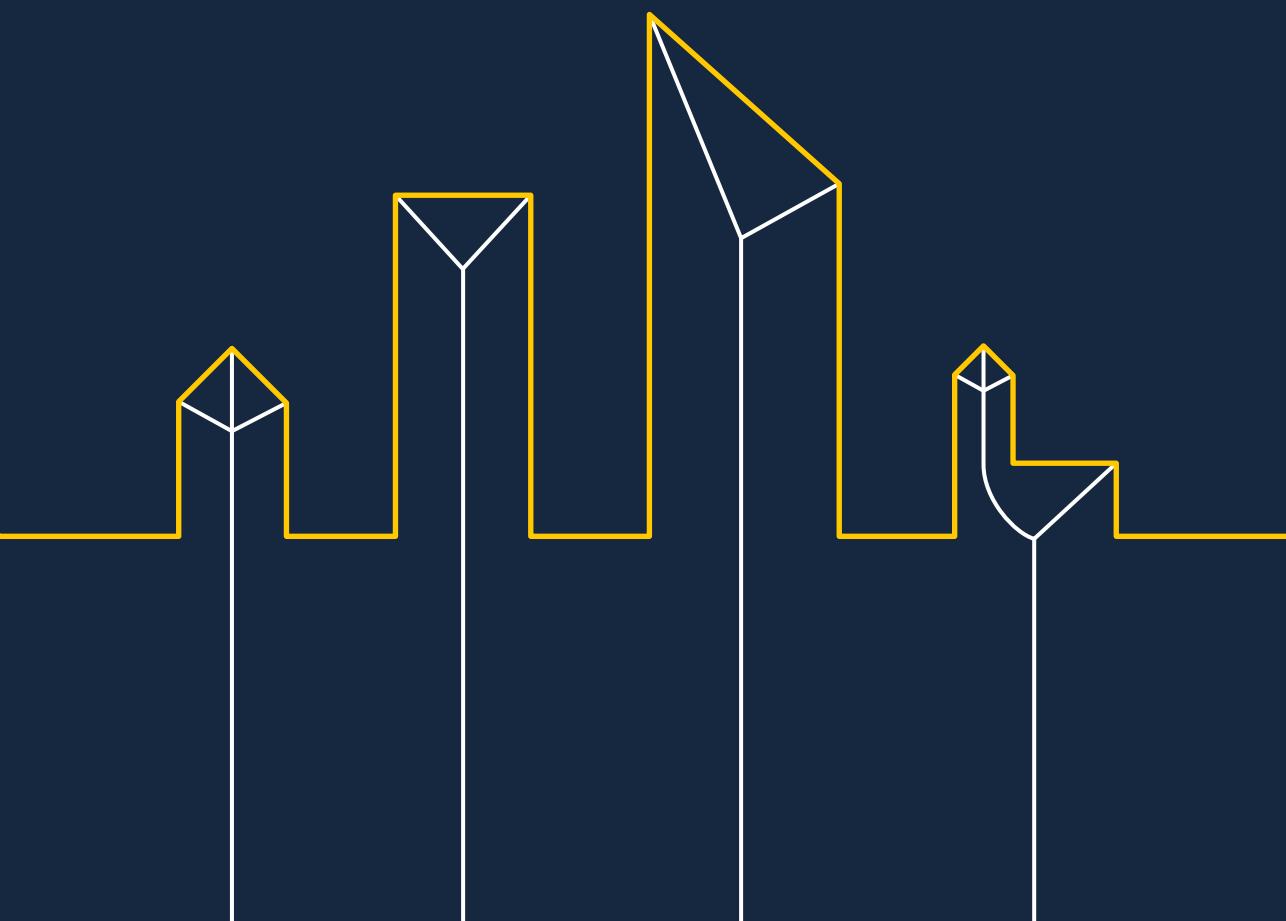
Publication date
2018

Document Version
Final published version

Citation (APA)
Peters, R. (2018). *Geographical point cloud modelling with the 3D medial axis transform*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:f3a5f5af-ea54-40ba-8702-e193a087f243

Geographical point cloud modelling with the 3D medial axis transform

Ravi Peters



Geographical point cloud modelling with the 3D medial axis transform

Ravi Peters

Written in 2017-2018 by Ravi Peters.

ISBN: 978–94–6186–899–2

Printed by Ridderprint BV

A digital version of this thesis is available at http://repository.tudelft.nl.

Geographical point cloud modelling with the 3D medial axis transform

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus prof. dr. ir. T. H. J. J. van der Hagen
chair of the Board of Doctorates
to be defended publicly on
Wednesday 14 March 2018 at 10.00 o'clock

by

Ravi Ylan PETERS

Master of Science in Geomatics, Delft University of Technology, the Netherlands
born in Rotterdam, the Netherlands.

This dissertation has been approved by the promotor and the copromotor.

Composition of the doctoral committee:

Rector Magnificus	chairperson
Prof. dr. J.E. Stoter	Delft University of Technology, promotor
Dr. H. Ledoux	Delft University of Technology, copromotor

Independent members:

Univ.Prof. Dipl.-Ing. Dr.techn. N. Pfeifer	Vienna University of Technology
Prof. dr. F. Remondino	University of Bologna
Prof. dr. ir. arch. I.S. Sariyildiz	Delft University of Technology
Dr. K.A. Hildebrandt	Delft University of Technology
Dr. R.C. Lindenbergh	Delft University of Technology
Prof. dr. ir. A. van Timmeren	Delft University of Technology, reserve member

This work is part of the Open Technology programme with project number 12217 which is financed by the Netherlands Organisation for Scientific Research (NWO).

Contents

Acknowledgments

First and foremost, I would like to express my gratitude to my supervisors Hugo Ledoux and Jantien Stoter. I always enjoyed working with you and I am very pleased that I can continue to do so in the foreseeable future. Thank you for creating such a wonderful working environment.

I would like to thank the Netherlands Organisation for Scientific Research (NWO) for funding my PhD project, the NWO program director Margriet Jansz, and the members of the NWO user committee for their participation and valued input: Bentley Systems, City of Rotterdam, Ministry of Infrastructure and Water Management, Safe Software, Sweco, and The Waterschapshuis. A special mention goes out to Kevin Wiebe and Jeff Coukell from Safe Software for integrating my point cloud simplification algorithm into their FME software, and for contributing their improvements on my code back to me.

I am grateful to Fabio Remondino and the members of his 3D Opitical Metrology group at the Fondazione Bruno Kessler for hosting me and letting me experiment with their photogrammetric point clouds.

I thank Roger de Crook of Hoogheemraadschap de Stichtse Rijnlanden, for providing the watercourse detection use case (see Section 7.1).

I also thank the people at the GIS Technology group, where I started my PhD: Elfriede Fendel, Tjeu Lemmens, Martijn Meijers, Peter van Oosterom, Wilko Quak, Radan Šuba, Theo Tijssen, Edward Verbree, and Marian de Vries.

Special thanks to everyone who is or used to be part of the 3D geoinformation group: Abdoulaye Diakité, Anna Labetski (she edited the introduction of this thesis), Balázs Dukai, Filip Biljecki Jinjin Yan, Kavisha, Ken Arroyo Ohori, Liangliang Nan, Liu Liu, Sangmin Kim, Stelios Vitalis, Tom Commandeur, Weixiao Gao, and Zhiyong Wang. I had so much (organised) fun and I thoroughly enjoy being be part of the group.

I am also grateful to Margo van der Helm and Daniëlle Karakuza for their administrative and organisational support.

Last but not least, I would like to express my gratitude to my parents Levity and Saskia, my brothers, my grandfather and the rest of my family for their continued love and support during my PhD research.

Ravi Peters *Zoetermeer, February 2018*

1 Introduction

Point clouds speak to the imagination. Today one can capture point clouds of vast geographic regions with relative ease. It is a fully three-dimensional, often coloured, and high resolution representation of the natural and built environment, where even small details such as benches and lamp posts are visible (see Figure 1.1). The beautiful thing about such point clouds is that—provided a capable and well executed immersive visualisation system[1]—they enable us to explore any viewpoint imaginable and study a 3D scene. There are no physical restrictions on how you can move around nor in the relative scale of your surroundings to yourself. This freedom facilitates the ability to observe things in ways that are normally inaccessible to us, thereby enabling us to learn new things about the environment with increased ease and efficiency.

Geographical point clouds, i.e. point clouds of geographic regions typically acquired though aerial measurements using aerial LiDAR systems [Mallet and Bretar, 2009], have numerous applications in asset management, crisis management, city and landscape planning, and environmental simulations [Axelsson, 1999; Snyder, 2013]. While these applications cover a broad range of different expertises, they often require some of the same basic elements from a point cloud. One such element is the ability to distinguish between points that represent water, terrain, vegetation, roads or individual buildings. In other words: they require *semantics*. In fact, nowadays point clouds are often used to construct a 3D city model, a semantically rich collection of surface models of objects in urban areas, i.e. where each urban object (e.g. a building or a tree) is represented as a separate entity with a well defined meaning [Gröger and Plümer, 2012]. In practice there is often a preference to utilise a 3D city model, rather than a point cloud directly. This should not come as a surprise, given that information in a 3D city model is well-structured, compact, and explicitly labelled. We can not say the same of point clouds and certainly not of raw and unprocessed point clouds.

This brings us to the question: How do we obtain a semantically rich 3D city model from a raw point cloud? Or, more specifically, how do we make a computer do that? If we—humans—look at a point cloud, we can understand what we see. We recognise man-made and natural structures in the terrain, such as buildings

[1] There are specialised systems capable of fluently rendering massive (billions of points) point cloud datasets in a virtual or augmented reality environment (see for example Kreylos et al. [2008]).

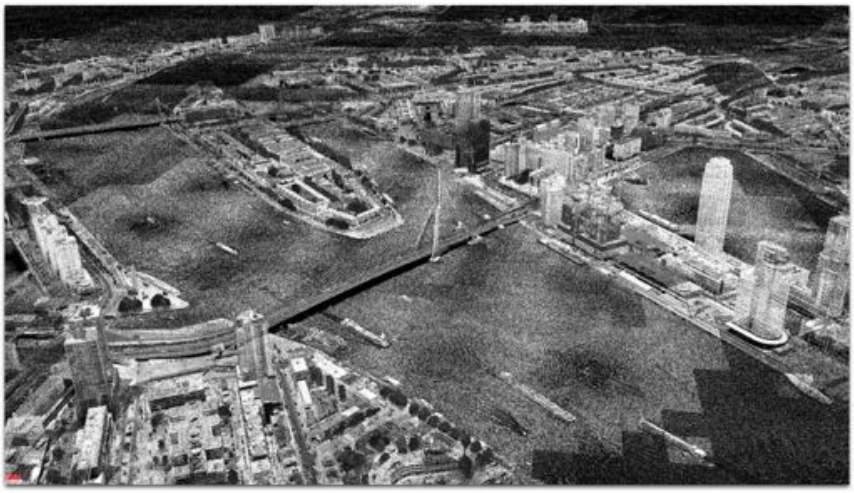
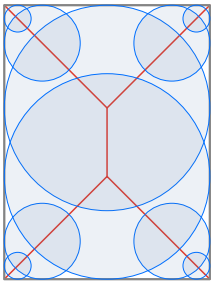


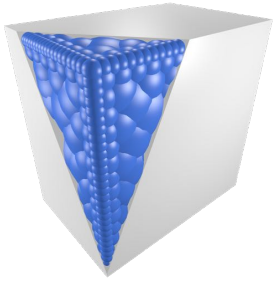
Figure 1.1: An aerial LiDAR point cloud (Hoogtebestand Rotterdam 2012).

and mountains, even though the groups of points that constitute these structures are not explicitly or even implicitly labelled as such in the data structure in our computer. This is because we have a visual cortex; the part of our brain through which we make sense of visual information. However, to a computer, a point cloud is nothing more than a list of points embedded in 3D space, without any form of organisation or semantics [Schnabel et al., 2008]. So how can we make the computer recognise some of the same structures that we see when we look at a point cloud? With this thesis I hope to contribute to an answer to this question, by developing a method to automatically create a so-called semantic point cloud [Virtanen et al., 2017].

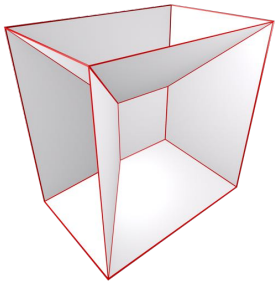
I propose a new approach to point cloud modelling based on the 3D Medial Axis Transform (MAT) [Blum, 1967], an alternative way to describe shape (See Figure 1.2). This approach is fully 3D, in contrast to most existing methods for geographical point cloud modelling that work in 2.5D. Tagliasacchi et al. [2016] say the MAT to be *dual* to the conventional boundary (e.g. a surface point cloud or mesh) and volumetric (e.g. voxel) representation, because the MAT contains the same information, but models key properties of a shape in a more intuitive and explicit way. The key benefit of the MAT here is its skeleton-like representation that explicitly encodes the structure of a shape through a set of connected *medial sheets*. At the same time it also encodes the full geometry of a shape as the union of its medial balls that effectively describes the volume of the shape.



(a) The MAT for a rectangular shape consists of medial balls (blue) and the medial axis (red).



(b) 3D medial balls for a box shape.



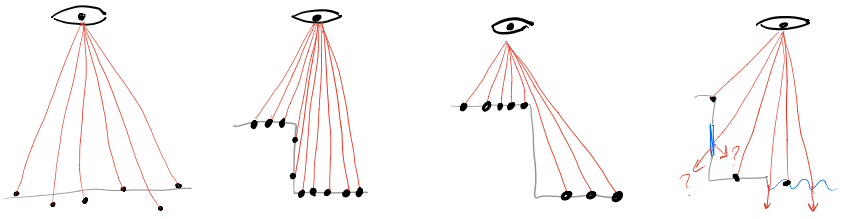
(c) 3D medial axis of a box shape.

Figure 1.2: The MAT in 2D and in 3D.

This combination makes it a powerful shape descriptor, especially for point clouds that lack any form of organisation. For example, in this thesis I demonstrate that a point cloud can be decomposed in meaningful parts such as separate buildings, by using a decomposition of the MAT (Chapter 3). The MAT brings us a new perspective in point cloud modelling and the underlying hypothesis of this thesis is that certain operations on point clouds can be achieved more effectively by using its MAT, rather than its surface points directly.

The novelty of this thesis lies in the application of the 3D MAT to geographical point clouds, which to my knowledge has not been done before. My core contribution is three-fold. First, geographical point clouds are noisy and the MAT is notorious for being extremely sensitive to noise, rendering the resulting MAT—when constructed with conventional methods—is practically useless. In Chapter 3 however, I demonstrate how to adapt the ball-shrinking algorithm from Ma et al. [2012] to be robust to noise. This enables us to approximate the geometric part of the MAT for geographical point clouds in a way that is scalable to very large point cloud datasets as described in Chapter 5. Second, in Chapter 4 I show how to organise the MAT into a connected set of medial sheets that form so-called *medial clusters* that gives us a natural decomposition of the point cloud into objects. Third, I show how the MAT can be applied for object detection and classification, point cloud simplification and visualisation, and visibility analysis in geographical point clouds (Chapters 6 and 7). Furthermore, despite my focus on geographical point clouds in this thesis, some of my results, such as the segmentation approach presented in Section 4.2, should also be applicable in a more general context.

In this thesis I conclude that the MAT offers a novel perspective to geographical



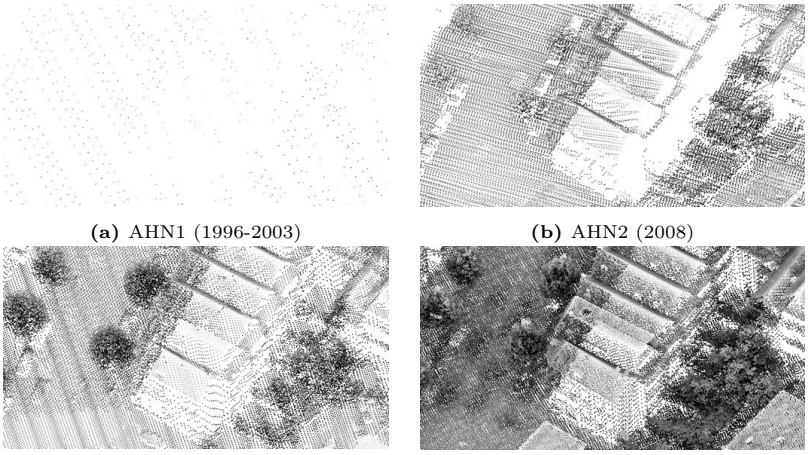
(a) Measurement noise **(b)** Uneven point distribution **(c)** Missing surfaces **(d)** Absorbance and reflectance

Figure 1.3: Increasing point cloud density over the years in different versions of the national Dutch elevation dataset (AHN) and the point cloud dataset maintained by the City of Rotterdam.

point cloud modelling. In particular, the MAT gives us 1) an elegant way to structure and decompose a point cloud into meaningful objects such as buildings and watercourses in a geographical scene, 2) an explicit separation into interior and exterior volumes in the point cloud, and 3) the local medial geometry: a set of measures to characterise shapes. Consequently, I believe that applications such as object detection can greatly benefit from these unique properties of the MAT. Thereby bringing the computer that much closer to understanding some of the same things that we do when we look at a geographical point cloud.

1.1 Related work in geographical point cloud modelling

The geographical point clouds that I consider in this thesis are acquired using airborne LiDAR-based systems [Shan and Toth, 2008]. Due to practical and physical constraints such point clouds suffer from a number of flaws that generally makes it difficult to process them. In Figure 1.3 I have illustrated the main problems when working with geographical point clouds. First, there is measurement uncertainty (a combination of positioning and ranging errors); points will not lay exactly on the sampled surface (Figure 1.3a). Second, not all parts of the surface get equal exposure during acquisition. As a result the point distribution will vary significantly (Figure 1.3b). In the extreme case a part of the surface is not seen at all during acquisition and will be completely missing in the resulting DSM (Figure 1.3c). And even if a surface gets adequate exposure, the physical properties of the surface material can make it invisible to the acquisition technique, e.g. for water or glass surfaces (Figure 1.3d). These characteristics set geographical point clouds apart from the typically much denser and higher quality point clouds that are used in the more general field of geometric modelling, e.g. for 3D surface reconstruction as in Amenta et al. [2001], Alexa et al. [2001], Kolluri et al. [2004], Dey and Goswami [2003] or Kazhdan et al. [2006].



(a) AHN1 (1996-2003) **(b)** AHN2 (2008)

(c) AHN3 (2014) **(d)** City of Rotterdam (2016)

Figure 1.4: Increasing point cloud density over the years in different versions of the national Dutch elevation dataset (AHN) and the point cloud dataset maintained by the City of Rotterdam.

I use the term geographical point cloud modelling to describe all methods that are designed to semantically or structurally enrich geographical point clouds to facilitate the extraction of useful information. For instance, the automatic detection, recognition and reconstruction of urban objects such as buildings, for the purpose of creating 3D city models, have proven to be popular topics among researchers for the last two decades (see e.g. Haala and Kada [2010]; Musialski et al. [2013]; Rottensteiner et al. [2014]). Building reconstruction methods are particularly interesting, not only because there are many applications that need them [Biljecki et al., 2015], but also because they are quite demanding in the sense that they require an extensive range of point cloud modelling techniques to be able to reconstruct the required structured geometry from a geographical point cloud. However, despite the large body of research in building modelling and the increasing point density of geographical point clouds (see Figure 1.4d), it remains challenging to set up a fully automated workflow to derive accurate and geometrically valid building models from geographical point clouds [Rottensteiner et al., 2014].

Following is a discussion of relevant literature in parts, where each part corresponds to a processing phase intended to gradually increase the level of semantics and structure in a geographical point cloud.

Point classification

One of the most elementary forms of semantics in point cloud is a point classification, i.e. assigning a each point a label that signifies the type of object it belongs to. The earliest developed methods classify points into a ground and a non-ground class for the generation of DTMs, e.g. using morphological filters [Kilian et al., 1996; Vosselman, 2000; Zhang et al., 2003], progressive TIN densification [Axelsson, 1999], or by fitting interpolation surfaces [Kraus and Pfeifer, 1998, 2001]. Other classification methods also detect e.g. buildings, vegetation and clutter classes based on point attributes, return count, or characteristics of the neighbourhood of a point, such as planarity, slope, elevation, local height difference, and curvature. These methods are using techniques such as grid-based region growing (e.g. Forlani et al. [2006]), and graph-cut based optimisation (e.g. Golovinskiy and Funkhouser [2009]; Lafarge and Mallet [2012]). A point cloud classification in itself does not distinguish between different objects of the same type, however it can be used to quickly discard points that are not relevant to a particular application (see e.g. Awrangjeb and Fraser [2014]; Lafarge and Mallet [2012]; Xiong et al. [2014]).

Object detection

When individual objects, e.g. single building structures, need to be detected in a point cloud a variety of techniques can be applied. For instance, one can filter out ground points and use Euclidean clustering [Sun and Salvaggio, 2013] or grid-based clustering [Awrangjeb and Fraser, 2014]. Another approach is to use object footprints from external source, e.g. from a cadastral registry (e.g. Brenner [2000]; Haala and Brenner [1997]; Henn et al. [2013]; Xiong et al. [2016]), or derived from aerial imagery using computer vision techniques (e.g. Sohn and Dowman [2007]). While additional data sources can, if available, compensate for flaws in the point cloud and simplify the development of an approach, adding other data sources may cause problems due to different data accuracies, acquisition dates, or other types of errors in the additional data source.

Geometry extraction

In building modelling, object detection is often followed by object reconstruction, i.e. the computation of yhedral model with detailed roof structures and simplified facades similar to the LoD2 building specification of the CityGML standard [Open Geospatial Consortium, 2012]). Data-driven object reconstruction methods often perform a 2.5D triangulation of the original data points or a simplification thereof (see e.g. Constantin et al. [2010]; Wahl et al. [2008]; Zhou and Neumann [2010]), or the building geometry is derived directly from a

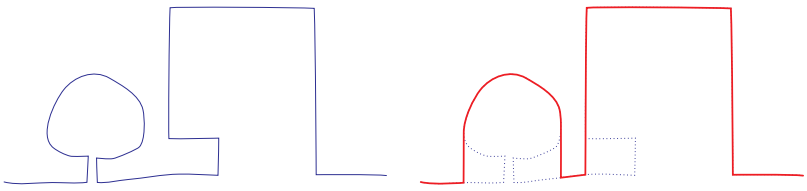
set of best-fitting planes by carefully intersecting them [Dorninger and Nothegger, 2007; Dorninger and Pfeifer, 2008]. Other researchers employ model-driven reconstruction methods, either by directly fitting parametrised models [Henn et al., 2013] or by first performing a segmentation or primitive fitting step for a decomposition into simple geometric entities such as planes and smooth surface patches (e.g. Verma et al. [2006]) or slightly higher level geometric primitives such as cylinders, spheres and cubes (e.g. Lafarge and Mallet [2012]; Schnabel et al. [2008]). This decomposition phase is often applied in building modelling and is based on techniques such as the Random Sampling Consensus (RANSAC, Fischler and Bolles [1981]), see e.g. [Ameri and Fritsch, 2000; Brenner, 2000; Tarsha-Kurdi et al., 2008], the Hough transform (Hough [1962]), see e.g. Overby et al. [2004]; Vosselman et al. [2001], and region-growing segmentation, see e.g. Rottensteiner [2006]; Verma et al. [2006]. The underlying assumption is often that the resulting groups of points, e.g. planar segments or shape primitives, are part of separate building objects.

Extracting topological structures

Finally, a vital, yet challenging, step in model-based building reconstruction is to organise and structure the available segments or primitives in such a way that a regularised, watertight and geometrically valid polyhedron can be derived. One approach is to find and classify topological relations between neighbouring surface segments. The resulting topology is captured in a roof topology graph, i.e. an abstract graph where each node represents a planar segment and each edge represent an adjacency between neighbouring segments (see e.g. Verma et al. [2006] or Elberink and Vosselman [2009]; Forlani et al. [2006]; Schnabel et al. [2008]; Xiong et al. [2014]). The constructed roof topology graph is then matched against a library of parametrised building models and the best fitting model is fitted to the LiDAR points. Other approaches to induce the required structure for building reconstruction are based on e.g. footprint decomposition (see e.g. Kada and McKinley [2009]; Xiong et al. [2016]) or by projecting class labels to a 2D grid and using advanced image-based segmentation and optimisation techniques [Lafarge and Mallet, 2012]. Yet, the fully automatic reconstruction of LOD2 buildings remains challenging [Xiong et al., 2016], especially for the construction of geometrically valid models [Rottensteiner et al., 2014].

1.2 Research questions and scope

Automatic building detection and reconstruction, an advanced topic in geographical point cloud modelling, involves the organisation of points into higher level geometrical entities, e.g. planar facets or other types of simple surfaces and shapes,



(a) A 3D scene. **(b)** Reduction to 2.5D.

Figure 1.5: Information is lost when a 3D model is reduced to 2.5D model.

and finding structural relations between those entities. That so-called *structuration* of a geographical point clouds—often with a poor point distribution—is not easy, is illustrated by the plethora of methods that were developed during the last two decades. Virtually all these methods are based on the boundary representation, and to simplify the structuration problem, many methods reduce the dimensionality of the point cloud, e.g. using field-based representations like a rasterisation or a 2D triangulation [Kumler, 1994; Li et al., 2005]. This considerably simplifies reconstruction, because it is 1) a simple way to clearly separate between the interior and exterior part of an object, and 2) it simplifies neighbourhood relation. However, it also has a fundamental limitation, because state-of-the-art geographical point clouds, with ever increasing point densities [Virtanen et al., 2017], can contain complex 3D objects that can not always be reduced to 2.5D ones without a significant loss of information [Axelsson, 1999; Filin and Pfeifer, 2005], e.g. a building with an overhanging structure such as a balcony as illustrated in Figure 1.5.

In this thesis I explore a novel approach to point cloud modelling based on the 3D Medial Axis Transform The main benefits of the MAT are that

1. it is a truly 3D structure with a well defined interior and exterior, and
2. it gives a natural decomposition of a geographical scene into meaningful objects such as buildings and watercourses.

As such I believe that the 3D MAT would be a powerful and complementary addition to the existing methods in geographic point cloud modelling that are based on the boundary representation.

My main research objective is the following:

> Exploring to what extend can the MAT be applied to effectively analyse geographical point clouds.

As far as I know, the application of the 3D MAT to geographical point clouds is completely novel. Consequently, this thesis is of an explorative nature. The main goals are to

1. investigate how the MAT can be made to work for geographical point clouds,
2. to find what characteristics and properties of the MAT are relevant to geographical point cloud modelling, and
3. to demonstrate how these can be used using real-world datasets and applications.

This leads to the following main research questions:

1. Can the MAT be robustly computed or approximated for geographical point clouds?
   a) How to effectively deal with noisy and incomplete point cloud inputs during MAT approximation? (i.e. the *unstructured* MAT)
   b) How to effectively structure the MAT? (i.e. the *structured* MAT)
2. What applications benefit most from MAT-based geographical point cloud modelling?

Scope

1. I focus mainly on geographical point clouds that are obtained using aerial laser scanning.
2. The aim of this thesis is to explore the feasibility of the MAT for geographical point cloud modelling and to identify promising applications. Further, more in-depth developments of particular applications are left for future research.

1.3 Thesis outline

This thesis consists of 9 chapters. Following is a brief overview of the structure of this thesis

- Chapter 2 introduces the reader to the Medial Axis Transform and describes all relevant foundational concepts for the following chapters.
- In Chapter 3 I introduce a method to robustly compute the geometrical part of the MAT for geographical point clouds, i.e. the *unstructured* MAT.
- In Chapter 4 I further develop the MAT for geographical point clouds, by proposing methods for decomposing the MAT into its constituent parts and reconstructing the topology of those parts, i.e. the *structured* MAT.

- In Chapter 5 I give practical details on how to implement the MAT.
- Chapter 6 describes applications of the unstructured MAT, such as point cloud simplification and visibility analysis.
- Chapter 7 describes applications of the structured MAT, i.e. object detection.
- In Chapter 8 I present my conclusions for this thesis as a whole and I give recommendations for future work.

Finally, Appendix A provides an overview of general characteristics of datasets that have been used for the various experiments in this thesis.

2 Background of the Medial Axis Transform

The Medial Axis Transform (MAT)[1] was introduced by Harry Blum in 1967 [Blum, 1967]. Blum was a biologist and visual scientist concerned with the perception and analysis of biological shapes using a mathematical approach. He was dissatisfied with conventional Euclidean geometry, which he describes as being "rooted in the primitive act of surveying" [Blum, 1974].

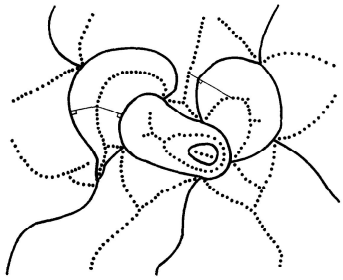
Describing a shape merely by its boundary he considered too limiting, as it did not allow one to easily 'tease out some of the essential properties of shape'. For example, Blum wondered how to naturally decompose a shape into parts, a decomposition that would remain stable even after distorting that shape in various ways (e.g. see Figure 2.1b).

With the MAT, Blum introduced a new kind of geometry that was 'a plea to re-approach the problem of shape with fresh and naïve eyes'. And indeed, researchers today consider the MAT a representation of shape that is *dual* to the conventional boundary of volumetric shape representations, because the MAT captures all the same shape aspects, i.e. it is fully equivalent in the mathematical sense. Yet, at the same time the MAT allows for a simpler, more intuitive and computationally effective way to analyse or change properties of shape when compared to the other shape descriptors [Tagliasacchi et al., 2016].

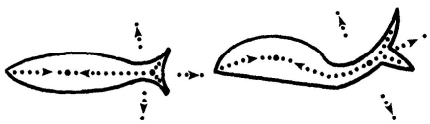
The key problems that Blum considered the MAT useful to are 1) segmentation of the field into objects (see Figure 2.1a), 2) defining locations and 3) the recognition and morphology of shapes. Unsurprisingly, these are also core applications of the MAT in this thesis. The difference is that I study not biology, but aerial point clouds, acquired using that primitive act of surveying.

Partly thanks to the advance of computers, the MAT—and particularly its 3D variant—is still an active field of research today. In this chapter I will formally define the MAT and elaborate on its properties and applications in light of the latest contributions to the field.

[1] Sometimes the MAT is referred to as medial axis function, stick figure, symmetry axis, skeleton or surface skeleton. Blum himself settled on symmetry axis, as he considered symmetry to be the crucial role of the MAT [Blum, 1973].



(a) Segmentation: each open space translates to a connected medial axis.



(b) Invariance to distortion: the topology of MAT branches remains the same despite a deformation of the boundary.

Figure 2.1: Some key features of the MAT [Blum, 1973].

2.1 The geometry of shape and geographical data models

We use geographical data models to represent in a computer, what Goodchild [1992] calls *geographical reality*: empirically verifiable fact about the real world. These data models are limited representations of reality and constrained by the finite, discrete nature of computers. Through the use of such data models we are able to (automatically) analyse and interpret geographical information about natural and artificial features in the terrain [Zhou and Zhu, 2013].

The geometry of a geographical data model is implied by its shape descriptor. Most common is the boundary representation, i.e. a description of a shape or object by means of its boundary.

Definition 2.1. *An object or shape is a compact spatial subset* $\mathcal{O} \subset \mathbb{R}^n$ *with a 2-manifold boundary* $\mathcal{S} = \delta\mathcal{O}$.

As shown in Figure 2.2 the earth as a whole can be seen as an object that satisfies this definition, where the space occupied by $\mathcal{O}$, i.e. the earth itself, is called the interior of $\mathcal{O}$ and its complement, i.e. $\mathbb{R}^n \setminus \mathcal{O}$ or the sky, the exterior.

Usually a study area is only a very small part on the surface of the earth with a boundary that is not closed. However, because we know that the study area is a subset of the object that represents the entire earth, we can imagine the boundary to extend at the edges of the study area to form a closed object (see Figure 2.2). Consequently, the concepts of interior and exterior are conceptually still valid.

This reasoning can be taken even further, i.e. we can consider the entire study area to be a single object or a composition of many objects, i.e. a *scene*, where each object may correspond to a structure of interest on the surface.

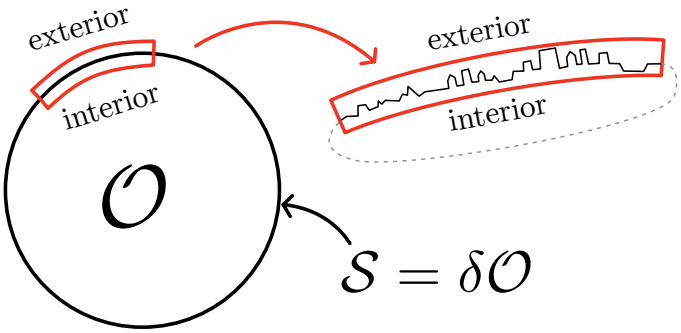


Figure 2.2: The earth can be considered a single object $\mathcal{O}$ with a manifold surface $\mathcal{S}$.

These definitions and concepts are important not only for the content of this chapter, but also for understanding the meaning of the MAT, that is described in detail in the next chapter, for a geographical data model.

2.2 Defining the Medial Axis Transform

The MAT of an object $\mathcal{O}$ consists of interior and exterior *medial balls*; a set of balls residing in the interior or the exterior of $\mathcal{O}$ respectively. Medial balls can be characterised by their maximality property which states that a medial ball can never be contained by another medial ball.

Definition 2.2. *A ball B is an interior medial ball of an object $\mathcal{O}$ if it is maximal in $\mathcal{O}$, i.e. if B is a subset of $\mathcal{O}$ and any ball that contains B is not contained in $\mathcal{O}$.*

See Figure 2.3 for an example of an interior medial ball. The exterior medial ball can be defined analogously.

Definition 2.3. *A ball B is an exterior medial ball of an object $\mathcal{O}$ if it is maximal in $\mathbb{R}^n \setminus \mathcal{O}$, i.e. if B is a subset of $\mathbb{R}^n \setminus \mathcal{O}$ and any ball that contains B is not contained in $\mathbb{R}^n \setminus \mathcal{O}$.*

A medial ball is guaranteed to be empty, i.e. it does not contain any part of $\mathcal{S}$. However, it does by definition touch $\mathcal{S}$ at two or more points. And, in case of a smooth—i.e. C^2-continuous—boundary, a medial ball is tangent at those points where it touches $\mathcal{S}$ [Ma et al., 2012].

The set of all medial balls of $\mathcal{O}$ is called the *Medial Axis Transform*, denoted $\mathcal{M}(\mathcal{O})$.

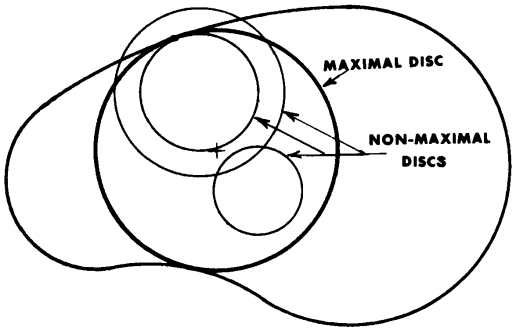
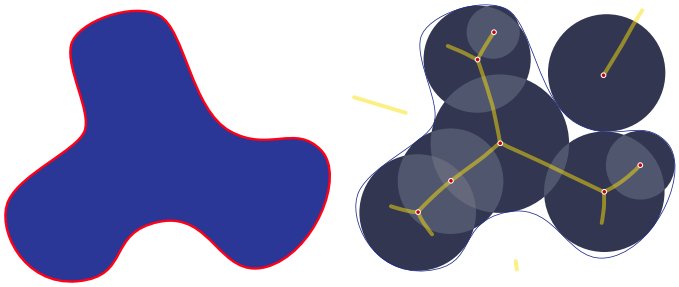


Figure 2.3: A maximal ball (disc in 2D) touches the boundary in at least two points and does not intersect with it [Blum, 1974].



(a) A object or shape $\mathcal{O}$.

(b) The corresponding medial axis and some of its medial balls.

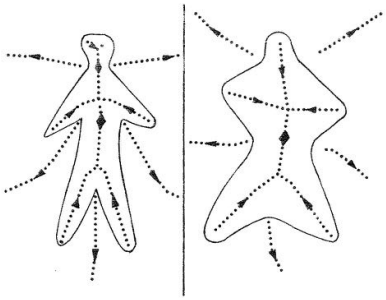
Figure 2.4: The Medial Axis Transform of a two-dimensional shape.

Definition 2.4. *The Medial Axis Transform $\mathcal{M}(\mathcal{O})$ of an object $\mathcal{O}$ is the set of centres $\mathcal{C}$ and corresponding radii $\mathcal{R}$ of all medial balls in $\mathcal{O}$, i.e. $\mathcal{M}(\mathcal{O}) = \langle \mathcal{C}, \mathcal{R} \rangle$.*

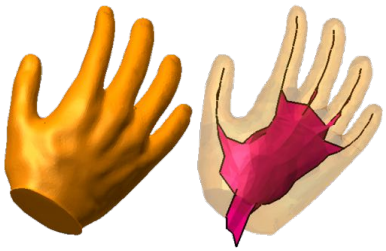
Figure 2.4b illustrates this. $\mathcal{C}$, in itself sometimes also referred to as the Medial Axis (MA), represents a skeleton-like structure that is medial to $\mathcal{O}$. $\mathcal{R}$ relates $\mathcal{C}$ to the surface $\mathcal{S}$.

Various other equivalent definitions of the MAT exist (see e.g. Tagliasacchi et al. [2016]), however the one stated here is most appropriate for this thesis.

In order to be able to study local properties of the MAT, I will now define the *medial atom*. The MAT can be seen as a collection of medial atoms, each atom representing a corresponding tuple of a centre and a radius.



(a) Early illustration of the 2D MAT (consisting of edges and points) of two 'anthropomorphs' [Blum, 1967].



(b) The 3D MAT (right) of a surface model of a hand (left) consists of faces, edges and points [Amenta and Kolluri, 2001].

Figure 2.5: The MAT

Definition 2.5. *A medial atom a is a tuple of a centre $\mathbf{c} \in \mathcal{C}$ and radius $r \in \mathcal{R}$ that correspond to the same medial ball, i.e. $a = \langle \mathbf{c}, r \rangle$.*

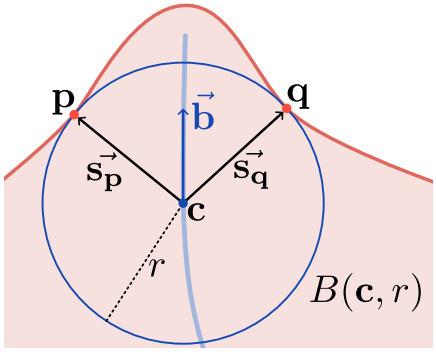
In this thesis I will often simply use the term *atom* to refer to a medial atom.

The MAT can be divided into an interior part and an exterior part. The *interior MAT* consists of its interior atoms, and the *exterior MAT* consists of its exterior atoms. A single object can only have an exterior MAT if that object is non-convex, since for convex object it is not possible to find exterior medial balls. In case of a composition of multiple objects, the interaction between these object can also lead to an exterior MAT, even if all objects are convex.

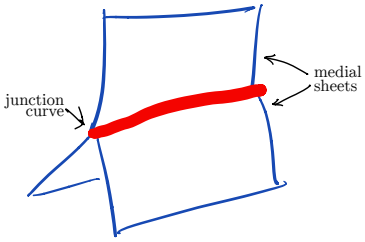
Finally, a remark on the dimensionality of the MAT. Indeed, Blum, while he himself mostly wrote about the 2D MAT (see Figure 2.5a), already said the MAT deserves to be extended to three and possibly higher dimensions [Blum, 1974]. Since I study 3D point clouds, in this thesis I am focussing on the three-dimensional MAT, i.e. its embedding in $\mathbb{R}^3$ (see Figure 2.5b). For reasons of simplicity and clarity I do use the 2D MAT in many illustrations, because the MAT typically behaves the same in $\mathbb{R}^2$ as it does in $\mathbb{R}^3$. An intuitive way to think about this, is to consider the 2D MAT to be a planar cross section of the 3D MAT.

2.2.1 The local geometry of a medial atom

The medial atom is elementary to the MAT, since ultimately all parts of the MAT are merely aggregates of medial atoms. Therefore also the intrinsic properties



(a) The local geometry of a medial atom.



(b) The structure of medial sheets and junction curves in the 3D MAT.

Figure 2.6: Characteristics of the MAT.

of the medial atom are fundamental to understanding the MAT. I refer to these properties as the *local geometry* of a medial atom.

Considering a medial atom $a = \langle \mathbf{c}, r \rangle$, see Figure 2.6b, I use the following terminology to describe its local geometry.

medial point : the point $\mathbf{c}$

radius : the scalar r

medial ball : the medial ball corresponding to a with center $\mathbf{c}$ and radius r, i.e. $B(\mathbf{c}, r)$

feature point : a surface point where the medial ball of a touches $\mathcal{S}$. Notice that all feature points are per definition equidistant with a distance r from $\mathbf{c}$. I consider each medial atom to have two feature points $\mathbf{p}$ and $\mathbf{q}$, respectively the primary feature point and the secondary feature point/footnoteIn special cases there can be more than two feature points, however these are not important in the context of this thesis since the ball-shrinking algorithm (Section 3.1) that I use to approximate the MAT can only compute medial atoms with exactly two feature points..

spoke/spoke vector : the vector from $\mathbf{c}$ to a feature point. The primary spoke, to $\mathbf{p}$, is denoted $\vec{\mathbf{s}}_{\mathbf{p}}$ and the secondary spoke, to $\mathbf{q}$, is denoted $\vec{\mathbf{s}}_{\mathbf{q}}$.

medial bisector : a vector of unit length that bisects the spoke vectors $\vec{\mathbf{s}}_{\mathbf{p}}$ and $\vec{\mathbf{s}}_{\mathbf{q}}$, denoted $\tilde{\mathbf{b}}$.

separation angle/object angle : the angle θ between the spoke vectors $\vec{\mathbf{s}}_{\mathbf{p}}$ and $\vec{\mathbf{s}}_{\mathbf{q}}$.

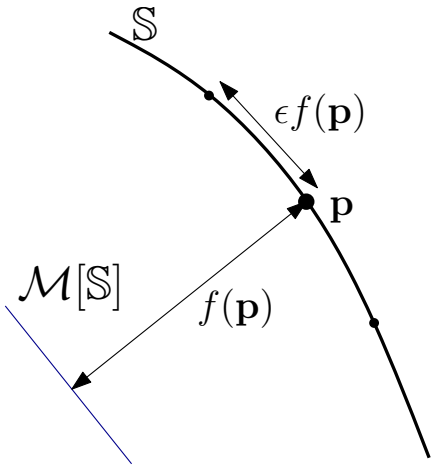


Figure 2.7: The local feature size $f(\mathbf{p})$, i.e. the shortest distance from $\mathbf{p}$ to the point approximation of $\mathcal{M}[\mathcal{S}]$, and the ϵ-sample

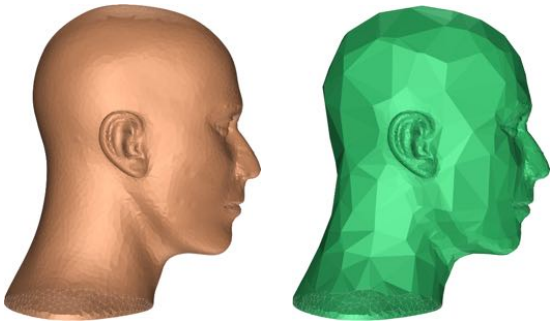
As Siddiqi and Pizer [2008] explain in detail, an atom's local geometry is useful for a number of reasons: 1) it easily quantifies local characteristics of a shape such as thickness and curvature; 2) it makes the interaction between the atom and its corresponding surface points, i.e. feature points, explicit; and 3) it can be used to define a local coordinate system.

2.2.2 The local feature size and the ϵ-sample

The local feature size is a property of surface points that is computed using the MAT. It is denoted $f(\mathbf{p})$, and defined as the Hausdorff distance from a point $\mathbf{p}$ on the boundary of an object $\mathcal{O}$ to $\mathcal{M}(\mathcal{O})$ (see Figure 2.7). The local feature size captures the curvature at $\mathbf{p}$ and the proximity of other parts of $\mathcal{S}$, since in both these cases the medial axis is close to $\mathcal{S}$.

The ϵ-sample, proposed by Amenta et al. [1998b], attempts to define a minimum sample of object that takes into account the differences in the level of detail in different parts of the same shape. A point cloud P of the boundary of $\mathcal{O}$ is an ϵ-sample if distance from any point on the boundary of $\mathcal{O}$ to the nearest point $\mathbf{p} \in P$ is at most $\epsilon f(p)$.

Many MAT approximation methods [Amenta and Kolluri, 2001; Dey and Zhao, 2004; Ma et al., 2012] assume the input to be an ϵ-sample, because it ensures that fine details are sufficiently sampled and it is often used to mathematically prove that some approximation method converges to the true MAT when $\epsilon \to 0$.



(a) Surface reconstructed from oversampled point sample (12772 points)

(b) Surface reconstructed from ϵ-sample with $\epsilon = 0.4$ (4272 points)

Figure 2.8: Decimation of a point sample using local feature size preserves points were needed. [Dey et al., 2001]

An ϵ-sample can be obtained from a densely sampled point cloud of an object by using the work of Dey et al. [2001] or, and with more efficiency, the work of Ma et al. [2012]. It is also an effective method to perform feature aware point decimation based on the MAT, with the advantage that the ϵ-parameter is scale-independent. See Figure 2.8 for an example.

2.2.3 MAT structure

The medial atoms of the MAT are organised in a branching topology, i.e. a hierarchical skeleton-like structure [Siddiqi and Pizer, 2008]. In $\mathbb{R}^2$ this means a subdivision into medial curves that intersect at points. In $\mathbb{R}^3$ it typically means a subdivision into manifold surfaces with boundaries, called *medial sheets*, that intersect at Y-intersection curves, called *junctions* see Figure 2.6b.

More formally, the MAT can be referred to as a Whitney stratisfied set, i.e. a space formed from a collection of interconnected smooth manifolds of varying dimensions (see e.g. [Damon, 2003; Mather, 1983]). A formal decomposition can be made into

surface components/medial sheets smooth 2-manifold surfaces with boundaries,

curve components curves that are not incident to medial sheets, i.e. in the case of tubular shapes, and

point components points without any incident components, i.e. the case where $\mathcal{S}$ is a sphere.

One can further decompose the MAT and classify each point or curve boundary of these components according to their local topology [Giblin and Kimia, 2003; Siddiqi and Pizer, 2008; Tagliasacchi et al., 2016]. One example is the earlier mentioned junction curve or Y-intersection curve that forms the intersection between 3 or more sheets. However, for this thesis it is sufficient to view the MAT as the composition of medial sheets and junctions.

2.2.4 Notable properties of the MAT

The MAT has a number of valuable properties, e.g. for shape analysis. These are listed here:

Complete The MAT completely describes the shape of an object. As a result, we can not only compute the MAT from $\mathcal{S}$, but we can also reconstruct that boundary using solely the information that is present in the MAT [Blum, 1967, 1973].

Topology preserving This means that $\mathcal{O}$ and $\mathcal{M}(\mathcal{O})$ have the same homotopy, i.e. the same number of connected components, voids and tunnels. This is interesting for e.g. the segmentation of a scene into distinct objects [Siddiqi and Pizer, 2008; Tagliasacchi et al., 2016].

Compact MAT components are of most of dimensionality $d-1$, hence one dimension lower than $\mathcal{O}$ itself, and the space in which it is embedded. In general, structure of a lower dimensionality are easier to analyse and process [Siddiqi and Pizer, 2008].

Hierarchical composition The structure of the MAT enables an hierarchical traversal of the different parts (each part corresponds to a medial sheet) that define an object. [Blum, 1967, 1973].

Symmetry/position The MAT is centred exactly in the middle of a shape [Blum, 1967, 1973].

Instability Small perturbations in the object boundary may cause large perturbations in the Medial Axis of that object. This means the MAT is extremely sensitive to noise [Attali et al., 2009; Choi et al., 1997; Katz and Pizer, 2003].

Smoothness the MAT components are known to be at least piecewise C^2 continuous [Pizer et al., 2003; Siddiqi and Pizer, 2008].

2.3 Algorithms and methods for the MAT in practice

The MAT has appealing theoretical properties, and for the last two decades researchers have been studying how to bring the theoretical benefits into practice[2]. The process of obtaining a useful MAT representation can be described in two main steps:

Approximation computing a discretised MAT for a given object.

Regularisation and pruning filtering parts of the MAT in order to suppress spurious medial atoms that result from the instability of the MAT and is sometimes worsened by the chosen discretisation.

While the approximation and regularisation steps are needed in practice for robustness and efficiency, these processes may introduce significant distortions to the obtained MAT approximation (see also the excellent overview paper by Tagliasacchi et al. [2016]). As a result some of the MAT properties that are guaranteed in theory may not be met in practice. The completeness property may be violated, because the MAT is only an approximation of an already discretised object boundary. This makes it for example more difficult to perform the inverse MAT, i.e. a process called *garbing*: reconstructing a boundary representation from an approximated MAT (see e.g. Amenta et al. [2001]). Also, the homotopy property may be broken during regularisation, resulting in different number of connected components. In short, there are many compromises to be made, and making the right choices also depends on the nature of the input data and the intended application.

2.3.1 Approximation

Exact computation of the MAT is difficult and computationally expensive, even for shapes bounded by simple curves in 2D [Attali et al., 2009; Biasotti et al., 2008]. Fortunately, one can also *approximate* the MAT. This is usually much less expensive in computational terms and gives adequate results in practice. To approximate the MAT essentially means that a discretised form of the MAT is computed. This means the MAT is represented using a triangulation, a point cloud or a set of voxels, rather than an algebraically defined set of surfaces. The form of discretisation naturally depends on the employed approximation method. In particular,

1. Voronoi diagram and bisector based methods yield a triangulated MAT,
2. distance transform and thinning based methods represent the MAT as a set of connected voxels, and

[2] Sheehy et al. [1995] was—as far as I know—the earliest attempt using a computer but it only targeted a limited set of polyhedral shapes.

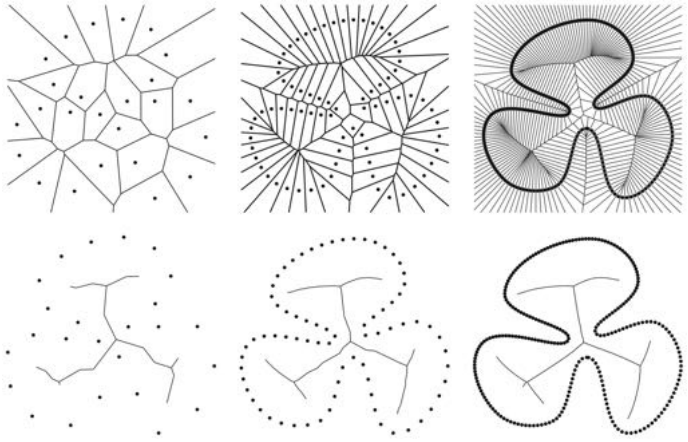


Figure 2.9: The VD can be used to approximate the MAT: the quality of the approximation (bottom row) increases with an increasing density of the boundary sample (top row). Sampling density increases from left to right. [Attali and Montanvert, 1996]

3. ball-shrinking methods yield the MAT as a point cloud.

Below, I will discuss these three groups of methods to approximate the MAT.

Voronoi and bisector based methods

Given a sufficiently dense point sample S of the boundary of a shape $\mathcal{O}$, it can be observed that a subset of the Voronoi Diagram (VD) of S approximates $\mathcal{M}(\mathcal{O})$. Brandt and Algazi [1992] gave a proof for the two-dimensional case. Observe from Figure 2.9 that those VD edges that do not intersect the boundary of $\mathcal{O}$ contribute to $\mathcal{M}(\mathcal{O})$. Furthermore, following the duality between the VD and the Delaunay Triangulation, if we take the dual of the remaining (boundary-intersecting) VD edges, we end up with an approximation of the boundary that is sampled by S. Attali and Montanvert [1997], Amenta et al. [1998a] and Gold and Snoeyink [2001] have used this duality to design algorithms that compute both this approximation of the boundary of $\mathcal{O}$, and an approximation of $\mathcal{M}(\mathcal{O})$. The approach by Gold and Snoeyink [2001] uses a simple local test to construct these.

However, these results cannot be easily extended to the three-dimensional case. As explained by Amenta and Choi [2008], the main difficulty is the presence of *slivers* in the 3D DT. These are almost flat Delaunay tetrahedra that appear

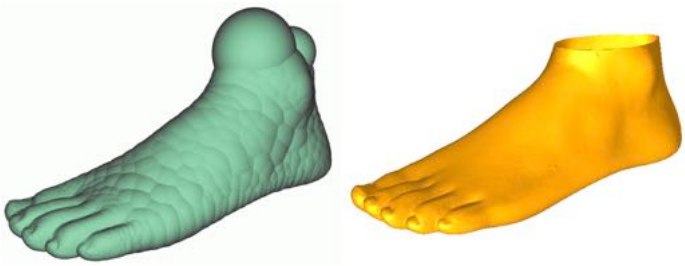
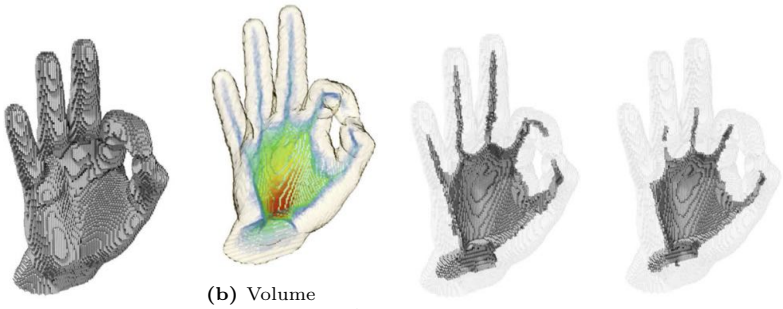


Figure 2.10: Reconstruction of the boundary of an object from just its MAT. Left: using a union of a finite number of medial balls. Right: using the power crust method. Figure from [Amenta and Kolluri, 2001].

when four or more sample points on the boundary of $\mathcal{O}$ are (almost) co-circular, which is quite common in practice. Unlike the two-dimensional case, the Voronoi vertices corresponding to these slivers lie far from $\mathcal{M}(\mathcal{O})$. These slivers are unrelated to the presence of noise in the boundary sample S. In fact an arbitrarily dense sampling will result in Voronoi vertices that are arbitrarily far from $\mathcal{M}(\mathcal{O})$. As a result, the part of the VD that corresponds to these slivers needs to be filtered out before the MAT of a 3D shape can be approximated using the VD.

Approximating the 3D MAT based on the VD can be achieved in three ways. One is by iteratively peeling away Voronoi cells starting from the boundary of an object. In the method of Attali and Montanvert [1997], a cell is only removed if that does not change the topology of the MAT. Furthermore, they implement a threshold for filtering slivers. A second approach is based exclusively on filtering the elements of the VD. To achieve this, Dey and Zhao [2004] use two different local criteria that are independent of scale and sampling density. This results in an approximated MAT that insensitive to noise to some extent. Unfortunately the outputted MAT may have holes, which is topologically incorrect. The third approach was presented by Amenta and Kolluri [2001]: the power shape. Instead of using the VD directly to construct the MAT, they use it only to find *poles*: the farther vertices of the Voronoi cell of a sample point on the boundary of $\mathcal{O}$. Amenta et al. [2001] show formally that these poles converge towards the exact MAT, as the point density of S increases. A weighted VD, i.e. the *power diagram*, of poles (weighted with the radius defined by their Voronoi balls) is constructed, and the dual of this diagram, the *regular triangulation*, approximates the MAT of $\mathcal{O}$. Notice that the power diagram can also be used for garbing, i.e. to reconstruct the surface of $\mathcal{O}$ from the MAT (See Figure 2.10). A drawback from the power shape method is that it requires twice the calculation of a Voronoi-like diagram, whereas other Voronoi-based methods only require one such calculation. Furthermore, the output includes tetrahedra, which may be



(a) Input model

(b) Volume rendering of the distance transform.

(c) $\lambda = 11$ voxels

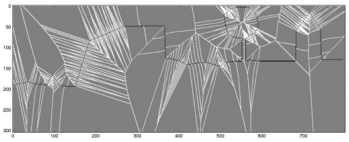
(d) $\lambda = 15$ voxels

Figure 2.11: MAT extraction using the discrete λ-MAT method of Chaussard et al. [2011]. For Fig. 2.11b: Blue-green-yellow-red in order of increasing distance.

problematic for further processing. Tam and Heidrich [2003] therefore choose to extend and improve an earlier variant of the power shape algorithm, which outputs many duplicate geometries but does not exhibit the sliver problem [Amenta and Kolluri, 2001]. Miklos et al. [2010] further improved the robustness of the algorithm and simplified parts of it by assuming the input to be a surface mesh rather than a point cloud.

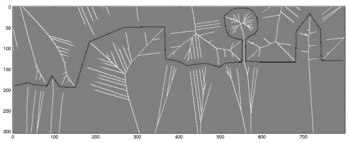
Jalba et al. [2012] are eager to point that Voronoi methods, such as the one of Miklos et al. [2010], which is supposedly one of the best continuous Voronoi-based MAT approximation algorithms, may suffers from numerical degeneracies that affect the quality of the output. In theory this can be fixed by using robust predicates. However, robust predicates are costly and using them would add to the complexity of the implementation.

Bisector based methods are conceptually similar to the Voronoi based methods, although they do not explicitly compute the VD. Instead, bisector planes between surface points—analogous to the faces of a VD—are calculated in a brute force manner which is computationally very expensive [Barequet et al., 2008; Culver et al., 2004; Lee, 1982]. The medial scaffold of Leymarie and Kimia [2007] achieves a higher efficiency by only computing the bisector planes that are relevant to the MAT. This is achieved by finding particular types of medial points prior to the bisector computation. Results are similar to Voronoi based methods and have the added benefit of the medial point classification. However, the medial scaffold is still more expensive to compute than Voronoi methods [Tagliasacchi et al., 2016].



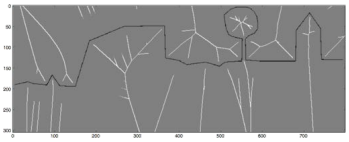
(a) MAT without pruning ($\lambda = 0$)

(b) Reconstructed surface without pruning.



(c) MAT with medium pruning ($\lambda = 6$)

(d) Reconstructed surface with medium pruning. Despite the simplified MAT, the corresponding boundary surface remains almost unchanged.



(e) MAT with strong pruning ($\lambda = 10$).

(f) Reconstructed surface with strong pruning. Notice how the tree is separated from the the ground and that edges have become rounder.

Figure 2.12: The effect of different levels of pruning on the 2D MAT. Own implementation based on a variant of the λ-MAT described by Hesselink et al. [2005].

Voxel based methods

This category of approximation methods obtains the MAT as a voxel image. Thinning methods take a voxel image as input, and iteratively peel away layers of voxels. A 3D example is described by Borgefors et al. [2008]. A prime feature of thinning is the preservation of topology. However, the shape of the resulting skeleton often depends on the order in which voxels are thinned.

Other voxel-based approximation methods are based on the distance transform: an image where each element is labelled with the shortest distance to the boundary of an object $\mathcal{O}$ [Chaussard et al., 2011; Chazal and Lieutier, 2005; Hesselink et al., 2005; Sud et al., 2004]. Different measures for distance can be used and will result in different skeletons. From the distance transform the MAT can be intuitively recognised as a set of 'ridges': lines that indicate local directional maxima. Voxels that contribute to the MAT-approximation can be identified in a number of ways. Foskey et al. [2003] take the gradient field, a derivative of the distance transform, where every element consists of a unit vector in the direction of its closest boundary point. For every pair of adjacent vectors, the angle between them is computed and if this angle is larger than a threshold θ a facet between them is added to their MAT-approximation; the θ-SMA.

Voxel based methods are often rotation invariant, i.e. they suffer from spurious branches in the MAT when the object is not axis-aligned [Borgefors et al., 2008] (Figure 2.12a illustrates this) and sub-optimally centred, i.e. the axis is not precisely medial (with the notable exception of Sud et al. [2007]). This is due to the non-continuous and gridded coordinates the output is not rotation invariant.

While parallelised computation is possible both for the computation of the distance transform [Cao et al., 2010] and the thinning methods [Saha et al., 2016], this does not mean that scalability of is feasible. Voxel-based methods are relatively fast for small inputs, but for inputs that have a resolution of 1000^3 voxels or more, the algorithms become significantly slower and more memory demanding since these algorithms usually scale linearly with the number of input voxels [Sobiecki et al., 2013].

Shrinking ball methods

Ma et al. [2012] introduced the shrinking ball algorithm. Like the Voronoi-based methods and unlike the voxel-based methods, it outputs continuous coordinates, but it is based on a much simpler idea. It takes an oriented point cloud as input, i.e. one that includes point normals, and outputs a set of points as the MAT approximation. If normals are not available, they can be estimated beforehand using local plane fitting. The shrinking ball algorithm is based on the assumption that the centre of the medial ball corresponding to a sample point s must be

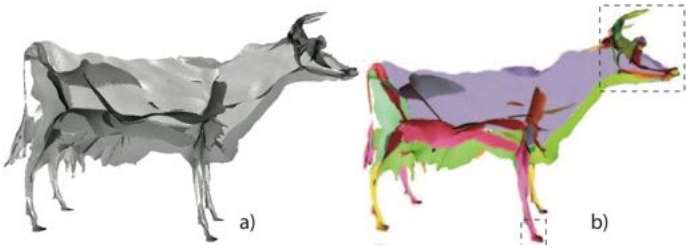


Figure 2.13: Two different approaches to reconstruct a surface MAT in the method of Jalba et al. [2012]. Left: Delaunay reconstruction, right: reconstruction based on clustering of sheets (rendered in different colours).

positioned somewhere along line through the normal of s. Starting with a large ball B, B is iteratively shrunk until it neither touches nor contains any other points than s and only one more sample point. During this process a KD-tree is used as an efficient spatial index. A more detailed description of the algorithm is given in Section 3.1.

The simplicity of the algorithm makes it very efficient and scalable. According to Tagliasacchi et al. [2016] the shrinking-ball algorithm is the fastest MAT approximation method currently available. Compared to the power shape method of Amenta and Kolluri [2001], it is between 2 and 15 times faster, and an additional speed up of 5 to 10 times is gained when using a GPU implementation [Ma et al., 2012]. [Jalba et al., 2012] gained another order of magnitude in computation time by sacrificing some degree of quality by using an approximate KD-tree rather than an exact one.

Structuration The main limitation of the ball-shrinking algorithm is that it outputs only an unstructured set of medial atoms, i.e. the medial sheet surfaces are not explicitly reconstructed. Structuration is the process of overcoming this limitation by constructing the sheet surfaces, e.g. as a triangular mesh, from which explicit structural information about the hierarchy of the MAT can be obtained [Delame et al., 2016].

If the input shape is available as a mesh, structuration can be achieved by collapsing the surface mesh to the medial atoms along the spoke vectors [Jalba et al., 2012]. However, this results in a polygons soup of triangles, which cannot be directly used for analysis or processing operations besides visualisation [Delame et al., 2016]. Other methods do not require a surface mesh, and work solely with the medial atoms. These are typically generic point cloud reconstruction algorithms, such as the ball pivoting algorithm [Bernardini et al., 1999] as used

by [Jalba et al., 2012]. However, also with such methods there is no guarantee that a topologically accurate medial mesh is obtained [Delame et al., 2016]. Kustra et al. [2014] have proposed a surface reconstruction method that is designed specifically for extracting complex manifolds with intersecting surfaces. They show it works well on very dense MAT approximations computed by the ball-shrinking algorithm.

Comparison of 3D MAT approximation methods

To summarise, I will now list the main characteristics of the three categories of MAT approximation methods.

For Voronoi-based methods:

- Approximates also the structure of the MAT;
- Proven to convergence to the theoretical MAT;
- Computation of the VD may suffer from numerical degeneracies;
- Presence of slivers;
- Most complex implementation and difficult to parallelise (no known parallelised variants).

For the voxel-based methods:

- Highly parallelised computation possible;
- Gridded coordinates lead to an output that is not rotation invariant and has sub-optimal centredness;
- Slow computation for large inputs, when compared to the other approaches;
- Very high memory requirements for large inputs.

For the ball-shrinking methods:

- Fastest and highly parallelisable computation;
- Simple algorithm and thus simple to implement;
- Lower memory requirements than Voronoi methods [Ma et al., 2012];
- Structure of the MAT is not computed;
- Requires in advance the approximation of point normals.

2.3.2 Regularisation and pruning

Tagliasacchi et al. [2016] remark that virtually all MAT papers mention the MAT's instability, i.e. its sensitivity to small shape changes, as *the* key challenge in computing a usable MAT. Measures to prevent the occurrence of unstable medial atoms can be taken before, during or after the MAT approximation. For instance, some researcher smoothen or simplify the input surface prior to MAT approximation (see e.g. Dey and Zhao [2004]). However, by far most methods aim to remove unstable medial atoms after the MAT approximation. These so-called *regularisation* or *pruning* methods usually work by computing an importance measure for each medial atom, followed by a thresholding step in which unstable parts of the MAT are filtered out or simplified. Notice that, while the primary objective is often to remove spurious or noisy parts of the MAT that overly complicate its structure, pruning methods can often also be used to simplify the MAT even further with the goal of generalising the input shape (see Figure 2.14).

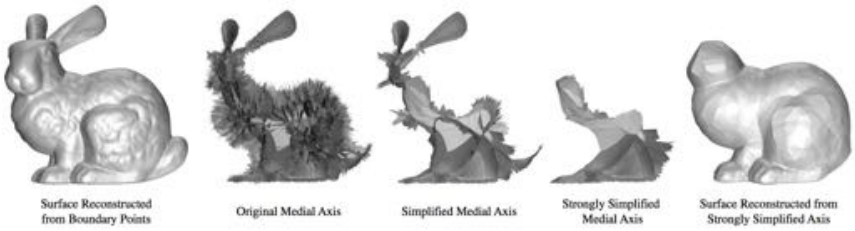
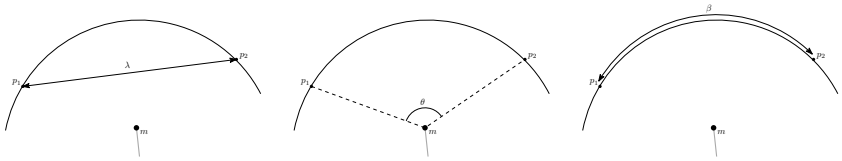


Figure 2.14: Simplifying the MAT (figure from Tam and Heidrich [2003]).

Following is an overview of prominent 3D pruning methods.

Atom based pruning methods are the most common. They usually define an importance measure for each medial atom in the MAT and filter medial atoms by thresholding the importance measure. The resulting (pruned) MAT is usually a subset of the original MAT. Some methods preserve topology, others do not or only up to a certain level. The main challenge is often selecting the optimal threshold value, which often turns out to be a manual process.

- λ-MAT, where λ is the shortest distance between the set of boundary points that correspond to the same point on the MAT [Chazal and Lieutier, 2005] (see Figure 2.15a). All points on the MAT for which this distance is higher than λ contribute to the λ-MAT. Topology is preserved until the so-called weakest feature size is reached: the largest λ for which the (inner) MAT for a single object is still completely



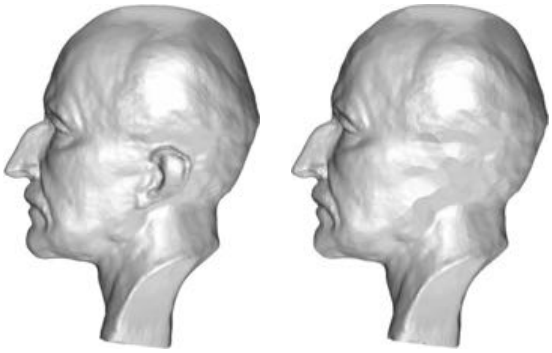
(a) λ-MAT. **(b)** θ-MAT **(c)** Boundary potential.

Figure 2.15: Common pruning measures from literature. They all define a metric based on the two boundary points p_1 and p_2 that correspond to the same medial axis point m.

connected (compare Figures 2.12c and 2.12e). The λ-MAT has proven particularly popular in approximation methods that are based on the continuous distance transform [Chaussard et al., 2011; Hesselink et al., 2005].

- θ-MAT, where θ, or the separation angle, is the angle between two boundary points that correspond to the same point on the MAT (see Figure 2.15b). In case of more than two corresponding boundary points, the ones that result in the largest angle are chosen. The larger the separation angle of a given point on the MAT, the more stable that point is considered to be. It is generally not considered to be topology preserving. But it has been used by many researchers [Amenta and Kolluri, 2001; Attali and Lachaud, 2001; Attali and Montanvert, 1996; Foskey et al., 2003]. Sometimes it is used together with other criteria [Dey and Zhao, 2004], or even in combination with topology-preserving constraints [Sud et al., 2007].

- The scale axis transform was introduced by Giesen et al. [2009], and then implemented by [Miklos et al., 2010]. It is different from other continuous pruning methods because it can result in a *superset* of the original MAT, where the other methods always result in a subset. The idea is to 1) scale the radii of the medial balls of an initial MAT with a factor $s > 1$, 2) recompute the MAT of the union of scaled medial balls, and 3) re-scale the new medial balls with a factor of $1/s$. The resulting MAT has a simplified shape, because many relatively small medial balls will have disappeared in the second step. When comparing this method to the θ-MAT, Miklos et al. [2010] argue that it works in a more global fashion (because medial balls can disappear due to a merging with relatively far but significantly large balls) and is therefore of a higher quality. Topology is preserved up to a certain value of s, where after the topology might change due to the closing of holes. The resulting object may thus have a different genus. However, this problem is fixed in the work by Faraj et al. [2013], which improves



(a) Before pruning **(b)** After pruning.

Figure 2.16: Using the part-based pruning method of Tam and Heidrich [2003], features can be selectively removed from a model.

and extends the scale axis transform. They use a progressive filtration technique to compute the simplified MAT for all scales at once which enables real-time exploring for the effects of different scale-threshold.

- The boundary potential of a point on $\mathcal{M}(\mathcal{O})$ is defined as the shortest distance between two corresponding boundary points over the surface of $\mathcal{O}$ (see Figure 2.15c). It was originally introduced in the 2D case by Ogniewicz and Ilg [1992] and a similar concept was used by Matuk et al. [2006]. Dey and Sun [2006b] define essentially the same thing, but call it the *medial geodesic function*, and uses it to compute the curve skeleton: a topology equivalent subset of the MAT that has no area. Jalba et al. [2012] provide an efficient implementation and argue that it is a good measure because it takes into account the global shape of an object.

Sheet based pruning means that the MAT is reduced based on the selective removal of entire MAT-sheets, usually while preserving topology.

- Tam and Heidrich [2003] introduce a three-step approach: 1) decomposing the MAT into parts (the MAT sheets are cut where they meet), 2) assigning a significance value to the parts based on a) the number of triangles in the part, or b) the volume that would be removed from the input object as a result of pruning the part, and 3) performing an ordered pruning process that removes all parts with significance values within a specified range that do not alter the topology of the MAT. The order is such that outer parts are removed first. The number of iterations depends on the required level of simplification. Tam and Heidrich [2003] specifically mention that their aim is to perform

feature-based pruning, i.e. to remove specific parts of a model as demonstrated in Figure 2.16.

- Sud et al. [2007] similarly decompose the MAT into parts and iteratively remove sheets (starting with the outer ones), but in contrast to Tam and Heidrich [2003], they assign a significance value based on the highest separation angle (as defined earlier) in a sheet. Their primary reason to use part-based pruning is that it preserves topology and their objective is to identify a stable MAT.

Edge-collapse based pruning methods simplify the MAT structure by collapsing edges. Evidently a structured MAT is required for these methods, e.g. as obtained by Voronoi based approximation methods.

- Sun et al. [2013] introduce a MAT-based volume representation called the *medial mesh*. An object can be reconstructed from the medial mesh by buffering its faces and edges while considering the medial ball radii at the medial vertices in the medial mesh. An initial mesh is computed using a Voronoi-based MAT approximation methods and then simplified by collapsing edges. The edge collapses are driven by a global error between the original input surface mesh and the reconstruction of the simplified medial mesh. The error metric of Sun et al. [2013] is based on the one-sided Hausdorff distance and edges are collapse in order of increasing error until an accepted level of simplification is reached. Li et al. [2015] propose a very similar approach, but they use the quadric error metric—originally introduced by Garland and Heckbert [1997] for mesh simplification—extended with a term that quantifies the likelihood that an edge is spurious in the MAT. The advantage over Sun et al. [2013] is that this allows for optimal vertex placement after collapsing an edge.

An interesting outlier to how most MAT-related noise handling methods work is the work of Berger and Silva [2012]. They introduce the so-called medial kernel for surface points, a similarity measure defined as the likelihood that two surface point belong to a common medial ball. While their method does not explicitly compute the MAT itself, it is designed specifically for noisy and incomplete point clouds. The main difference with classical pruning methods is that they carefully consider the defects in the surface point cloud *during* the generation of medial balls, whereas pruning methods are applied only after MAT generation. In their method they generate a candidate ball for each pair of surface points and quantify the likelihood it is a medial ball based on the emptiness of the candidate ball and its tangentiality—i.e. how well pre-computed surface point normals align with the local medial geometry. Unfortunately, this method has quadratic time complexity.

To conclude, the described pruning methods are mostly applied to a MAT that was derived from fairly dense and high quality surface meshes or point clouds—when compared to typical geographical point clouds. For these inputs, existing pruning methods can be considered to be effective with the main caveat that pruning is often a trade-off between robustness to noise and the topological and geometric fidelity of the pruned MAT when compared to the original input shape.

2.4 Summary

This chapter introduced the MAT and reviewed method and algorithms from literature. I explained how the MAT is defined in relation to the conventional boundary representation, that it is a non-manifold structure made out of so-called medial sheets that are in turn made out of medial atoms that can be described by their local medial geometry. Then I reviewed existing algorithms to compute to approximate the MAT, given a boundary surface or point cloud, and various ways to prune the MAT. I identified the ball-shrinking algorithm by Ma et al. [2012] as the most promising for geographical point clouds.

In the next chapter I explain how I have modified the ball-shrinking algorithm to be robust to the kind of noise that is typical for geographical point clouds.

3 The unstructured MAT

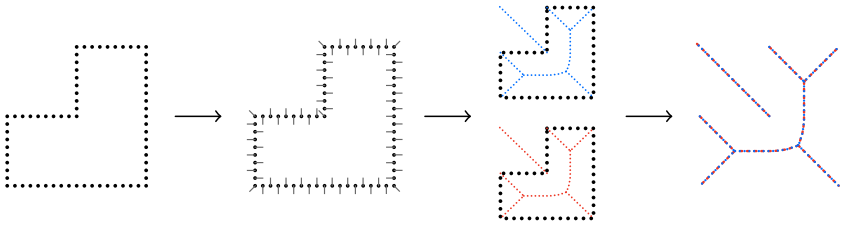
In an ideal world we would have complete access to the MAT of a geographical data model including all of its properties as described in Chapter 2. But, our world is not an ideal one. The core contribution of this thesis is that I show that the MAT can be computed robustly for aerial point clouds in a form that is practically useful even in our non-ideal reality. In the coming two chapters I explain the challenges in computing the MAT for real-world datasets, how to meet them, and what are the resulting implications on the usefulness of the resulting MAT representation.

The first challenge that we encounter is that—even assuming a perfect representation of the surface—there are no algorithms to compute the MAT exactly for all shapes [Attali et al., 2009]. And although there is a subset of shapes for which we can compute it exactly, the computational cost of these algorithms is so high that they are not practical [Attali et al., 2009; Biasotti et al., 2008]. As a result we must work with algorithms that merely *approximate* the MAT, which of course implies that we compromise a bit on the different MAT characteristics (most notably full preservation of information in comparison to surface representation) that made the MAT so compelling in the first place.

Secondly, as described in Section 1.1, an aerial point cloud is always incomplete and inaccurate to some degree and even if we had an algorithm that computes the MAT exactly for all shapes in feasible time, the result would still be distorted when compared to reality.

The computation of the MAT of a geographical data model thus requires an approximation algorithm that on the one hand does not assume a continuous and well sampled surface, and on the other hand is robust to measurement noise, since the MAT is notably unstable for small distortions in the surface geometry (as explained in Chapter 2). This is the core topic of this chapter.

Finally, for the next few chapters the reader should be aware that I break down the MAT into an *unstructured* part (this chapter) and a *structured* part (next chapter). The unstructured MAT (Figure 3.1) concerns its geometry, i.e. the complete set of medial atoms, whereas the structured MAT concerns the subdivision of the MAT into medial sheets and the connectivity of those sheets. Further details and considerations on the implementation of the MAT, such as data structures and scalability, are given in Chapter 5.



1. Surface point cloud

2. Normal approximation

3. MAT approximation for both normal orientations

4.Unstructured MAT

Figure 3.1: The main steps in computing the unstructured MAT from a point cloud

3.1 The ball-shrinking algorithm

Out of all the known MAT approximation algorithms that I have described in Chapter 2, I believe the ball-shrinking algorithm from Ma et al. [2012] is most suitable for the geographical case because it is (1) point-based, (2) simple, (3) fast, and (4) scalable. Since modern geographical data models are typically acquired as point clouds it is preferred to have methods that work directly with points. This avoids costly conversions to limiting lower-dimensional or simply less accurate representations. One might even say it is the only way that takes full advantage of the wealth of information that is contained in a point cloud. Apart from point-based, the ball-shrinking algorithm is also conceptually simple. With this I not only mean that it is intuitive and easy to understand and that it can be implemented in a dozen lines of code, it also means it does not suffer from numerical robustness issues as triangulation algorithms do. Furthermore because of its simplicity it is also easy to extend and adapt the method. I demonstrate that in Section 3.3.

The ball-shrinking algorithm was designed on the basis of three important observations about the MAT:

1. every medial ball touches the surface in at least two feature points,
2. wherever a medial ball touches the surface it is tangent to it, and
3. every medial ball is empty, i.e. there are no surface points on its interior.

From the second observation it immediately follows that the normal vector $\vec{\mathbf{n}}$ of a feature point $\mathbf{p}$ must be aligned with the centre of the medial ball $\mathbf{c}$. In other words, in addition to $\mathbf{p}$, also $\mathbf{c}$ must lie on the line L through $\vec{\mathbf{n}}$.

We then only need one additional point $\mathbf{q}$ and we can construct a ball that touches $\mathbf{p}$ and $\mathbf{q}$ and is centred on L. If we now consider that both $\mathbf{p}$ and $\mathbf{q}$

Algorithm 1: The extended `shrinkBall` algorithm. Highlighted in yellow are the denoising heuristics.

Input : a KD-tree of the surface point cloud T,
a surface point $\mathbf{p}$, and
it's normal vector $\vec{\mathbf{n}}$, and
for denoising the parameters θ_0 and θ_1
Output: the medial ball centre $\mathbf{c}$,
the medial ball radius r

$i \leftarrow 0$
$r \leftarrow r_{init}$
$\mathbf{c} \leftarrow$ `computeCentre`$(\mathbf{p}, \vec{\mathbf{n}}, r)$
repeat
  $q_{next} \leftarrow$ `nearestNeighbour`$(T, \mathbf{c})$
  $r_{next} \leftarrow$ `computeRadius`$(\mathbf{p}, \vec{\mathbf{n}}, \mathbf{q_{next}})$
  $\mathbf{c_{next}} \leftarrow$ `computeCentre`$(\mathbf{p}, \vec{\mathbf{n}}, r_{next})$
  if $r_{next} > r - \epsilon_{conv}$ **then**
    break
  else if $i = 0$ *and* $\theta_0 > \angle \mathbf{p c_{next} q_{next}}$ **then**
    break
  else if $i > 0$ *and* $\theta_1 > \angle \mathbf{p c_{next} q_{next}}$ **then**
    break
  $\mathbf{c} \leftarrow \mathbf{c_{next}}$
  $r \leftarrow r_{next}$
until *a break statement is executed*

are surface points, then we only need to check if the ball is empty in order to determine if it is medial. If this is the case then we have found ourselves a medial ball with the feature points $\mathbf{p}$ and $\mathbf{q}$.

We are thus trying to find a medial ball from a given tuple $\mathbf{p}, \vec{\mathbf{n}}$. The question therefore is: how do we select an appropriate $\mathbf{q}$ so that the triplet $\mathbf{p}, \vec{\mathbf{n}}, \mathbf{q}$ defines a medial ball? The ball-shrinking algorithm (see Algorithm 1) does this using a process of iterative ball shrinking. At each iteration a new candidate ball is constructed that is smaller than the previous one and closer to the final *medial* ball. Every ball is constructed so that it touches $\mathbf{p}$ and is centred on L, only $\mathbf{q}$ changes. A new $\mathbf{q}$, denoted $\mathbf{q_{next}}$, is found by selecting the closest point from the centre $\mathbf{c}$ of the current ball. Using $\mathbf{p}, \vec{\mathbf{n}}, \mathbf{q_{next}}$ we can compute the centre of the next ball $\mathbf{c_{next}}$, at which point we move on to the next iteration. The algorithm terminates when an empty ball is found.

The ball-shrinking algorithm always converges because at each iteration $\mathbf{q_{next}}$

Algorithm 2: The `shrinkBalls` algorithm

Input : an oriented point cloud P_o
Output: an interior and an exterior medial ball for each $\mathbf{p} \in P_o$

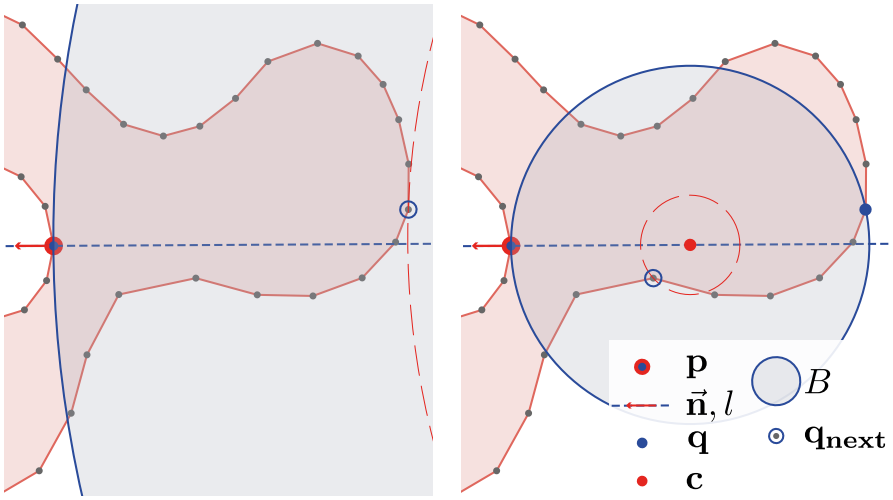
```
1 T ← computeKDTree(P)
2 foreach p, n ∈ P_o do
3 |   shrinkBall(T, p, n)
4 |   shrinkBall(T, p, −n)
```

will be closer to $\mathbf{c}$ than $\mathbf{q}$, which means the new ball will always fit inside the current one. In other words: shrinkage is guaranteed. Finally an empty ball must occur because we only have a finite number of surface points. According to Ma et al. [2012] and my own tests, it typically takes less than 10 balls before an empty one is found, and this empty ball must be medial because it also touches the surface at two points. Figure **??** shows an example of a series of balls that is computed for one surface point.

Notice that Algorithm 1 lists my own variant of the core ball-shrinking algorithm. The main difference from the original pseudo code by Ma et al. [2012] is the addition of denoising heuristics (lines 10-13). I will further explain these in Section 3.3, but notice that when these lines are omitted the behaviour of the original algorithm is obtained. In addition, the `shrinkBall` algorithm would typically be run inside a loop that iterates over every point of an oriented input surface point cloud. In each iteration of that loop, the ball-shrinking algorithm is ran twice: once with $\mathbf{p}, \vec{\mathbf{n}}$ and once with $\mathbf{p}, -\vec{\mathbf{n}}$. In this way we compute both the interior and the exterior medial ball for $\mathbf{p}$. Algorithm 2 shows this.

Inputs and outputs

Evidently the ball-shrinking algorithm takes a surface point cloud as input and in addition to the point coordinates it also requires a normal vector for each point. Such a point cloud, where each point has an associated normal vector, is called an *oriented* point cloud. In Section 3.2 I give further considerations on how to obtain the normal vectors. On the output side of the algorithm we basically obtain the ball centres and radii. Because we obtain both the interior and exterior medial balls, we will compute two medial balls for each surface point. The space complexity of the output is therefore $2n$, thus linear with the input size.



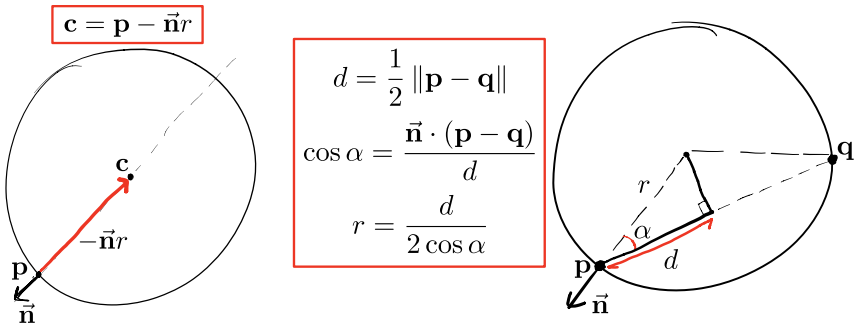
(a) Initial ball

(b) Second iteration.

(c) Third iteration.

(d) Fourth iteration yields an empty ball.

Figure 3.2: Ball shrinking iterations with the ball-shrinking algorithm. A legend is given in b.



(a) `computeCentre(`$\mathbf{p}, \vec{\mathbf{n}}$`)`

(b) `computeRadius(`$\mathbf{p}, \vec{\mathbf{n}}, \mathbf{q}$`)`

Figure 3.3: Geometry functions used in the `shrinkBall` algorithm.

Supporting functions

Algorithm 1 calls three different functions on lines 5-7:

1. The function `nearestNeighbour(`T, q`)` simply returns the point from point cloud P that is closest to the query point q. This function can be implemented efficiently using a KD-tree [Bentley, 1975] and has a time complexity of $O(\log n)$. Notice that the KD-tree of P must be constructed prior to running the nearest neighbour queries, this has a time complexity of $O(n \log n)$ [de Berg et al., 2000].
2. The function `computeRadius(`$\mathbf{p}, \vec{\mathbf{n}}, \mathbf{q}$`)` computes the new ball radius in constant time ($O(1)$).
3. The function `computeCentre(`$\mathbf{p}, \vec{\mathbf{n}}, r$`)` computes the new ball centre in constant time ($O(1)$).

Diagrams for the latter two functions are given in Figure 3.3.

Time complexity

The time complexity of the entire ball-shrinking algorithm is $O(n \log n)$. This follows from two observations.

1. The `shrinkBall` algorithm is called $2n$ times.
2. Each call to the `shrinkBall` algorithm takes $O(\log n)$ time, because the average number of iterations can be considered to be constant on average (Ma et al. [2012] around 6 7) and the only non constant time operation is the `nearestNeighbour` call on line 5, which takes $O(\log n)$ time.

We can therefore conclude that the expected time complexity of the `shrinkBalls` algorithm is $O(n \log n)$, which includes the $O(n \log n)$ construction of the KD-tree. As noted by Ma et al. [2012] the worst-case time complexity is actually $O(n^2)$, i.e. the case where all surface points are visited for the approximation of each medial ball (e.g. it could happen if the input surface is a sphere). However, this is extremely unlikely in practice and the real-world performance is described as 'near-linear'. I should point out here that the average number of iterations does depend on the complexity of the shapes in the surface point cloud. For instance, it will be higher for a surface point cloud with a large number of curved surfaces.

3.2 Normal estimation

The ball-shrinking algorithm requires an oriented point cloud as input, i.e. each points should come with a associated normal vector. Earlier works with the shrinking-ball algorithm [Jalba et al., 2012; Ma et al., 2012] always obtained these from a high quality mesh in which case the normal is well-defined and easily obtained everywhere. In the geographical case on the other hand, we usually deal with unoriented point clouds with a very heterogeneous point distribution. Because an aerial point cloud does not usually come with normals, we need to estimate these before we can run the ball-shrinking algorithm. Of course we should be aware of the effects of all this on the quality of the estimated normals and the implications for the MAT approximation that we finally obtain.

Normal estimation by local plane fitting

The most popular method to estimate normals for point clouds is to locally fit tangent planes. If we want to estimate the normal for a point p, we take a set of points N close to p (usually found using a KD-tree), and fit a plane through them. The underlying assumption is that these points are 1) all part of the same surface as p and 2) locally planar. The normal vector should then be taken orthogonal to the fitted plane. Computing the best fitting plane can be seen as a least-squares problem that is often solved using a PCA decomposition (see for instance Gross and Pfister [2011]) and everything is computed locally.

The two main advantages of this method are that it is 1) simple to implement and 2) fast to execute. The main disadvantage is that the normals are typically distorted around sharp edges. Furthermore, this distortion effect may be strengthened by the heterogeneous point distribution of aerial point clouds. Figure 3.4 illustrates this. The problem here is that a k-neighbourhood of a point is not always a good approximation of the actual surface, because the neighbouring points may be part of several different planes.

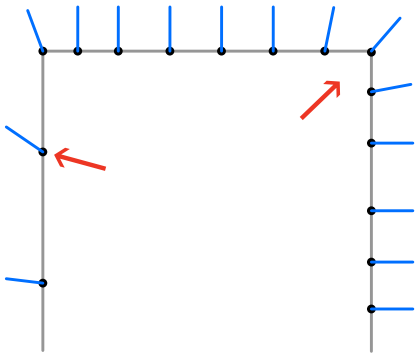
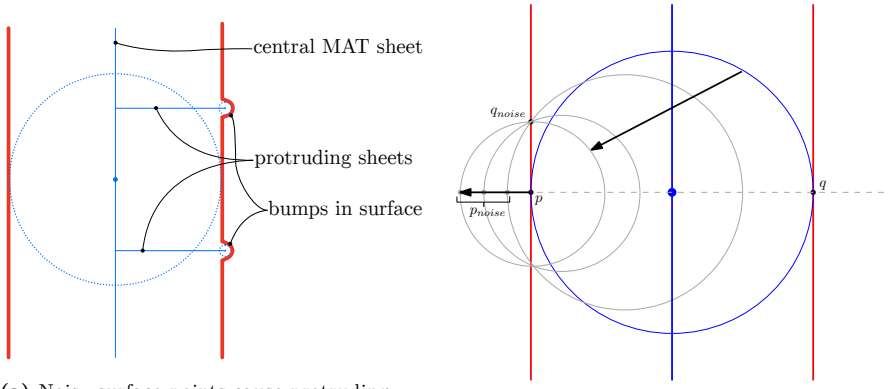


Figure 3.4: Distortion of normal estimation around edges due to non-planar areas and heterogeneous point distribution.

Another consideration is that even though an oriented point cloud often approximates a manifold surface, there is no clear distinction between the inside and the outside of the surface. As a result it is rather difficult to obtain consistent normal orientation throughout the dataset or even within the same object. For the computation of the MAT using the ball-shrinking algorithm this means that we cannot simply expect two well distinguished sets of interior and exterior points merely by flipping the point normals (as done in lines 3-4 of Algorithm 2). Yet with a consistently oriented point cloud—i.e. one obtained from a mesh—it would be as simple as that. This problem is addressed in Chapter 4

Other methods

There are more sophisticated normal estimation methods that specifically deal with the edge problem (e.g. Boulch and Marlet [2012]; Huang et al. [2013]; Li et al. [2010]).However, while these may deliver better normals around edges, they are significantly slower (e.g. Boulch and Marlet [2012] says an order of magnitude slower) and may introduce new problems (e.g. Huang et al. [2013] notice there method fails around open boundaries). It is therefore questionable if there is a net benefit of employing these methods. In this thesis I have therefore decided to use the simple and fast PCA normal estimation. In Section 3.5 I elaborate on the implication on the approximated MAT and show that a usable MAT can in fact still be obtained. As a last remark I should point out that there is some work that indicates that the MAT itself can be used to obtain higher quality normals [Dey and Sun, 2006a]; i.e. start with quickly approximated normals, compute the MAT and refine them by using the properties of the MAT.



(a) Noisy surface points cause protruding sheets.

(b) Scattering of medial ball due to varying degrees of noise in primary feature point.

Figure 3.5: Effects of noise in surface points on MAT.

3.3 Making the ball-shrinking algorithm robust

The original ball-shrinking algorithm of Ma et al. [2012] was designed to handle well-sampled point clouds with very little noise. However, aerial point clouds contain significant noise, and—as explained in Chapter 2—the MAT is by definition highly sensitive to it. The core issue is that small perturbations in the surface points can lead to large perturbations in the MAT. For geographical point clouds this is a real problem because the MAT can get scattered to such a degree that its actual form is no longer perceivable (see for example Figure 3.12). In this section I first describe in detail how the MAT deforms in the presence of noise. Then I present a novel extension to the ball-shrinking algorithm that makes it robust to noise, effectively enhancing the practical usefulness of the MAT of an aerial point cloud.

3.3.1 The problem of noise

The cause of the MAT's sensitivity to noise is that every bump in the surface should by definition have a medial sheet protruding into it. Noisy points cause many small bumps in the surface and each of them can thus be fitted with one or more medial balls that contribute to the medial sheets that protrudes into the bumps. These medial balls are centred close to the surface and away from the more centralised MAT sheets. This is illustrated in Figure 3.5a.

Let us now consider a point $\mathbf{p}$ and its primary medial ball. How do they change

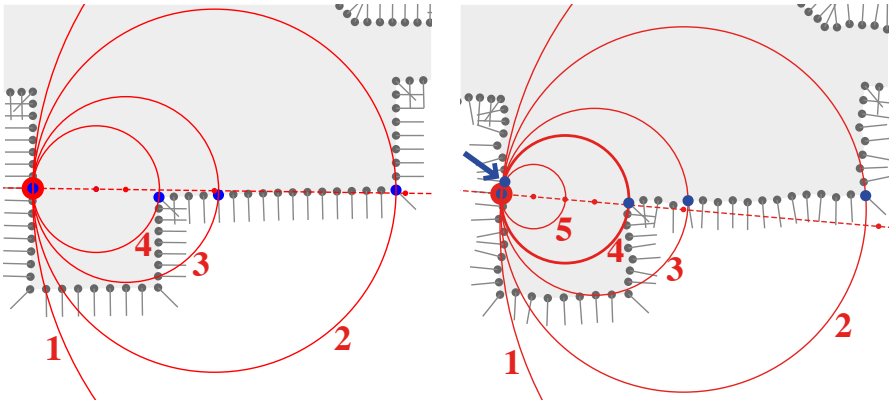
if p is shifted due to noise? See Figure 3.5b. As you can see, the noisy version of $\mathbf{p}$, denoted $\mathbf{p_{noise}}$ has only moved by a small amount. The ball on the other hand has shifted by a much larger amount. One way to explain this is to say that due to the noise, the secondary feature point of the ball, denoted $\mathbf{q_{noise}}$ has actually moved to the surface on the opposite side. Therefore, $\mathbf{q_{noise}}$ is much closer to $\mathbf{p}$ than $\mathbf{q}$, and as a result the ball has become much smaller and is now also centred much closer to $\mathbf{p}$. Thus, even though $\mathbf{p}$ was only shifted by a small amount due to noise, its primary ball changed in four ways: 1) it was shifted by a relatively a large amount, 2) its radius became much smaller, 3) it no longer contributes to the central sheet, and 4) its secondary feature point $\mathbf{q}$ shifted to the opposing surface.

How exactly a medial ball changes depends on the normal at $\mathbf{p}$ and the magnitude of the noise. In Figure 3.5b I show a range of different possibilities under the assumption that the normal of $\mathbf{p}$ does not change due to noise. As the figure shows, this means that B ends up being centred anywhere in between its original position and $\mathbf{p}$. One could say it is effectively scattered away from its original place in the central sheet. In case of significant noise in the surface points—e.g. the geographical case—this means that the entire MAT will be scattered up to the point it is hardly distinguishable.

In literature, the instability of the MAT in presence of noise is a well-discussed topic. Unsurprisingly, many solutions have been proposed (see also Section 2.3.2). Ma et al. [2012] for instance note that the medial ball radii of nearby sample points should be similar. Noisy medial balls typically are an exception to this rule, and can therefore be detected by comparing medial ball radii of nearby surface points. If the ball radius is significantly smaller, the ball is removed. Pruning methods, e.g. based on ball metrics such as the separation distance or separation angle (see Figure 2), can also be effective for detecting and removing noisy samples. However, a common characteristic of the known noise filtering methods, is in fact that they *filter*, i.e. they completely remove the noisy medial balls. While that may be sufficient when the surface point cloud has only very few noisy points, it is a problem for aerial point clouds since the MAT point cloud may be thinned considerably . And, to make it worse, the point density in aerial point clouds is already relativelylow when compared to point clouds obtained from close range laser scanning or mesh sampling. In other words, if one removes all noisy medial balls in a geographical MAT, there will be hardly any medial balls left. Noise filtering methods are therefore not adequate for aerial point clouds.

3.3.2 A novel denoising heuristic

My solution to deal with noisy surface points stands out primarily in the fact that it is not a *filtering* method. Instead of removing noisy medial balls, I aim to

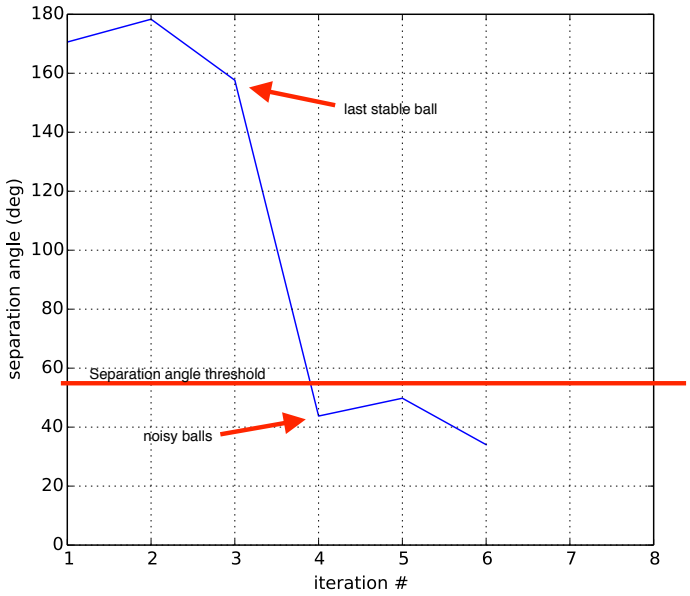


(a) Noise-free case. **(b)** Noisy case.

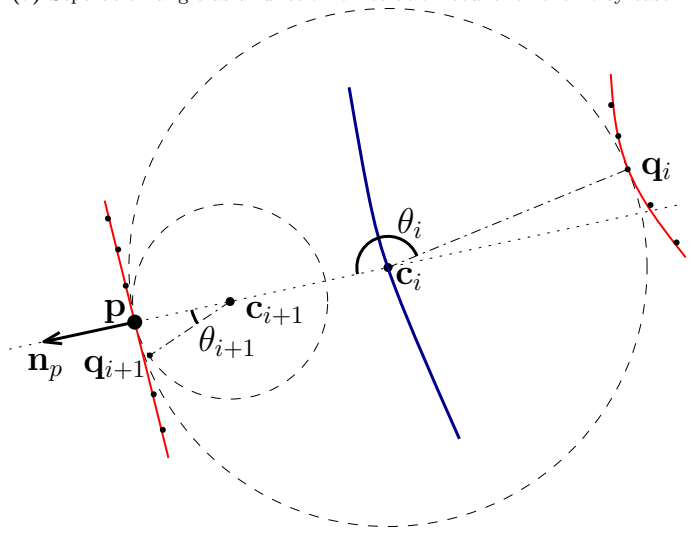
Figure 3.6: Comparison of ball-shrinking iterations with and without noisy surface points.

denoise, i.e. maintain a version of those medial balls that is robust to the effects of noise in the surface. I achieve this by extending the ball-shrinking algorithm. Recall that the algorithm computes a converging series of shrinking balls for each surface point. The key observation that I make, is that even in case of a noisy surface point, a 'good' medial ball is often computed before it shrinks further to become a noisy medial ball. Figure 3.6 illustrates this. If we compare the noisy case (Figure 3.6b) to the noise-free case (Figure 3.6a) we see that an additional ball is computed, i.e. there is an additional 5^{th} iteration. Yet, the ball of the 4^{th} iteration—the final and medial ball for the noise-free case—is very similar in both cases. So even in the noisy case a good medial ball is computed. This observation lies at the core of the denoising heuristics that I propose.

In the case of Figure 3.6b we thus prefer the ball at the 4^{th} iteration over the one at the 5^{th} iteration, because it is much closer—almost identical—to the medial ball that we find in the noise-free case. How do we know which ball to pick if we do not want the final ball? In agreement with what I explained earlier, we can see that the secondary feature point $\mathbf{q}$ shifts to the opposite side of the ball. Moreover, when the secondary feature point $\mathbf{q}$ flips side, the separation angle θ suddenly becomes much smaller (Figure 3.7b). This shift can therefore be detected by monitoring the separation angle as a function of the iteration counter. This is illustrated in Figure 3.7a. The simplest way to detect a noisy ball is thus to set a threshold on the separation angle. The last ball that has a separation angle higher than this threshold should be a good representative medial ball. The extended ball-shrinking algorithm (Algorithm 1) implements two such thresholds on the separation angle, namely θ_0 and θ_1 on lines 10-13. If



(a) Separation angle as a function of iteration count for the noisy case.



(b) Two ball iterations for $\mathbf{p}$. The noise-distorted ball can be detected by the small separation angle $\theta_{\mathbf{i+1}}$.

Figure 3.7: Comparison of ball-shrinking iterations with and without noisy surface points.

the iteration counter i equals 0, θ_0 is used, and else θ_1 is used. In practice, this allows us to better fine-tune the algorithm. Especially for the case that the first ball encountered is already a noisy ball which normally only happens for exterior balls on large planar areas. And in the latter case, which typically happens for interior balls, the threshold can be set lower to achieve a better balance between robustness and sensitivity to small features in the surfaces.

Key benefits of the denoising heuristic

The main property of the denoising heuristic of the extended ball-shrinking algorithm is that is keeps medial balls on the centre sheets. As a result we obtain less noisy balls and a denser MAT approximation. The idea of using the separation angle for noise detection in itself is not new. The novelty lies in how it is integrated in the ball-shrinking algorithm in such a way that noisy points are not removed but denoised instead. While it is a simple extension to the ball-shrinking algorithm, it proves to be very effective in practice, as I will show in the next section.

An additional benefit of the simplicity of the denoising heuristic is that it does not increase the time complexity of the ball-shrinking algorithm, nor does it require an extra pass through the MAT points. In fact, I found that the extended ball-shrinking algorithm runs faster, since the average number of ball iterations per point is lower as a result of the denoising heuristic.

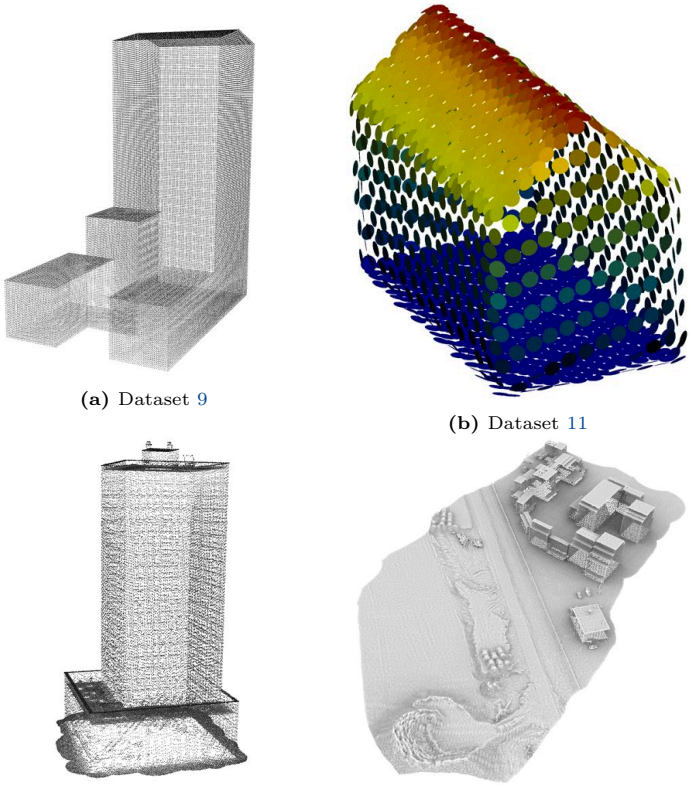
3.3.3 Results

In this section I demonstrate the effectiveness of the novel denoising heuristic that I introduced above. I present experiments both with artificial datasets for which a noise-free ground truth is known and with aerial LiDAR datasets from practice.

Figure 3.8 shows the datasets used in this section. Further details on the different datasets used here and throughout the rest of this thesis are given in Appendix A.

Experiments on artificial data

A noise-free ground truth is important in the evaluation of the effectiveness of the MAT approximation using my novel denoising heuristic. Only with such a ground truth, we can know what the noise-free MAT would look like, and quantify and determine whether the denoising of the MAT actually brings it closer to the noise-free MAT.



(a) Dataset 9

(b) Dataset 11

(c) Dataset 5

(d) Dataset 1

Figure 3.8: Datasets used for experiments

The goals of the experiments in this section are

1. to study the effect of surface noise on the approximated MAT,
2. to study the effect of my novel denoising heuristic on the approximated MAT both in the absence and the presence of noise, and
3. to compare with a conventional noise filtering method.

In particular I will look at 1) the distance between the denoised MAT and the noise-free MAT and 2) the point density of the denoised MAT approximation.

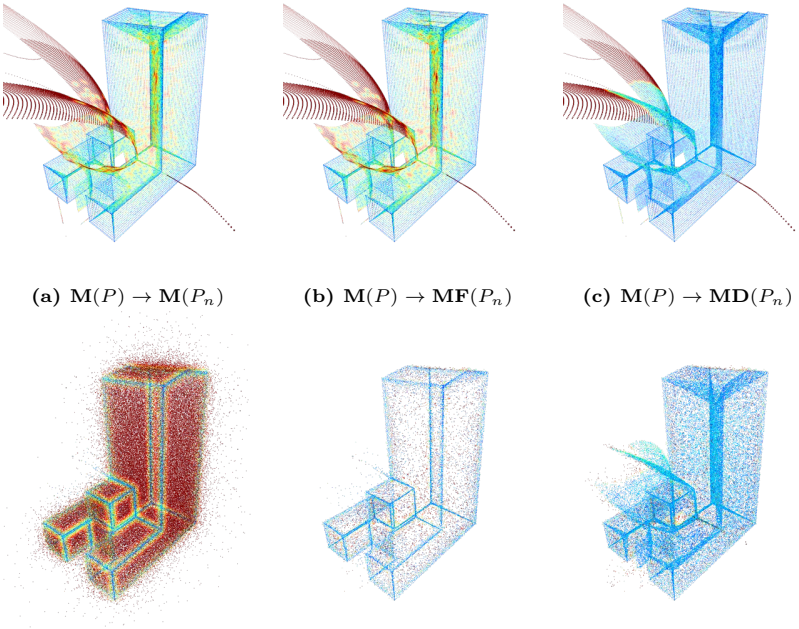
Experiment 1 For the first experiment two variants of `Dataset` 9 are used, one variant with artificially added noise, referred to as P_n, and one variant without artificially added noise, referred to as P. The artificial noise was added using a Gaussian function with a standard deviation of 0.01% of the bounding box diagonal of the dataset.

Then, the MAT of P_n is approximated using three different approaches, namely

1. the unmodified ball-shrinking algorithm denoted $\mathbf{M}(P_n)$ [Ma et al., 2012],
2. the unmodified ball-shrinking algorithm followed by a removal of all medial atoms that have a separation angle $\theta < 30^\circ$ denoted $\mathbf{MF}(P_n)$, and
3. the extended ball-shrinking algorithm as developed in this chapter, denoted $\mathbf{MD}(P_n)$ with parameters $\theta_0 = 30^\circ$ and $\theta_1 = 60^\circ$. This approach is included to serve as a reference to what is commonly done by other researchers to filter noisy medial atoms (see e.g. Amenta and Kolluri [2001]; Attali and Lachaud [2001]; Attali and Montanvert [1996]; Foskey et al. [2003]).

Finally, the MAT of each approach is compared to $\mathbf{M}(P)$, i.e. the MAT of the noise-free point cloud approximated using the unmodified ball-shrinking algorithm. Recall that the MAT approximation that we compute using the ball-shrinking algorithm can be considered a point cloud. Given two point clouds A and B, we can compute for each point in A the distance to the closest point in B. The resulted set of distances is denoted $A \to B$. By studying both $A \to B$ and $B \to A$ one obtains a good impression of the similarity in the spatial distribution of points between point clouds A and B.

In Figure 3.9 I present the distance computations for the three approaches to approximate the MAT. Each column in this figure corresponds to one approach to approximate the MAT. The upper row shows visualisations of the distances from the noise-free MAT $\mathbf{M}(P)$, i.e. the ground truth, to the MAT obtained using one respective approach, and the bottom row gives the opposite distances.



(a) $\mathbf{M}(P) \rightarrow \mathbf{M}(P_n)$ **(b)** $\mathbf{M}(P) \rightarrow \mathbf{MF}(P_n)$ **(c)** $\mathbf{M}(P) \rightarrow \mathbf{MD}(P_n)$

(d) $\mathbf{M}(P_n) \rightarrow \mathbf{M}(P)$ **(e)** $\mathbf{MF}(P_n) \rightarrow \mathbf{M}(P)$ **(f)** $\mathbf{MD}(P_n) \rightarrow \mathbf{M}(P)$

Figure 3.9: Experiment 1: Three approaches to approximate the MAT and the colour coded distances between approximations with and without added noise for `Dataset` 9. Large distances in red, small distances in blue. Each image uses the same colourmap.

Figures 3.9a and 3.9d show the case without denoising heuristic. From Figure 3.9a it is evident that the inner parts of the MAT are badly approximated since the distances $\mathbf{M}(P) \rightarrow \mathbf{M}(P_n)$ are large, indicating a lack of samples there. Many points in $\mathbf{M}(P_n)$ are distributed around the surface points and far away from $\mathbf{M}(P)$ (see Figure 3.9d). This indicates many unstable balls. The outer sheets that are clearly visible in Figure 3.9a, are almost completely missing in 3.9d.

Figures 3.9b and 3.9e show the effect of simple filtering using the separation angle. While this approximation has much less points far away from $\mathbf{M}(P)$ (see Figure 3.9e), it does not solve the problem of a lack of samples on the inner parts of the MAT, as seen from Figure 3.9b.

This lack of samples is no longer apparent when my denoising heuristic is used (Figures 3.9c, 3.9f). From the small distances in these figures, it can be concluded that most of $\mathbf{M}(P)$ is densely approximated by $\mathbf{MD}(P_n)$, including the interior parts of the MAT. The remaining points in $\mathbf{MD}(P_n)$ that are far from $\mathbf{M}(P)$ are distributed close to the surface points and correspond to relatively small medial balls similar to the remaining blue points in Figure 3.9e.

From this experiment it can thus be concluded that $\mathbf{MD}(P_n)$ is 1) significantly closer to $\mathbf{M}(P)$ than $\mathbf{MF}(P_n)$ and 2) a much denser approximation of $\mathbf{M}(P)$ than $\mathbf{MF}(P)$.

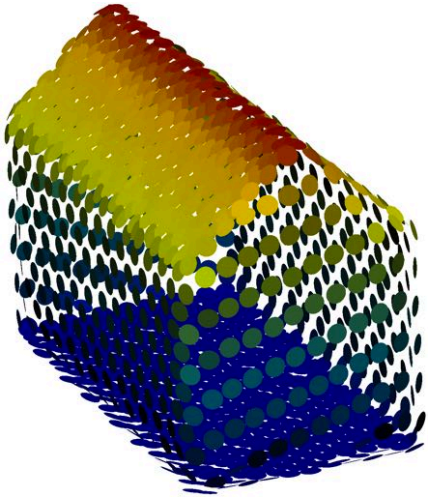
Experiment 2 The aim of this experiment is to study in more detail how the approximated MAT of an aerial LiDAR point cloud is changed, i.e. how medial atoms are moved towards the MAT of the ground truth, as a result of the denoising heuristic. In addition the approximation error of $\mathbf{MD}(P_n)$ with respect to a ground truth is quantified as a function of the denoising parameter θ_1.

Since a ground truth is required an actual aerial LiDAR point cloud is not sufficient. Therefore I use `Dataset 11`: a point cloud that I obtained by simulating an aerial LiDAR scan of a simple building mesh[1] (see Figure 3.10a).

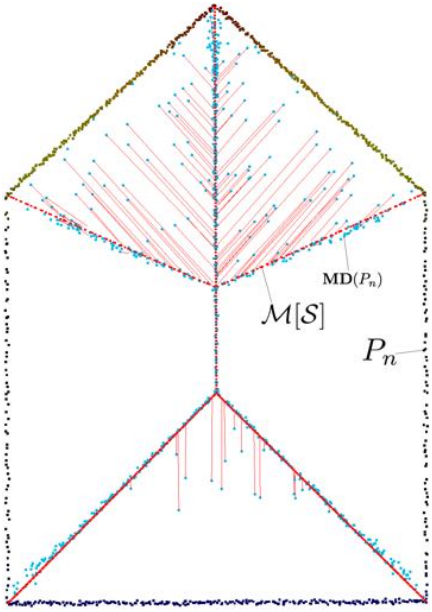
Because this dataset is derived from a mesh model, it is possible to generate a noise-free and very dense MAT approximation. This I consider the reference MAT (e.g. the ground truth) of the experiment, denoted $\mathcal{M}(\mathcal{S})$.

For this experiment two simulated point clouds are considered, namely a simulated point cloud with added noise (P_n) and one without added noise (denoted P). The noise is simulated by adding a normally distributed noise component with a standard deviation of 2 cm along the scanning direction and an additional 2 cm in the position of the scanner.

[1] Using industry-standard flight parameters that were used for the acquisition of the national AHN3 dataset (flight height 400 m, flight line spacing: 400m) and the BlenSor software [Gschwandtner et al., 2011]



(a) `Dataset 11`: a simulated aerial LiDAR point cloud of a building mesh (dimensions: $10 \times 20 \times 15$ m, 1751 points).



(b) Cross section of simulated point cloud with noise (P_n), reference MAT ($\mathcal{M}(\mathcal{S})$), and $\mathbf{MD}(P_n)$. Red lines indicate how points MAT are moved as a result of my denoising heuristic ($\theta_0 = 30^\circ$ and $\theta_1 = 30^\circ$)

Figure 3.10: Experiment 2; dataset and cross-sections

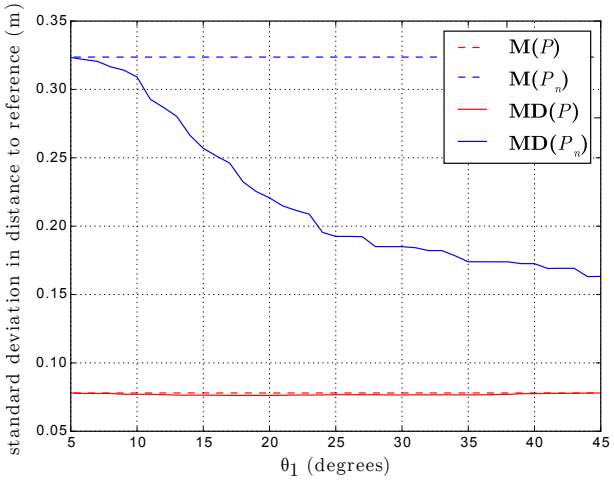


Figure 3.11: Experiment 2: overall error in MAT approximation with respect to reference MAT with and without our denoising heuristics.

For both simulated point clouds the MAT is approximated using once $\mathbf{M}()$ and $\mathbf{MD}()$ for a range of values for θ_1. The case for $\theta_1 = 32^\circ$ is visualised in Figure 3.10b. In this figure lines are drawn between the medial atoms of $\mathbf{M}(P_n)$ and $\mathbf{MD}(P_n)$ that correspond to the same surface points. One can thus see how medial atoms are effectively moved towards $\mathcal{M}(\mathcal{S}]$ as a result of my denoising heuristic.

To quantify the quality of the denoised MAT approximation the distances to the reference MAT are measured for each medial atom, similar to Experiment 1. The plot in Figure 3.11 show how the standard deviation of distances of $\mathbf{MD}(P_n)$ gradually decreases with increasing values of θ_1. For instance, an overall decrease of 31% can be observed in the standard deviation of $\mathcal{MD}(P_n)$ as a result of the denoising heuristic (i.e. a drop from 0.33 to 0.22 at a conservative $\theta_0 = 20^\circ$), while in this case only 11% of the medial atoms are affected by the denoising heuristic.

Furthermore, from the plot of $\mathbf{MD}(P)$, we see that my denoising heuristic has a negligible effect on the approximation of the MAT of the noise-free point cloud P. In other words, it has very little effect on medial atoms that are not affected by noise.

Experiments on real-world data

From the experiments on artificial datasets we can indeed conclude that we can obtain a robust and dense MAT approximation from noisy surface point clouds by using ball-shrinking algorithm with my novel denoising heuristic. Experiments on real data confirm this, as illustrated in e.g. Figure 3.12. As highlighted in this figure, we can observe two distinct effects on the approximated MAT due to the denoising heuristic. First, the approximation of the interior MAT is much denser, i.e. the signal of the medial sheets is significantly stronger as they are much easier to distinguish. Notice that these medial atoms are mostly affected by the θ_1 parameter. Second, the visible scatter of medial atoms exterior to the object is significantly less. These atoms are mostly affected by the θ_0 parameter.

The same behaviour can be observed from `Dataset` 6 (see Figure 3.14). Moreover, Figure 3.13 of the same dataset illustrates very clearly the skeleton-like nature of the MAT of mountain range, where the ridges and valleys of the mountain range are translated to branches of the interior and exterior MAT respectively. However, from Figure 3.14 it can also be observed that some of the minor medial sheets disappear or shrink in the denoised MAT.

Finally, for `Dataset` 1 the effect of the denoising heuristic on the estimated local feature size (LFS, as described in Section 2.2.2) is evaluated (see Figure 3.15). In this figure the LFS is estimated using 1) a MAT approximated using the unmodified ball-shrinking algorithm (Figure 3.15a), 2) the same MAT, but with simple filtering based on the separation angle and separation distance (see Figure 3.15b), and 3) and denoised MAT obtained using my denoising heuristic (see Figure 3.15c).

From Figures 3.15a and 3.15b it is evident that without the denoising heuristic, computing the LFS is not feasible since it the distances from the surface points to the MAT are so heavily distorted by noisy medial atoms. However, with the denser and cleaner denoised MAT approximation, computation of the LFS is feasible. This enables applications such as feature-aware point cloud simplification as described in Chapter 6.

From these experiments it can be concluded that 1) in case of geographical point clouds a much more distinctive and more useful MAT approximation can be obtained by using my novel denoising heuristic and 2) there is a trade-off between the amount of detail captured by the denoised MAT and its robustness to the noise present in the surface point cloud.

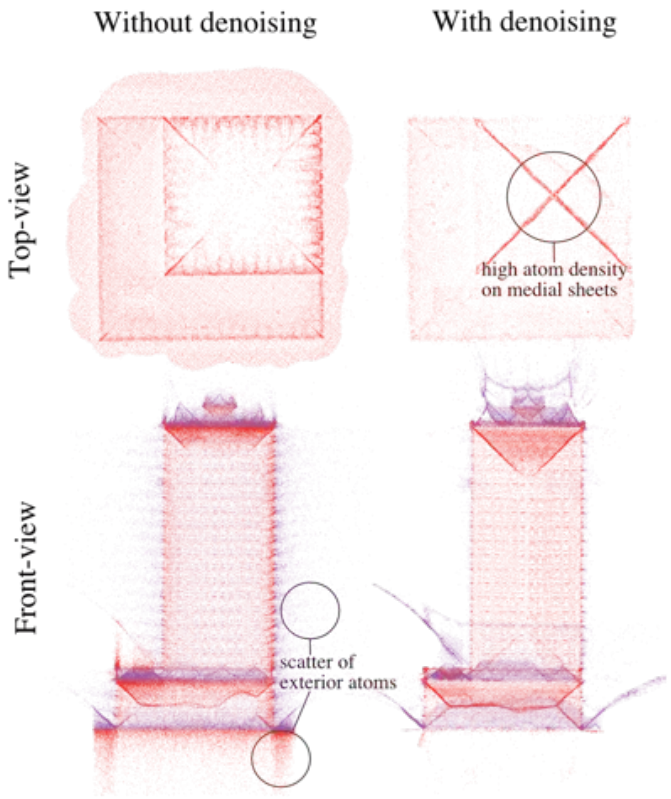
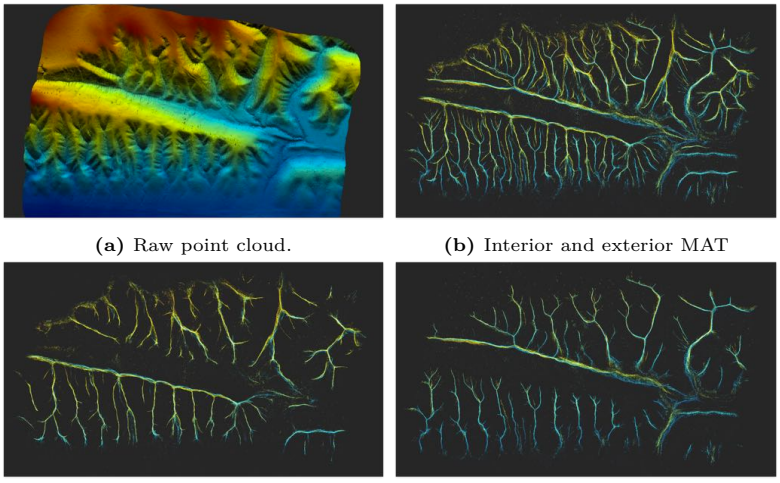


Figure 3.12: MAT approximation for `Dataset` 3 without denoising (left) and with denoising (right) for $\theta_0 = 20°$ and $\theta_1 = 32°$. Shown are top views (top) and side views (bottom). Interior points in red, Exterior points in purple.



(a) Raw point cloud. **(b)** Interior and exterior MAT

(c) Interior MAT **(d)** Exterior MAT

Figure 3.13: The denoised MAT for the `Dataset` 6. Colours indicate elevation.

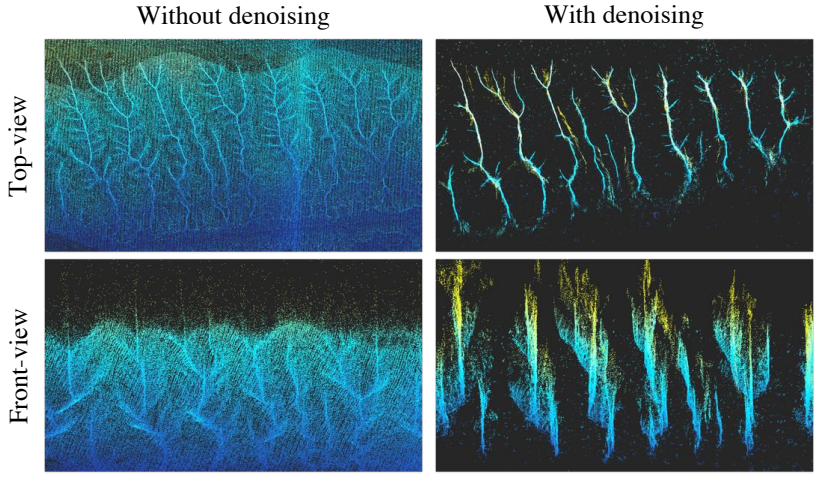
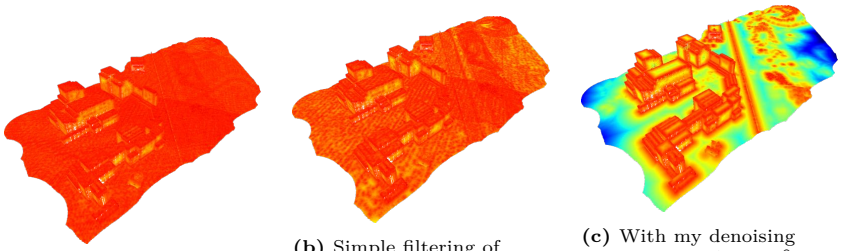


Figure 3.14: `Dataset` 6; exterior MAT approximation with and without novel denoising heuristic. $\theta_0 = 20°$ and $\theta_1 = 32°$



(a) Without denoising.

(b) Simple filtering of atoms with $\theta < 20^\circ$ and $\lambda < 0.05$m

(c) With my denoising heuristic; $\theta_0 = 20^\circ$, $\theta_1 = 32^\circ$ and ball filtering $\lambda < 0.05$m

Figure 3.15: Local feature size approximations for the urban dataset with different approaches to denoise. Red indicates low local feature size, blue high local feature size.

3.4 Impact on computational performance

There are mainly two ways to improve the performance of the shrinking-ball algorithm, i.e.

1. by decreasing the number of required ball iterations per surface point, and
2. by parallelising the ball-shrinking iterations either on a CPU or on a GPU.

The potential speed-ups are over an order of magnitude [Jalba et al., 2012; Ma et al., 2012] in the execution time of the ball-shrinking algorithm, and are significant in practice especially for massive (geographical) point clouds. This leads us to the question: how does the denoising heuristic impact the computational performance of the ball-shrinking algorithm?

A simple and effective way to improve performance is to choose a smaller initial ball radius r_{init}, i.e. one that is close to the final medial ball radius. This effectively reduces the total number of required ball-shrinking iterations for each surface point. Ma et al. [2012] do this by processing the surface point cloud in a spatially coherent order and setting r_{init} to the last found medial ball radius of a nearby point. This approach requires a sequential processing of the surface point cloud and is therefore not suitable for parallel computation. Jalba et al. [2012] propose a CPU parallelisation scheme where each thread processes a chunk of surface points and a GPU parallelisation scheme where each thread processes one surface points. Whenever a new medial ball radius is found in any of the threads, they update a global r_{init} to be a moving average of all medial ball radii found so far.

Unfortunately, while choosing a small r_{init} may improve performance, it also interferes with my denoising heuristic. The problem is that the ball-shrinking

process is not reversible, i.e. we can easily find the next—smaller—ball from the current one, but not the previous—larger—one. This is a problem because, with the denoising heuristic, once a noisy ball is detected, the previous ball should be outputted. However, if the previous ball is not known because it was never computed, it is not possible to output it. Therefore, performance enhancements based on choosing a smaller r_{init} should be avoided in case the denoising heuristic is used.

In principle, the parallelisation schemes of Jalba et al. [2012] are still possible without their initial radius heuristic. The CPU-GPU parallelisation scheme of Ma et al. [2012] does not use such an initial radius heuristic and is therefore also suitable to use with my denoising heuristic. In fact this approach may be preferred, because the lack of initial radius heuristic leads to a large variance in the number of required ball-shrinking iterations for each medial atom which leads to unbalanced threads. That is a problem in GPU computing, and Ma et al. [2012] take special care to balance these threads by communicating with the CPU after each iteration.

In my own implementation of the ball-shrinking algorithm I choose to use a simple CPU parallelisation based on OpenMP, where each thread computes one medial atom.

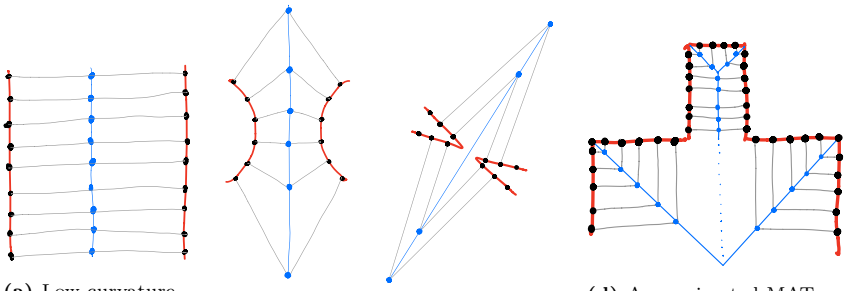
Note that my denoising heuristic by itself also decreases the number of required ball-shrinking iterations, because it is essentially an early break from the ball-shrinking loop.

3.5 Discussion

With the ball-shrinking algorithm one can very efficiently approximate the MAT of a point cloud, and with my denoising heuristic this can be done robustly for geographic point clouds. Yet, very few things come for free, and also in this case there is a small price that needs to be paid for these benefits. In this section I elaborate on limitations of the unstructured MAT, as obtained using the methods described in this chapter. None of these limitations are major bottlenecks, but knowing exactly its limitations can help to optimally make use of the unstructured MAT.

3.5.1 Limitations of the ball-shrinking algorithm for geographical point clouds

Due to the way the ball-shrinking algorithm works there are number of limitations in the way it approximates the MAT.



(a) Low curvature, normal MAT density.

(b) Medium curvature, low MAT density.

(c) High curvature, very low MAT density

(d) Approximated MAT can be split in case of high curvature due to a lack of medial atoms.

Figure 3.16: The effect of surface curvature on the density of approximated MAT sheets. Grey lines indicate spoke vectors.

1. The density of atoms in a medial sheet is proportional with the curvature of the surface point density [Siddiqi and Pizer, 2008]. Areas with high surface curvature lead to a low density of the MAT as illustrated in Figures 3.16a–c. In some cases this can even lead to a separation of sheets as illustrated e.g. in Figure 3.16d. The missing part of the MAT that changes the connectivity of medial sheets is referred to as a ligature in the literature [Siddiqi and Pizer, 2008]. Note that the occurrence of ligatures is not unique to the ball-shrinking algorithm (see e.g. Foskey et al. [2003]).

2. The unstructured MAT does not capture the 2D surfaces of the MAT, only a discretised point approximation is obtained. Also the topology between medial sheets is not captured explicitly. The latter is addressed with the structured MAT that I introduce in the next chapter.

3.5.2 Limitations of the denoising heuristic

The denoising heuristic that I have introduced comes with one notable compromise.

1. As noted in Section 3.3.3 there is trade-off between robustness to noise and the sensitivity to small features in the surface, as is the case with pruning methods for the MAT in general (see Section 2.3.2). This means small features are sometimes detected as noise.

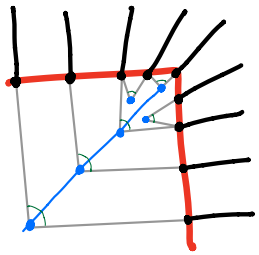


Figure 3.17: Distorted surface normals around edges affect the position and separation angle of medial atoms.

3.5.3 Effects of the quality of the surface point cloud

Finally, the quality of the unstructured MAT is affected by several aspects of the input surface point cloud. Because of their unpredictability and variability even within the same point cloud, I generally find these more significant than the limitations listed earlier in this section. The quality aspects of the input point cloud that affect the quality of the unstructured MAT are:

1. Poor point density in the surface translates to poor point density in the MAT.
2. Missing surface patches due to shadowing effects can cause erroneous medial sheets (see e.g. Figure 7.10b).
3. As shown in Figure 3.17 the normals around an edge are typically estimated in a "smoothened way". As a result the corresponding medial atoms are distorted in terms of position and their medial geometry. In combination with a relatively low point density, this leads to poorly estimated medial atoms towards the edges of a surface.
4. In case of wrong normal orientation (e.g. a normal that point inward instead of outward), the separation between interior and exterior medial atoms will also be wrong. However, this is issue is solved in the structured MAT that is presented in the next chapter.

3.6 Summary

In this chapter I explain in detail how to approximate the MAT for geographical point clouds. In particular, I introduced a method to *robustly* approximate the MAT for geographical point clouds. I call it the unstructured MAT, because it is purely a description of the geometry of the MAT, without structural information.

In the next chapter I will explain how to derive the structure of the MAT, i.e. its decomposition into medial sheets, the topology of the MAT, and its separation into an interior and an exterior part.

4 The structured MAT

In Chapter 3 I explained how to robustly obtain the unstructured MAT from a DSM point cloud using the extended ball-shrinking algorithm. The unstructured MAT is a collection of medial balls, each represented by a point and attributed with a radius and two corresponding feature points (the points where the medial ball touches the surface). This already enables a few useful applications of the MAT such as point cloud simplification and visualisation (see Chapter 6). It is however a purely geometric description that lacks an explicit subdivision into e.g. medial sheets and an explicit representation of the connectivity of medial sheets through e.g. a graph. More advanced applications of the MAT require a richer MAT in which the separate medial sheets are labelled, the connectivity of those sheets is known, and the distinction between the interior and the exterior sheets is known. This is what I call the structured MAT (Figure 4.1), and in this chapter I explain what it is and how to obtain it from the unstructured MAT.

The core scientific contributions of this chapter are

1. I explain the behaviour of the MAT specific to the geographical case,
2. I introduce the idea of using region-growing segmentation based on local medial geometry to segment the unstructured MAT into medial sheets and so-called medial clusters,
3. I explain how to obtain a graph to explicitly express the connectivity of medial sheets, and
4. I solve the problem of the inconsistent separation between the interior and exterior MAT.

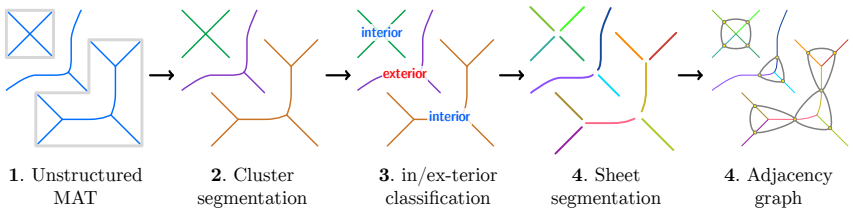


Figure 4.1: The main steps in computing the unstructured MAT.

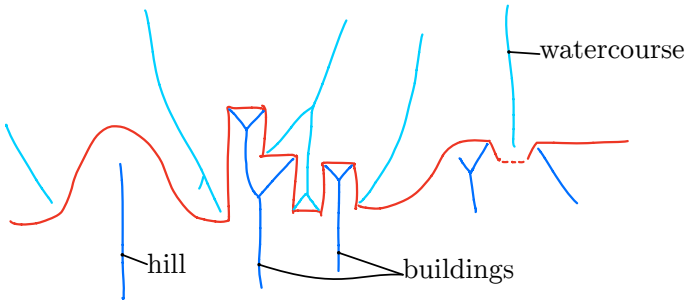


Figure 4.2: For a terrain the MAT is typically subdivided in open clusters that can correspond to distinct features of the terrain. Exterior MAT in light blue, interior MAT in dark blue.

4.1 The structured MAT as a geographic data model

So far, research on the MAT has been almost exclusively focused on shapes that are closed manifolds [Tagliasacchi et al., 2016]. My focus, however, is on the geographical context. Because this is rather different from how shapes are typically considered in mainstream MAT research, I will now elaborate on how this affects the structure of the MAT.

To start with, the geographical MAT is different because—even though we are talking about a single manifold surface—it does not guarantee that the MAT is connected. This is true for both the interior and the exterior MAT. For example, as can be seen from the interior MAT in Figure 4.2, the geometry of the terrain induces a separation of the interior MAT into separate, or nearly separate, parts. I call these parts *medial clusters*. As shown in Chapter 7, medial clusters of a terrain are very helpful in detecting urban features, such as buildings, or certain pronounced features of the natural terrain, such as hills or watercourses.

Second, for the geographical MAT there is a clear added value in maintaining both the interior and exterior MAT. In contrast, mainstream MAT research often exclusively considers the interior MAT (see e.g. Jalba et al. [2012]; Ma et al. [2012]). This is understandable, since the exterior MAT is redundant. One can in theory completely describe a shape using solely its interior MAT. However, the geographic case is different for mainly two reasons. First, some features like the water course in Figure 4.2 can be captured more elegantly in a single exterior medial cluster, rather than two separate interior clusters. And second, considering the relatively poor quality of an aerial point cloud (see Section 1.1), I would argue that the redundancy can actually be helpful. This is especially the case for surfaces with a low point density, where sharp edges may be more accurately represented by the exterior MAT.

Last, I propose a novel way to organise the MAT called the structured MAT. In literature, obtaining the explicit connectivity of the MAT after construction is commonly referred to as *structuration* (also see Chapter 2). Delame et al. [2016] defines structuration as the process of obtaining from the medial atoms the explicit surfaces and curves of the MAT, i.e. as (triangular) meshes and polylines. Their research shows that structuration is quite challenging, even for an MAT approximated from a high quality mesh. The available structuration methods struggle to maintain key structural properties of the MAT, such as that sheets are manifold and that they properly intersect at the junction curves. I therefore opt not to explicitly reconstruct the curves and surfaces from the medial atoms of unstructured MAT. Instead, I choose to maintain the point cloud representation in the structured MAT, representing medial sheets simply as sets of points. The connectivity of the structured MAT is represented as an abstract graph. It is the dual graph of the MAT, where each node represent a medial sheet, and the medial curves are implied by edges that signify adjacencies between sheets. This novel way to organise the MAT, i.e. the structured MAT, is easier and more robust to compute, without loosing its usefulness for advanced MAT applications such as object detection and classification (see Chapter 7).

In summary, and as illustrated in Figure 4.1, the main features of the structured MAT are

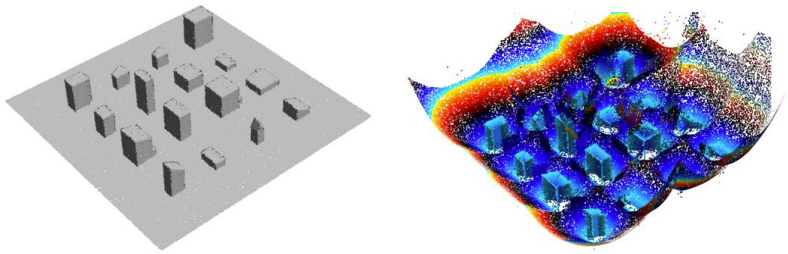
1. segmentation of the MAT into medial clusters,
2. segmentation of the MAT into medial sheets,
3. explicit connectivity of medial sheets through a graph, and
4. distinction between interior and exterior parts of the MAT.

In the following sections I introduce the algorithms to compute the structured MAT. Naturally I have implemented these algorithms (see Chapter 5 for more details), and throughout this chapter I show the results of the algorithms on `Dataset` 10. Figure 4.3 shows the surface points and the medial points of the unstructured MAT for this dataset.

4.2 Segmenting the unstructured MAT

An important feature of the structured MAT is its organisation into its constituent elements, e.g. medial clusters and medial sheets. In this section I describe a novel approach to identify medial clusters and sheets in the unstructured MAT using a segmentation algorithm.

For the segmentation of the unstructured MAT I use the `growRegion` algorithm (see Algorithm 3) that is conceptually similar to the one described by Rabbani et al. [2006]. However, I have based the segmentation on the local medial



(a) Surface points.

(b) Medial points coloured by medial radius using a repeating colourmap.

Figure 4.3: `Dataset` 10

atom geometry, instead of local surface geometry (i.e. Rabbani et al. [2006] use the difference in normal vectors of surface points). In the algorithm, each region (or segment or sheet) is started from an arbitrary seed atom and gradually grows by considering neighbouring atoms. The neighbours are tested using `validCandidate` and the region is complete when all of its neighbours are tested.

4.2.1 Segmentation into medial clusters

The goal of segmentation into medial clusters is to decompose the unstructured MAT into disjoint parts. This means that the medial balls of different medial clusters do not intersect (see Figure 4.4a). The union of all medial balls within one cluster, however, should be a connected space. This leads to the following lemma.

Lemma 4.1. *The medial balls of nearby atoms in the same medial cluster intersect.*

This lemma forms the basis of the region growing condition that I use for cluster segmentation.

To quantify the intersection between two medial balls I use a simple fraction as illustrated in Figure 4.4b and given below.

$$o = \frac{r_1 + r_2}{d}$$

The distance between the two medial atoms is denoted d, and the radii of the two respective medial balls are denoted r_1 and r_2. If and only if $o > 1$ there is

Algorithm 3: The `growRegion` algorithm.

Input : S; a set of medial atoms, i.e. the structured MAT
`validCandidate`(a, b); a function that tests whether medial atoms a and b should be in the same medial sheet
Output: Each medial atom is enriched with a segment label

```
1  i ← 0
2  while S ≠ ∅ do
3      segment(S) ← 0
4      s ← pop(S)
5      C ← a stack initialised with s
6      segment(s) ← i
7      while C ≠ ∅ do
8          c ← pop(C)
9          foreach n in nearestNeighbours(c) do
10             if validCandidate(c, n) then
11                 push n to C
12                 segment(n) ← i
13                 S ← S − n
14         i ← i + 1
```

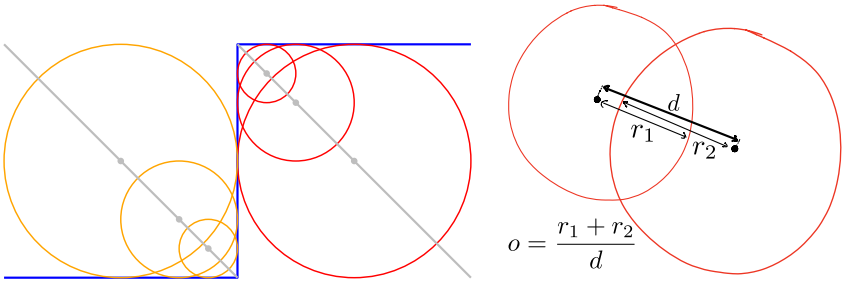
an intersection between the medial balls in theory, which then implies that they belong to the same cluster.

In practice it proves to be useful to test for $o > t_{overlap}$ in the `validCandidate` function of Algorithm 3, where $t_{overlap} > 1$ is a user-defined parameter. The higher the value of $t_{overlap}$ the bigger the required intersection, thus the more robust the resulting segmentation is to potentially noisy medial atoms.

Figure 4.5 shows the cluster segmentation for `Dataset` 10. All exterior sheets are all assigned to one big cluster and all building structures have clearly been assigned their own medial clusters. In Section 7.2 I present more results on different real-world datasets.

4.2.2 Segmentation into medial sheets

This section describes a novel method to decompose the unstructured MAT into separate medial sheets based on the local medial geometry of the medial atoms. Notice that Kustra et al. [2014] and Kustra et al. [2016] also perform sheet segmentation based on a medial point cloud, respectively using patch-based surface reconstruction and the medial geodesic function (similar to the boundary potential that is described in Section 2.3.2). Unfortunately these methods are

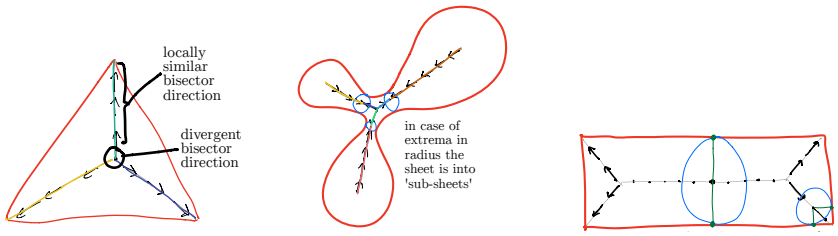


(a) Interior and exterior parts of the MAT have no intersecting medial balls.

(b) intersection expressed as the ratio of the summed ball radii over the distance between the balls.

Figure 4.4: The intersection between medial balls as a region growing condition.

Figure 4.5: Medial cluster segmentation with $t_{overlap} = 4$.



(a) Bisectors induce segmentation into medial sheets.

(b) Bisectors induce further segmentation into sub-sheets

(c) In the parallel plane case the bisector is unstable.

Figure 4.6: Behaviour of the medial bisector in different cases.

not suitable for the geographic case, because Kustra et al. [2014] requires a very dense surface point cloud, and Kustra et al. [2016] even requires a surface mesh and is computationally highly expensive especially for large inputs.

I will now explain what properties of the local atom geometry are best suited for the sheet segmentation, and how to employ those in a region-based segmentation algorithm. The medial bisector proves to be effective for sheet segmentation. It is based on the following lemma, that is true with the notable exception of two special cases that I will treat later.

Lemma 4.2. *Nearby medial atoms in medial sheet have similar medial bisectors.*

Recall from Chapter 2 that each medial ball is characterised by a medial bisector; the vector that bisects the spoke vectors and is always tangent to the medial sheet. Figure 4.6a illustrates the two main reasons why the medial bisector is effective for sheet segmentation. First, the medial bisector for nearby atoms on the same medial sheet is typically similar. This is true because the the medial bisector points in the direction of decreasing ball radius along a medial sheet. Second, the medial bisector is different for atoms of different sheets around a junction curve. This is true because the different sheets that meet at a junction curve are not overlapping and the bisector is always tangent to its medial sheet. Thus, the medial bisector is effective for sheet segmentation because we can use it to 1) identify medial atoms that are part of the same sheet, and 2) to distinguish between different sheets at the junction curves. The latter is especially important because it means that at the curves where different sheets meet we are able to achieve a clear separation and we prevent under-segmentation.

For sheet segmentation `validCandidate`(a, b) should initially be a comparison between the medial bisectors of atoms a and b. The test succeeds if the angle between the medial bisectors is below a user defined threshold t_{bisec}. What is

a good value for t_{bisec} depends on the atom density and the amount of noise in the medial sheet.

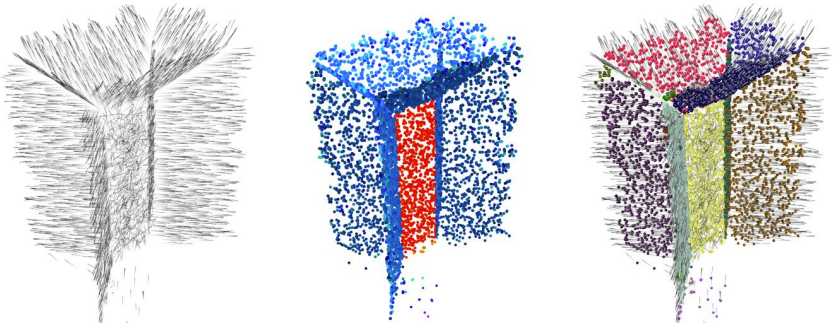
Special cases

Notice that the geometry of the surface may be such that the bisectors are not locally similar—or continuous—within one sheet. This is illustrated in Figure 4.6b. It occurs when there is a local extrema in the radius field of the sheet. In the figure there are minima in the radius field on the sheets and the medial bisectors point towards this minima from both sides, introducing a local singularity in the radius field. The effect on the segmentation is that those sheets may be split at the singularity. With the 3D MAT that only happens when there is no path of locally similar radii around the singularity, i.e. when it stretches from one sheet boundary to another. In my experience this is not a problem in practice, in fact one could consider these separate sheets split by a 2-junction curve, similar to the 3-junction (or Y-intersection) curves described in Section 2.2.3.

There is one other notable case where the medial bisector is not effective for sheet segmentation. This is depicted in Figure 4.6c. Here the surface comprises of two parallel planes, which often happens in building structures. Then, the medial ball radius is equal and the separations angle is 180° throughout the sheet. In this case the medial bisector is unstable because there is no direction of decreasing radius. The bisector direction, albeit still tangent to the sheet, is therefore arbitrary and not reliable for sheet segmentation. See e.g. the bisectors for the central sheet in Figure 4.7a.

Fortunately, the parallel plane case is not a problem in practice. It can be easily detected by looking at the separation angle, because in this case it must be equal to 180° by definition, see e.g. Figure 4.7b. In addition, because of the arbitrary bisector directions, the bisector-based `growRegion` algorithm will not find any significant regions there. A significant region in this case means a region that contains more than a few, e.g. $1 - 5$, atoms. As a result, the medial atoms on these sheets can easily be detected and be given the 'unsegmented' label. After that, they can be segmented in a second run of the `growRegion` algorithm with a test based on the separation angle. Figure 4.7c shows the final result.

The full sheet segmentation algorithm thus includes two runs of the `growRegion` algorithm is given in Algorithm 4. As can be seen on lines 1 and 5, the `validCandidate` function is defined differently for each run. First it is based on the angular difference in medial bisector and then it is defined based on the separation angle. Furthermore, the second call of `growRegion` only runs on the set of medial atoms that are unsegmented and have a separation angle that is approximately straight. In this way, it already segmented atoms and otherwise irrelevant atoms are not affected.



(a) Medial bisectors only.

(b) Medial points coloured by separation angle. The red colour indicates a value close to 180°.

(c) Medial sheets and medial bisectors.

Figure 4.7: Medial sheet segmentation based on medial bisector and separation angle.

Figure 4.8a shows the result for `Dataset` 10. From this figure and especially Figure 4.7c it can be observed that the separation of medial sheets is very good where different medial sheets meet.

4.3 Finding the connectivity of the medial sheets

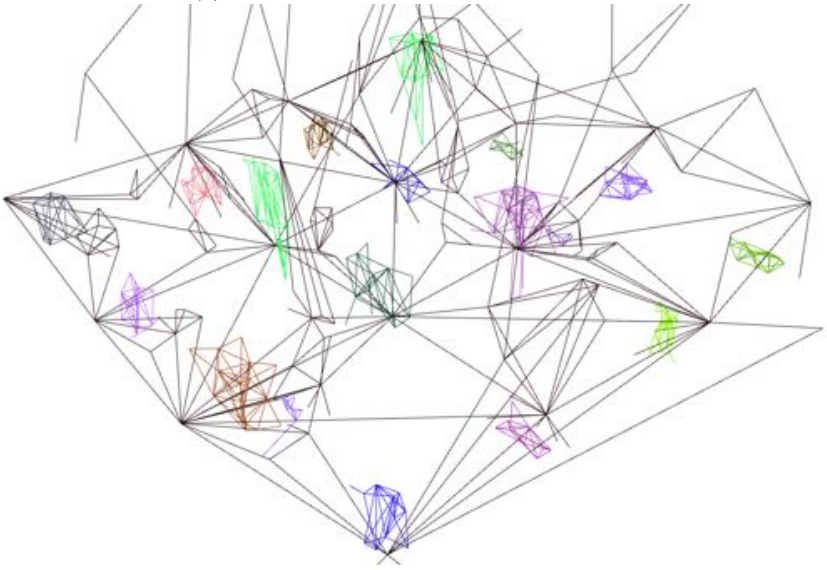
After sheet segmentation the next step towards the structured MAT is to capture the connectivity of the sheets in a graph. The connectivity of medial sheets is an important feature of the MAT. And having this connectivity explicit in a graph data-structure is essential, because it allows us to analyse and manipulate the structured MAT using common graph algorithms. As will be described in Chapter 7, this enables key applications such as object detection and reconstruction. There are two graphs that I construct here:

Adjacency graph In this graph the nodes are sheets, and the edges represent adjacencies between sheets.

Flip graph In this graph the nodes are sheets, and each edge relates an interior sheet to an exterior sheet if they have feature points on the surface in common.

They are illustrated in Figure 4.9. Using these two graphs one can easily 1) navigate inside interior or exterior parts of the MAT and 2) navigate from an interior sheet to a corresponding exterior sheet.

(a) Medial sheet segmentation with $t_{bisec} = 8°$.



(b) A connected component analysis of the sheet adjacency graph.

Figure 4.8: Medial sheet segmentation and corresponding sheet adjacency graphs.

Algorithm 4: The `segmentSheet` algorithm.

Input : S; a set containing all medial atoms that need to be segmented
$t_{mincount}$; a threshold indicating minimum atom count for a region
$t_{straight}$; a tolerance for straight angles
t_{bisec}; a threshold for the angular difference in medial bisectors
t_{θ}; a threshold for separation angle difference

Output: a segmentation of S

1 `validCandidate`$(a, b) \leftarrow \angle$ bisector(a) bisector$(b) < t_{bisec}$
2 `growRegion`(S, `validCandidate`)
3 remove all segments that contain fewer than $t_{mincount}$ atoms
4 $M \leftarrow$ all unsegmented atoms a from S with a straight separation angle, i.e. separation$(a) > t_{straight}$
5 `validCandidate`$(a, b) \leftarrow |$ separation$(a)-$separation(b) $| < t_{\theta}$
6 `growRegion`(M, `validCandidate`)

Finding the adjacency graph

Algorithm 5 lists the `adjacency` algorithm that I developed to find what sheets are adjacent. The adjacency graph is comparable to the topology graphs between planar segments in the surface point cloud computed by e.g. Schnabel et al. [2008], Elberink and Vosselman [2009] and Verma et al. [2006] As all previous algorithms that I presented, the `adjacency` algorithm relies on k-neighbourhoods. The output of the algorithm is an adjacency list of sheet-pairs with an associated value that indicates the strength of the adjacency between those sheets. This output is computed by counting for each atom the numbers of neighbours that belong to another sheet (see Figure 4.10a). These counts are aggregated for each unique sheet pair. Generally speaking, a higher count corresponds to a higher likelihood that a pair of sheets is indeed adjacent.

Figure 4.11a shows the adjacency graph for a single medial cluster as computed by the `adjacency` algorithm. Furthermore, medial clusters can also be obtained from the adjacency graph by means of a connected component analysis (e.g. as described by Hopcroft and Tarjan [1973]). This gives a set of sub-graphs whose nodes are connected to any other nodes in the adjacency graph, or in terms of the MAT: it gives groups of medial sheets that are not adjacent to any other sheets in the MAT; i.e. the medial clusters. Figure 4.8b illustrates this But, note that this is a computationally more expensive way to obtain medial clusters when compared to the cluster segmentation described in Section 4.2.1.

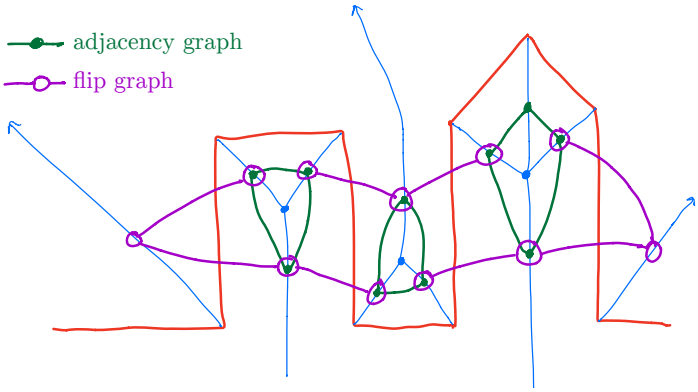
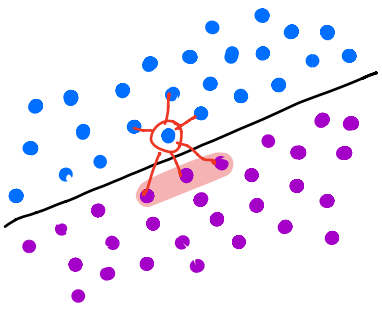
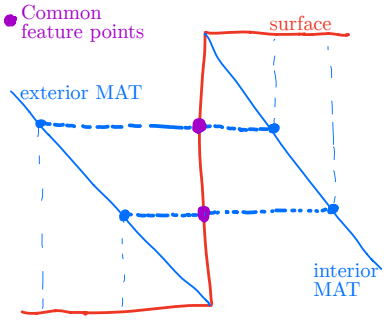


Figure 4.9: Adjacency and flip graphs



(a) Adjacency between sheets are found by counting atom neighbours.



(b) The flip graph is found by counting common feature points between different sheets.

Figure 4.10: Effects of noise in surface points on MAT.

Algorithm 5: The `adjacency` algorithm.

Input : S; the list of medial atoms
Output: A; an associative array with as *keys* tuples of two sheet and as *values* the number of connecting neighbours

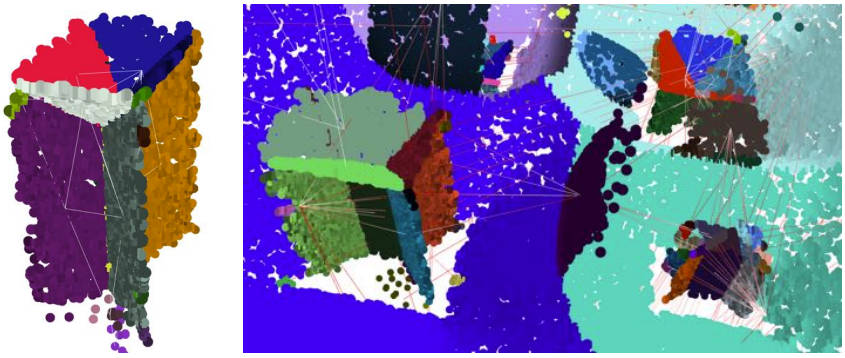
```
1  A ← an empty associative array
2  foreach atom a in S do
3      foreach atom n in nearestNeighbours(a) do
4          if segment(a) ≠ segment(n) then
5              if segment(a) < segment(n) then
6                  l ← ⟨a, n⟩
7              else
8                  l ← ⟨n, a⟩
9          if l[0] = 0 then
10             go to next iteration
11         if l is not a key of A then
12             A[l] ← 1
13         else
14             increment A[l] with 1
```

Finding the flip graph

The flip graph (recall Figure 4.9) is computed by counting common feature points between medial sheets. The `flip` algorithm (see Algorithm 6) computes the flip graph as an associative array by iterating over the medial atom pairs of the surface points. As illustrated in Figure 4.10b, these pairs consist of the two atoms that were generated from the same surface point (i.e. by flipping the normal vector, as described in Section 3.1). Figure 4.11b shows the flip graph as computed by the `flip` algorithm.

Time complexity

The `adjacency` and `flip` algorithms are both implemented using an associative array (also called dictionary or (hash) map) that, provided the right implementation, has constant-time lookup. This brings the expected running time of both algorithms to $O(n)$, where n is the number of atoms in the MAT, assuming that the nearest neighbours for each medial atom are precomputed (for the `adjacency` algorithm).



(a) Adjacency graph for a single medial cluster.

(b) Flip graph.

Figure 4.11: Connectivity of medial sheets in `Dataset` 10. Graph nodes are drawn on the average coordinates of each medial sheet.

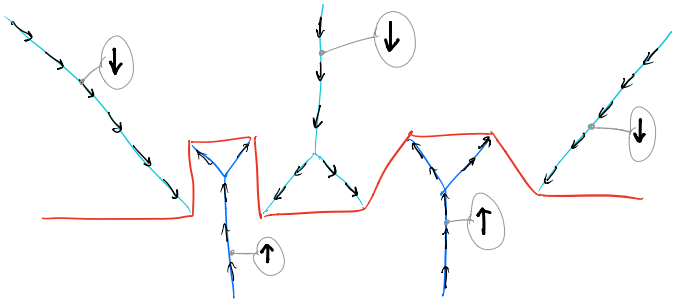


Figure 4.12: The z-components of medial bisectors can be used to determine whether a medial cluster is interior (direction up) or exterior(direction down).

Algorithm 6: The `flip` algorithm.

Input : S; the list of medial atoms
Output: A; an associative array with as *keys* tuples of two sheet and as *values* the number of common feature points

```
1  A ← an empty associative array
2  foreach pair of atoms a, b in S that correspond to the same surface
     point do
3      if segment(a) ≠ segment(b) then
4          if segment(a) < segment(b) then
5              l ← ⟨a, b⟩
6          else
7              l ← ⟨b, a⟩
8      if l[0] = 0 then
9          go to next iteration
10     if l is not a key of A then
11         A[l] ← 1
12     else
13         increment A[l] with 1
```

4.4 Interior and exterior classification of MAT clusters

It is very helpful to know whether a medial cluster is part of the interior or the exterior MAT. As explained in Section 4.1 it can help in finding certain types of features in the terrain, e.g. watercourse are easier to find with exterior clusters as shown in Chapter 7, while buildings are more conveniently described by their interior cluster.

Due to the problem of inconsistent normals (see Section 3.2), we can not get the distinction between interior and exterior medial atoms directly from the unstructured MAT. However, under the assumption that the boundary surface is manifold, we know that an entire medial cluster should be either interior or exterior. This raises the following question: can we determine whether a medial cluster is interior or exterior based on the properties of its medial atoms?

As shown by Figure 4.12, in case of a terrain the sheets of the MAT grow into the features of the surface either from above, or from below. One could say that the exterior MAT comes from the sky above, whereas the interior MAT comes from the earth below, simply because the manifold surface representing the terrain has an above and an underneath. We can in fact quantify this by looking—once again—at the medial bisectors of the medial atoms in each cluster. In case of

an exterior cluster, the medial bisector should predominately point downwards, and vice versa for an interior cluster.

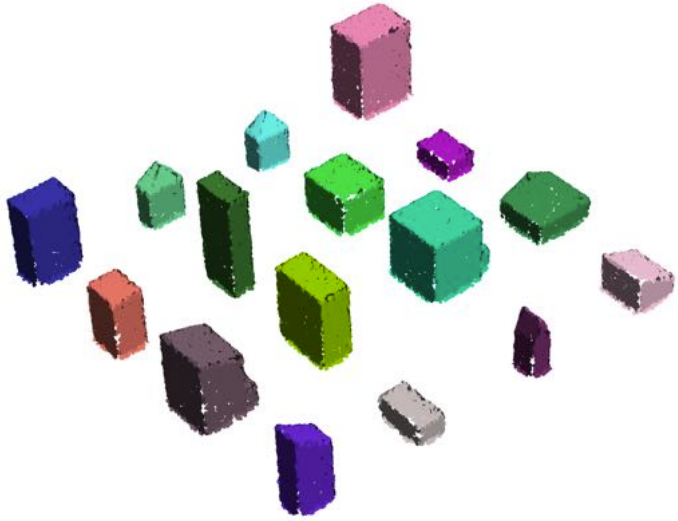
Therefore, if we analyse the medial bisectors of a cluster, i.e. by looking at the average z-component of all bisector in the cluster denoted as B_z, we can say whether it is probably interior or probably exterior. Thus, if $B_z > 0$ a cluster is internal and if $B_z < 0$ a cluster is external. Figure 4.13 shows the resulting interior clusters for `Dataset` 10, as well as the corresponding surface points. In Section 7.2 I present more results on different real-world datasets.

4.5 Summary

In this chapter I have proposed methods to enrich the unstructered MAT with structural and topological information. In the resulting structured MAT the decomposition into individual medial sheets is known. In addition there is explicit information on their connectivity and their organisation into so-called medial clusters. Also the separation of the MAT into an exterior and interior part is made explicit, something that was not directly apparent from the unstructured MAT due to inconsistent point normal orientations.

The following chapter offers details on how to practically implement the methods proposed in this chapter and the previous one.

(a) Interior medial clusters.



(b) Surface points corresponding to interior medial clusters.

Figure 4.13: Interior MAT part in `Dataset` 10.

5 Implementation and data-structures

In the last two chapters I explained how to make the MAT useful for geographical point cloud modelling by computing the unstructured and the structured MAT. In this chapter I elaborate on the practical aspects of the implementation of these methods. In particular I will look at 1) the prototype software that I developed for this thesis, 2) the data-structures I used and developed to represent both the structured and the unstructured MAT, and 3) how to scale the computation of the unstructured MAT for surface point clouds that are too big to store in a computer's main memory.

5.1 Prototype implementation

Over the course of this PhD research, I have developed a number of software packages to prototype my MAT-based algorithms. The most notable ones are:

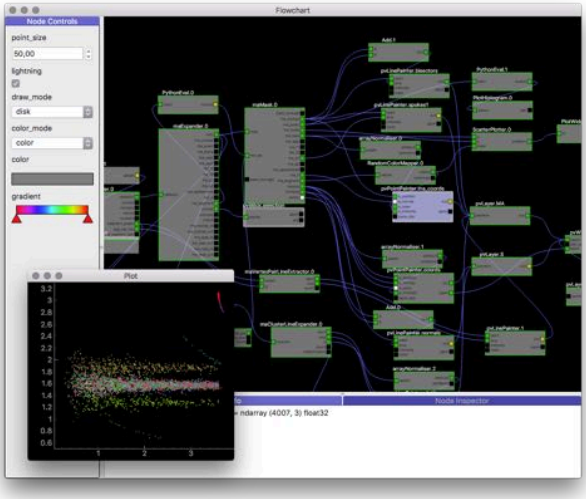
`masbcpp` is a set of command line utilities programmed in C++ to compute the unstructured MAT[1]. It has support for parallel computing with OpenMP[2] and includes:

- a tool for normal estimation based on local plane fitting (an obligatory pre-processing step for the ball-shrinking algorithm),
- an implementation of the ball-shrinking algorithm of Ma et al. [2012] with my novel denoising heuristic to compute the unstructured MAT as described in Chapter 3, and
- a tool to perform MAT-based point cloud simplification based on the method described in Section 6.1.2.

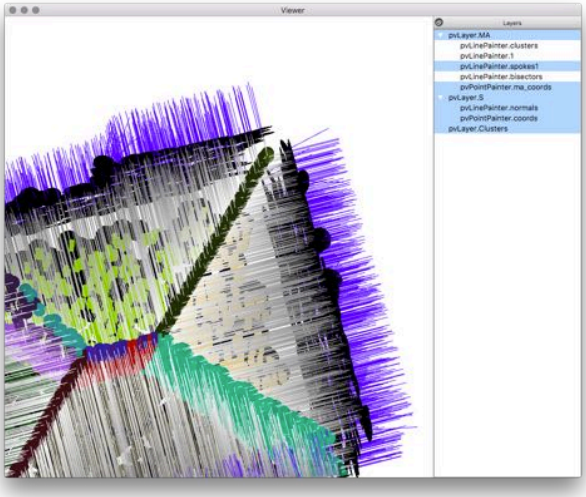
The company Safe Software contributed to the development of masbcpp and included it in their FME 2017.1[3] software as the `PointCloudSimplifier` transformer. FME is widely used program for geographical data integration.

`skel3d` is a Python library for working with the unstructured MAT[4], e.g. to

[1] https://github.com/tudelft3d/masbcpp
[2] http://www.openmp.org
[3] https://www.safe.com/fme/fme-desktop/
[4] https://github.com/Ylannl/skel3d



(a) Flowchart and plot windows. A scatter plot of medial radius versus separation angle is shown for the medial cluster visualised in (b).



(b) 3D viewer window. Medial points and spoke vectors are visualised as well as the surface points and their normal vectors.

Figure 5.1: Screen shots of `pyvi`; the interactive flowchart and 3D viewer software I developed.

calculate properties of the medial geometry, and methods to compute the structured MAT, e.g. the segmentation and graph methods, as introduced in Chapter 4.

`pyvi` is an interactive flowchart and visualisation tool[5]. Using the flowchart any possible processing pipeline can be quickly set up, debugged, evaluated and saved for later use. It is implemented using Python and OpenGL[6].

These three packages are open source and depending on open source libraries. Their source code can be found through the links in the footnotes.

The general workflow for using the MAT for geographical point cloud modelling is to first obtain the unstructured MAT using the `masbcpp` tool, then possibly do further processing such as segmentation of the MAT using `skel3d`, followed by a visualisation using the `pyvi` tool. Figure 5.1 showcases the user interface of `pyvi`. With the interactive flowchart shown in Figure 5.1a I can quickly test different processing pipelines. After a change in the parameters of a processing node or the connections between the processing nodes the rest of the processing pipeline and the resulting visualisations are immediately updated. Many processing nodes directly use functions of `skel3d`. This allows for much more convenient and specialised development than possible with other visualisation software.

5.2 Data-structure for the unstructured MAT

Ma et al. [2012] and Jalba et al. [2012] do not explicitly describe any data-structure for efficiently storing the unstructured MAT. Therefore, I describe a simple and efficient data-structure to store the unstructured MAT as computed by the ball-shrinking algorithm described in Chapter 3. ct that geographical point clouds are typically very large in storage and that the unstructured MAT has twice as many medial atoms as input points, the goal is to use as little storage space as possible without making a sacrifice in the functionality of the medial atoms. This means that for each medial atom we should be able to retrieve in constant time

1. the coordinates of the medial point,
2. the radius of the medial ball,
3. the coordinates of the two corresponding feature points in the surface point cloud,
4. the medial bisector and the separation angle.

[5] https://github.com/Ylannl/pyvi
[6] https://www.opengl.org

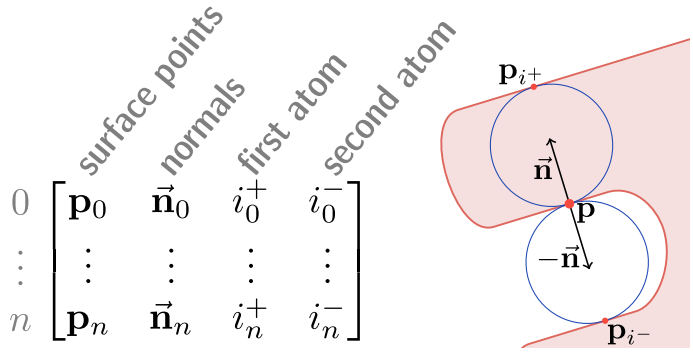


Figure 5.2: Compact representation of the unstructured MAT by extending the surface point cloud array. Each record contains an oriented surface point (and primary feature point) and indices to two secondary feature points, one for each normal orientation. With this information the complete local medial geometry can be reconstructed in constant time.

Some of these properties are redundant, e.g. one can compute the medial radius from the medial point and a feature point, or the medial point can be computed from the primary feature point, its normal vector and the medial radius. Here I aim to have minimal redundancy without sacrificing any functionality.

The unstructured MAT is a collection of medial atoms and each atom can be considered a (medial) point with a number of attributes. The unstructured MAT can thus be seen as a separate point cloud. However, in case of the ball-shrinking algorithm each surface point generates exactly two medial atoms; one in the positive normal direction and one in the negative normal direction. As a result each medial atom is uniquely related to a surface point, i.e. its primary feature point, in the surface point cloud. And, inversely, each surface point is the primary feature point of two medial atoms As I will explain in the following, this means that the coordinates of the medial points do not need to be stored explicitly, and it is in fact possible to store the entire unstructured MAT by adding just two attributes to the surface points cloud.

A point cloud is typically stored as an array of records, where each point corresponds to one record (see e.g. LAS [2013]). Each record is a collection of a fixed number of fields in a particular order. In the case of an oriented point cloud the fields of the record are the three coordinates of the point and the three components of the corresponding normal vector. Additional point attributes can be added by adding additional fields to the record.

Figure 5.2 illustrates how the array of the oriented surface point cloud can be extended to incorporate the medial atoms that form the unstructured MAT. Two

fields are added that are indices that point to other surface points in the array. Each of these indices implies one medial atom. Both medial atoms have the surface point of that record as the primary feature point. The secondary feature point is referenced by the two respective indices i^+ and i^-. In combination with the point normal in the record, it is now possible to derive the complete local atom geometry, most notably the medial radius r and the medial point $\mathbf{c}$ by using the constant time `computeRadius` and `computeCentre` functions as given in Figure 3.3.

5.3 Computation of the unstructured MAT for very large datasets

Geographical point clouds can be massive (e.g. contain billions of points) and not fit a computer's main memory, consequently in-core computation of the unstructured MAT can be problematic. Since this has not been researched before, Lam [2016] investigated a number of approaches to circumvent this problem by not loading all surface points at once into main memory (out-of-core). His preliminary conclusion is that a simple tiling scheme is most effective.

Here I propose an out-of-core tiling scheme for the ball-shrinking algorithm that processes the surface point cloud sequentially in small chunks that each fit main memory. This is possible because the only global operation performed by the ball-shrinking, i.e. the nearest neighbour query, is actually bounded by the initial ball radius r_{init} which is a user-defined parameter. Only medial balls with a radius up to r_{init} are constructed, and the radius of the largest medial ball of an object typically depends on the approximate size of that object. A sensible choice for the value of r_{init} is therefore an approximate size of the largest object in the input. For example, for urban datasets with structures that are up to 200 m in size, a value for r_{init} of approximately 100 m should suffice. This does mean that medial atoms with a radius larger than 100 m can no longer be constructed, but using this approach it is possible to spatially subdivide and process a massive dataset with a limited amount of main memory.

To partition, I subdivide the dataset into square tiles of fixed dimensions. The tile-size is chosen such that the contained points easily fit in main memory. Additionally, every tile is buffered with the value of r_{init} (see Figure 5.3). Then, the tiles are processed one by one. First I compute a kd-tree for the complete tile (including the buffer) to speed up nearest neighbour queries. Then I approximate the point normals and the MAT itself for the points that are inside the tile but not in the buffer region. With this simple out-of-core scheme I can obtain identical outputs compared to the regular in-core approach.

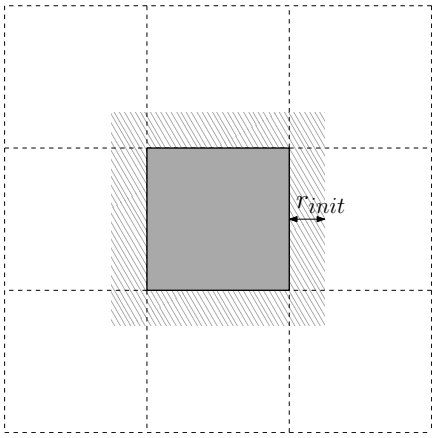


Figure 5.3: Simple tiling scheme. Each tile is buffered with the initial ball radius r_{init}.

Figure 5.4 shows the memory usage of my simple partitioning scheme during the MAT computation of a 1.3 billion point dataset (apart from the pointcount similar to `Dataset 1`). I must note that this experiment was performed using an early implementation of the extended ball-shrinking algorithm that has inefficient memory management, which is inherent to the used programming language (Python). The memory measurements in Figure 5.4 are therefore exhibiting a slightly increasing trend over time (i.e. as the number of processed points increases) that is not related to the theoretical memory requirements of the algorithm. Nonetheless, I make the key observation that the amount of required memory is successfully reduced. The amount of required memory is now bounded by the largest number of points inside a tile rather than the total point count of the dataset. As a result I am able to process massive surface point clouds.

The simple partitioning scheme that I have implemented effectively limits the amount of memory required to complete the computation of massive datasets. The main limitation of this approach is that the number of points in a tile is still bounded by the amount of available memory. Also, the size of a tile should be at least as large as the buffer radius. Thus for extremely dense datasets it may no longer be feasible to process reasonably sized tiles. However for the tested dataset, with point densities up to over 100 points per square meter, this was not a problem.

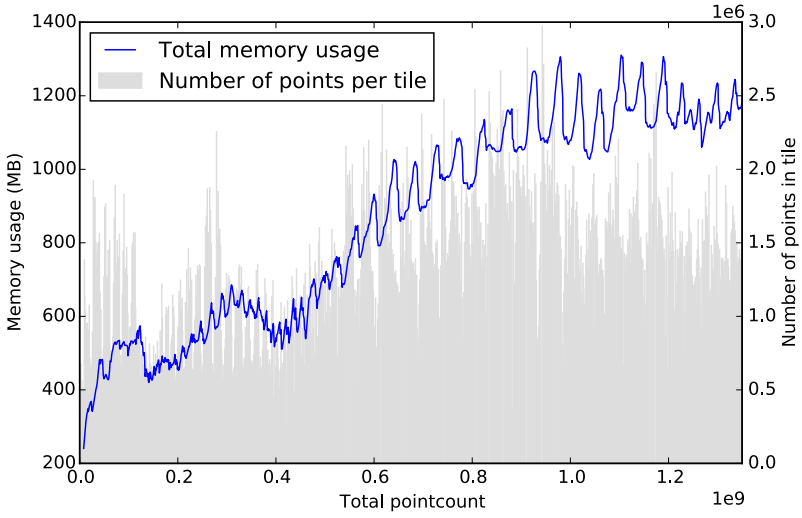


Figure 5.4: Memory consumption as a function of the number of processed points for the MAT approximation of the 1.3 billion point urban dataset.

6 Applications of the unstructured MAT

In Chapters 3 to 5 I laid out the groundwork for geographical point cloud modelling with the MAT. In this chapter I present a three applications based on unstructured MAT (Chapter 3). The novelty of the these methods lies in the application of the unstructured MAT to aerial point clouds.

I present two applications of the MAT that both aim to deal with the potentially massive size of aerial point cloud datasets. First, in Section 6.1 I present a method for feature-aware point cloud simplification, i.e. reducing the number of points in a point cloud in a way that relatively preserves areas with a small feature size (e.g. having a high curvature). Second, in Section 6.2, I present a method for feature-aware point splatting for visualisation purposes that be seen as an extension of the feature-aware point cloud simplification method. This section is based on the paper:

> **Robust approximation of the Medial Axis Transform of LiDAR point clouds as a tool for visualisation**. Ravi Peters and Hugo Ledoux. *Computers & Geosciences* 90(A), March 2016, pp. 123–133, doi: 10.1016/j.cageo.2016.02.019

Both sections are based on the same theoretical foundation: the concepts of Local Feature Size (LFS) and the *epsilon*-sample (as described in Section 2.2.2).

Section 6.3 presents a third application for point cloud-based visibility analysis. This section is based on the paper:

> **Visibility Analysis in a Point Cloud Based on the Medial Axis Transform**. Ravi Peters, Hugo Ledoux and Filip Biljecki. *Eurographics Workshop on Urban Data Modelling and Visualisation 2015*, November 2015, pp. 7–12, doi: 10.2312/udmv.20151342

6.1 Simplification

With recent advances in remote sensing technologies such as aerial LiDAR and photogrammetry we are able to acquire samples of the Earth in unprecedented quantities and with very high accuracy.

The elevation points collected are samples of a digital surface model (DSM) where every objects, natural (e.g. mountains and valleys) and man-made (e.g. buildings, dikes and bridges), are represented. A prime example is the Dutch national elevation the urban dataset AHN2[1] which has a total of over 600 billion points.

While such point clouds have a broad range of applications (such as flood modelling, dike monitoring, crisis management and city modelling [Snyder, 2013]), practitioners have problems dealing with them because of their massive size. They do not fit a computer's main memory, and, as a result, standard software (such as GIS or environmental modelling) can simply not be used.

This problem can be alleviated by designing algorithms so that the limitations of the computer's main memory are never exceeded, e.g. by using the geometry streaming paradigm [Isenburg et al., 2006] or designing external memory algorithms [Aggarwal and Vitter, 1988]. However, the former is only useful for certain *local* problems (e.g. interpolation and creation of grids), *global* problems such as visibility or flow modelling are not suitable. The latter requires careful design of algorithms for different problems (e.g. for extracting contour lines [Agarwal et al., 2008] or watersheds [Arge et al., 2006], which limits its use. Hence, in some cases it is preferred or even necessary to reduce the size of the point cloud prior to processing.

Massive point clouds often contain a lot of redundant information. A high sampling density is often preferred because this leads to a clearer definition of the sampled surface, especially for small features that may otherwise be undersampled. Yet, at the same time many large (and relatively planar) features may be adequately represented using a less dense sampling, i.e. they could be represented with only a subset of their original samples.

Point cloud simplification aims at lowering the overall point count, while maintaining a sufficiently dense sampling of both large and small features. As a result, the same overall surface shape can be adequately described with fewer samples, and less computational resources are needed for any subsequent processing. Joao [1998] even states that various spatial analysis methods can be performed more accurately with a simplified terrain than with the original one. Simplification also implies that practitioners can continue using their current tools, and no new algorithms or tools have to be developed to be able to deal with massive point clouds.

Current simplification methods either require the knowledge of the surface implied by the points (which is not *a priori* known, and is challenging to compute in practice), or randomly select a subset of the points (which ignores the features

[1] http://www.ahn.nl

of the surface). I propose to use a simplification method based on the unstructured MAT that is feature-aware and does not require explicit knowledge of the surface.

The key MAT-based concept that I employ in this chapter is the *local feature size* [Amenta et al., 1998b], i.e. the shortest distance between a surface point to the MAT. It effectively permits us to identify the points where the curvature is high or where the boundary of a shape is close to itself, thus giving a useful definition of the geometric significance of a point. I use this concept to preserve features in a DSM, e.g. roof tops, ridges, valleys, fences.

6.1.1 Related work in 3D point cloud simplification

Following is a discussion of point cloud based simplification methods. I do not elaborate on more traditional 2.5D approaches of simplification and visualisation that are based on raster grids or TINs (see for example Garland and Heckbert [1995]; Gold and Edwards [1992]; Heller [1990]; Kraus and Pfeifer [1998]; Lee [1989]; Vosselman [2000]).

Tools from GIS practice often offer simplistic simplification algorithms for point clouds: a subset is constructed using gridding or random or n^{th} point selection. From a scientific point-of-view, Pauly et al. [2002] implement and review three common approaches for point cloud simplification of densely-sampled smooth shapes. While aerial point clouds are not necessarily smooth and the sampling density varies greatly, I discuss these three approaches and assess them for aerial point clouds.

1. *Clustering* subdivides the point cloud into clusters that are each replaced by one representative sample. The cluster may be defined by the non-empty cells of a regular grid that is superimposed on the input point cloud, in which case the clusters are replaced by the centre points of these cells. Because of the fixed cell size the resulting points are uniformly distributed, which makes it impossible to achieve a sampling density that depends on local curvature or feature size. Alternatively, cluster may be more loosely defined as groups of neighbouring samples that are approximately planar. Each cluster is then replaced by the centroid of the cluster's points.

2. *Iterative simplification*, which can be considered a generalization of Lee [1989], reduces the number of points based on an error metric that quantifies the error that results from the removal of a point. Points are removed in order of increasing error, and every point removal affects the error of surrounding points. Pauly et al. [2002] choose the quadric error metric that was introduced by Garland and Heckbert [1997] for mesh simplification and adapt it for point clouds using estimated point normals and k-neighbourhoods.

3. *Particle simulation:* the idea is to use a point repulsion algorithm to redistribute a user defined number of particles that are randomly placed on the sampled surface. The points in the input point cloud exert a force on the simulated particles until an equilibrium is reached. This approach depends on local surface approximations that prevent particles from drifting away from the sampled surface. To push more particles to regions of high curvature Pauly et al. [2002] weigh the repulsion forces inversely with surface variation, a metric that is similar to curvature.

All of the three described approaches depend on approximate point normals. These are computed for each point by performing a principal component analysis of its k-nearest neighbours. The surface variation defined by Pauly et al. [2002] quantifies the variation along the approximated normal with respect to the tangent plane. They show that surface variation closely resembles curvature, but argue that surface variation is a more meaningful metric for point cloud simplification. Because, when two surfaces come close together, i.e. closer than the smallest enclosing sphere of the k-neighbourhood of the point in question, this is also leads to a higher surface variation. This is interesting because the local feature size metric, that I use in this section, also quantifies this, even without depending on k-neighbourhoods.

6.1.2 Method

I use the MAT-derived *local feature size* (LFS) (see Section 2.2.2) in order to perform feature-aware point cloud simplification. Using the LFS, one can compute an ϵ-sample, i.e. a sampling of the surface $\mathcal{S}$ that relates point density to the LFS.

An ϵ-sample can be approximated from the full surface point cloud by iteratively removing points that do not break the ϵ-sampling criterion. Ma et al. [2012] compute an approximate ϵ-sample from an oversampled input point cloud P by testing for each $\mathbf{p} \in P$ whether the ball $B(\mathbf{p}, \epsilon f(\mathbf{p}))$ contains any point from P other than $\mathbf{p}$ itself. If it does, $\mathbf{p}$ is removed from P.

Ma et al. [2012] implement the approach using a KD-tree This is inefficient in practice, because it requires many nearest neighbour queries on a changing KD-tree, and dynamically removing points from a KD-tree makes it unbalanced and hampers its performance.

My approach is novel in the sense that I do not use a KD-tree for the simplification. Instead I overlay a three-dimensional regular grid on the input point cloud and use a simple calculation to approximate the thinning factor for each grid cell, which is then randomly thinned accordingly. This algorithm is simple to implement and runs in linear time. Therefore it is much faster than the approach of Ma et al. [2012].

The thinning factor for each grid cell c in three steps. Cellsize d, the minimum and maximum pointcounts per cell n_{min} and n_{max}, and ϵ are user parameters.

1. Compute the average LFS f_{avg} of all points in c,
2. Compute the expected number of points $n_{target} = \frac{d}{\epsilon f_{avg}}$
3. Ensure that $n_{min} < n_{target} < n_{max}$, clamp otherwise.
4. Randomly thin the points in c using a factor $x = \frac{n_{target}}{n}$, where n is the number of points in c.

I thus obtain a fast geometry dependent simplification of the point cloud where areas with a large LFS are represented with relatively fewer points than areas with a small LFS.

6.1.3 Results

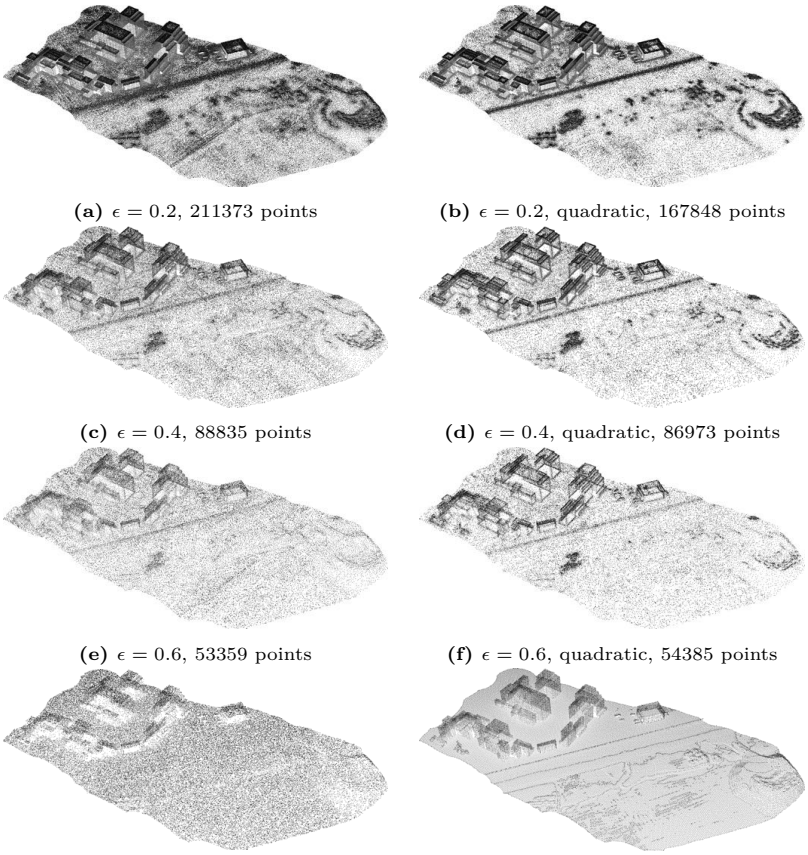
In Figure 6.1 the simplification results for the urban dataset are shown for different values of ϵ.

I have experimented also with a variant of ϵ-sampling, where the point density depends quadratically ($\epsilon f^2(\mathbf{p})$) rather than linearly ($\epsilon f(\mathbf{p})$) on the local feature size. By quadratically modelling this relation I increase the difference in sampling density between points with relatively low local feature size and points with relatively high local feature size.

For comparison I have also simplified the point cloud using random thinning and grid clustering.

I evaluate these results visually, because there is no trivial method to measure a quantitative approximation error for simplified point clouds (this is also noted by Pauly et al. [2002]).

The following observations can be made. For the ϵ-samples, points are clearly more concentrated around small features in terrain and man-made structures. The edges in the block-like structures are relatively densely sampled for all ϵ, even more so in case of the squared ϵ-samples. Planar features such as the ground and the sides of the blocks have lower sampling densities. And despite the lower sampling density on vertical walls when compared to the roofs, the ϵ-samples have comparable densities on both surfaces (unlike the random thinning result). This is expected because an ϵ-sample should not be affected by the input sampling density, as long as the MAT can be reconstructed adequately.



(a) $\epsilon = 0.2$, 211373 points

(b) $\epsilon = 0.2$, quadratic, 167848 points

(c) $\epsilon = 0.4$, 88835 points

(d) $\epsilon = 0.4$, quadratic, 86973 points

(e) $\epsilon = 0.6$, 53359 points

(f) $\epsilon = 0.6$, quadratic, 54385 points

(g) Random thinning to 10%, 74746 points

(h) Grid clustering, with cell size $0.8m$, 68896 points

Figure 6.1: Simplification results for the urban dataset and corresponding point counts

6.1.4 Limitations of LFS-based simplification

I will discuss two limiting aspects of the approach I introduce in this section.

First, the local feature size based simplification and splatting both depend on the estimated local feature size and thus on the quality of the approximated MAT. Distortions in the approximated MAT are therefore visible in the simplification results. While the denoising heuristic as described in Chapter 3 makes the MAT robust to small scale noise in the input point cloud, significant outliers in the input point cloud can still cause distortions that affect the simplification. Furthermore, the denoising heuristic causes a reduced sensitivity for areas with a small local feature size, which leads to a slight overestimation of the local feature size and somewhat affects the effectiveness of the simplification in those areas.

Second, there are a number of parameters for the simplification algorithm that I introduce such as the cellsize that need to be set by the user. While this gives the user more control of the output, an automated parameters setting may be preferred for convenience.

6.1.5 Conclusions

In this section I have made two main contributions. First, I have applied the unstructured MAT, as described in Chapter 3, to aerial point clouds. Second I present a novel method to quickly approximate an *epsilon*-sample.

As a result I am able to effectively perform feature-aware point cloud simplification.

6.2 Feature-aware point splatting based on the MAT

The effective visualisation of massive point clouds is perhaps the most essential instrument a scientist has to analyse and understand a point cloud. As argued by Dykes et al. [2005], visualisation can and should support the entire geoscientific process from the intitial data exploration to synthesis, analysis, evaluation and presentation. However, the visualisation of point clouds is currently hampered by two main problems: (1) due to their massive size they fit neither a computer's main memory nor a computer's graphics memory; and (2) how to achieve a visually pleasing rendering that strengthens the viewer's perception of depth and her sense of structure when only sparse and unstructured points are rendered. As I further describe in Section 6.1.1, it is indeed possible to visualise point clouds that exceed the capacity of a computer's internal memory through the use of out-of-core spatial indexing schemes and by applying methods such as view-frustum culling and multi-resolution hierarchies to select a subset of the points [Kreylos

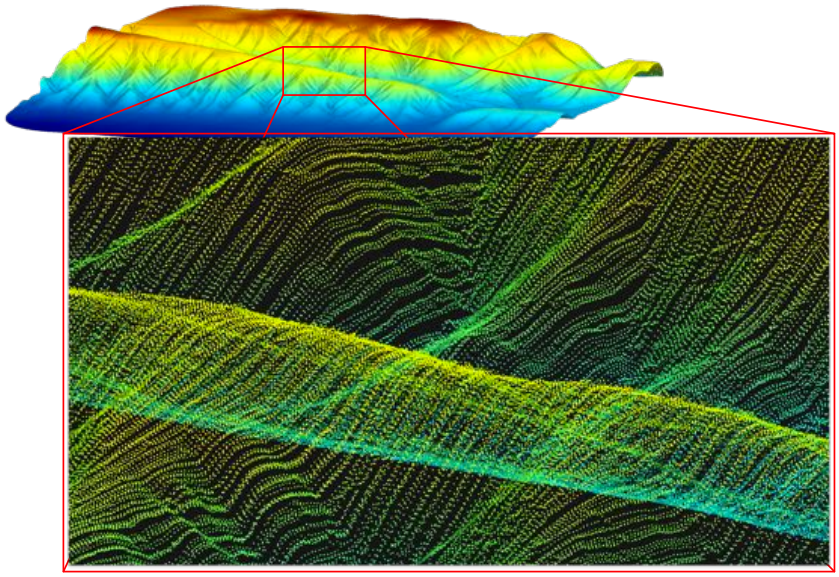


Figure 6.2: Point cloud rendered with shaded fixed-sized points. When zoomed in it is hard to perceive structure and depth, due to the large screen distances between points.

et al., 2008; Richter and Döllner, 2010; Wimmer and Scheiblauer, 2006]. Thus a high frame-rate can be achieved by limiting the number of points that is sent to the graphics card of a computer. I argue that the visual quality is linked to the spatial distribution of points on the screen and the applied point rendering technique. However, current point cloud visualisation methods often use the most basic point rendering techniques and always apply a regular grid-based point simplification scheme that fails to take into account the geometry of the sampled surface. See for instance Figure 6.2 that illustrates how the viewer's sense of depth and structure is distorted at closer viewing distances because of the large gaps between points.

Furthermore, because the geometry of the sampled surface is not taken into account during regular grid-based simplification, fine details can not be adequately represented.

6.2.1 Visualisation of massive point clouds

Kreylos et al. [2008] have implemented a multi-resolution out-of-core octree-based renderer. Their octree-based downsampling scheme is constructed in a pre-processing phase and designed to achieve a uniform point distribution at every level of detail, so it does not consider the geometry of the sampled objects in any way. At any time a subset of the input point cloud is displayed and points are rendered as simple fixed-sized squares with optional shading. While fast and simple to implement, this results in a distorted sense of depth and structure at closer viewing distances (see Figure 6.2). This is due to the presence of holes in the surface when the distances between points become too large on the screen. It is especially a problem for sparsely sampled areas such as vertical surfaces (walls) in aerial point clouds. Wand et al. [2008], Scheiblauer [2014]; Wimmer and Scheiblauer [2006], Richter and Döllner [2010, 2014] and Elseberg et al. [2013] all showcase comparable out-of core octree-based visualisation frameworks for large point clouds with uniform point downsampling. Elseberg et al. [2013] visualise coarser level of details presumably by rendering sets of octree cells as points located in the cells' centre rather than using a subset of the original point cloud.

As illustrated by Figure 6.3, splatting is a point rendering technique where points are rendered as surface aligned disks that are parametrised by some radius r (see also Gross and Pfister [2011]). Usually r is chosen such that the splats are overlapping each other so that holes are absent and the point cloud appears to be a closed surface on the screen regardless of the viewing distance, because r is defined in object space (i.e. in the coordinate system of the point cloud).

Wand et al. [2008] briefly discusses the possibility of using rectangular point splats that are aligned according to the two greatest principal components of

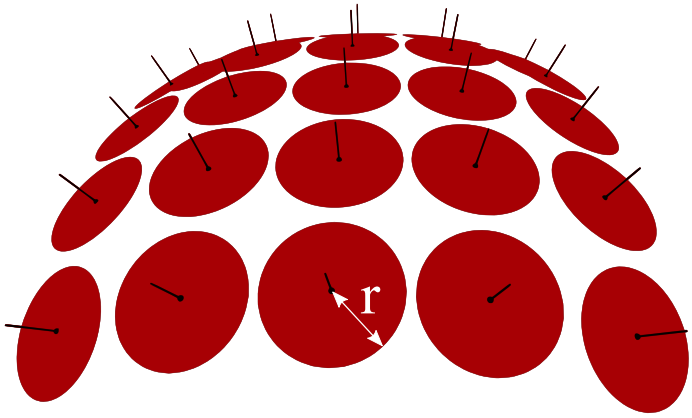


Figure 6.3: A splat is defined for each point as a normal-oriented disk with a radius r. Usually r is chosen such that there are no holes.

a k-neighbourhood. Scheiblauer [2014] and Richter and Döllner [2014] both implement a form of point splatting. The approach of Scheiblauer [2014] differs because he does not use point normals, and the size of his splats depends on the rendered level of detail or on a local point density estimate.

Kovač and Žalik [2010] propose to use a precomputed quadtree index that facilitates on-the-fly point loading and normal computation, but assumes good spatial coherence and is designed to work only for 2.5D surfaces. Points are rendered as oriented splats using the approach of Botsch and Kobbelt [2003]. They apply random subsampling and fixed splat radii, which does not necessarily result in a hole-free visualisation of the scene. As the authors themselves note, the holes are partly caused by inadequate sampling densities in some parts of the point cloud. Indeed, vertical and transparent surfaces are often relatively sparsely sampled in airborne LiDAR point clouds.

It can be concluded that many of the described approaches have solved the issue of managing huge point clouds using out-of-core spatial indexes, and that some of the approaches apply splatting, but none of them takes the geometry of sampled objects in consideration.

6.2.2 Method of feature-aware point splatting

I make the splat radius adaptive to the LFS by rendering each point as a splat with a radius set to $\epsilon f(\mathbf{p})$ (see Figure 6.4). Points in areas with a lower point density in the ϵ-sample are therefore drawn with larger splats. Moreover, because

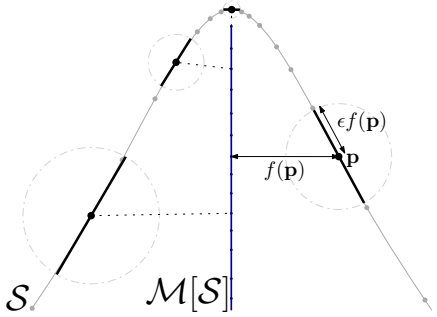


Figure 6.4: The local point density and the radius of the splat for $\mathbf{p}$. For this figure $\epsilon \approx 0.5$.

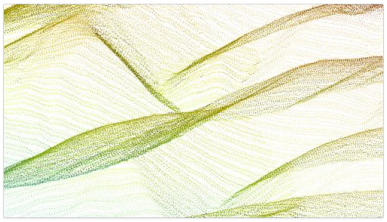
both the splat radii and the ϵ-sample are both based on the distance $\epsilon f(\mathbf{p})$, the resulting visualisation is such that a surface-like impression is obtained where holes are minimised despite the non-homogeneous geometry-aware point simplification.

Results of feature-aware splatting

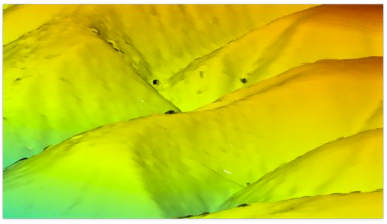
Figures 6.5 and 6.6 demonstrate the effect of the MAT-based simplification and splats with their radii adapted to the LFS.

In both cases the simplification removed 90% of the original points ($\epsilon = 0.4$), yet when rendered with LFS-sized splats the resulting visualisation is similar to the original splatted point cloud with fixed splat-radii. While the simplified LFS-splatted rendering is not absolutely free from holes for the urban dataset (see mark 2 in Figure 6.6), one can also observe that, despite the reduction in points, the sparsely sampled vertical surfaces (walls) now appear as solid surfaces (see mark 1 in Figure 6.6d). This is a notable improvement over fixed-sized splats, because this amplified the viewer's sense of structure and depth at all viewing distances.

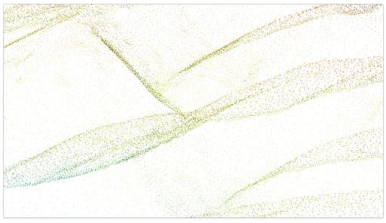
Figure 6.7 illustrates further how the distribution of points in the simplified point cloud respects the geometry of the sampled surface. Splats are drawn there with a decreased radius so that it is clearly visible that 1) more points are drawn in areas with a relatively high curvature such as the creases in the valleys and 2) the corresponding splats have a smaller radius when compared to the planar areas with fewer and larger splats (also apparent in Figure 6.6d). Finally, in Figure 6.8a I compare fixed-sized splats with LFS-sized splats for the simplified mountain dataset. The flat region has fewer samples due to the relatively high LFS. But, in the case of LFS-sized splats, the larger splat radii



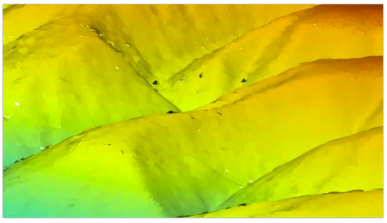
(a) Full point cloud with simple points



(b) Full point cloud with fixed-radius splats

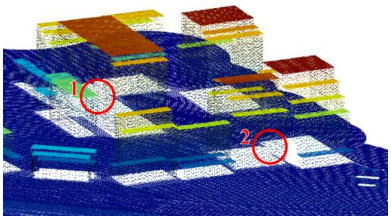


(c) Simplified point (90% of points removed) cloud with simple points

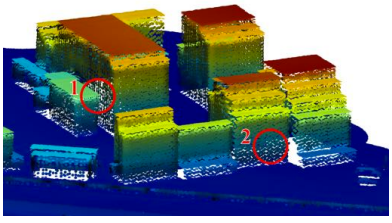


(d) Simplified point (90% of points removed) cloud with lfs-radius splats

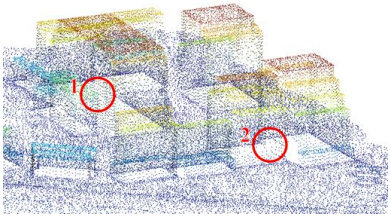
Figure 6.5: Visualisation results for the Mountain dataset ($\epsilon = 0.4$).



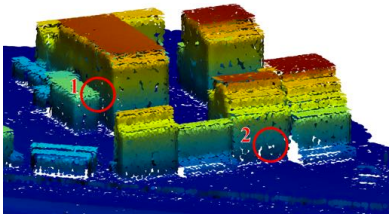
(a) Full point cloud with simple points



(b) Full point cloud with fixed-radius splats



(c) Simplified point (90% of points removed) cloud with simple points



(d) Simplified point (90% of points removed) cloud with lfs-radius splats

Figure 6.6: Visualisation results for the Urban dataset ($\epsilon = 0.4$). Note the different point-densities on vertical and horizontal surfaces (marked 1 and 2).

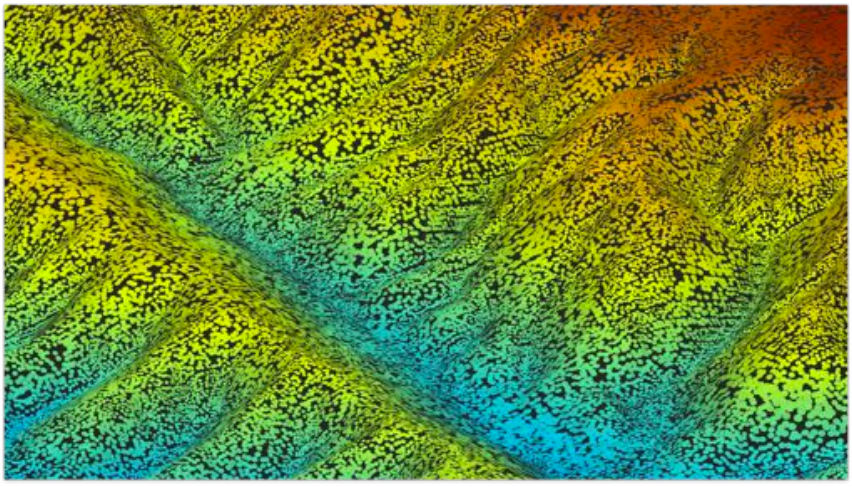
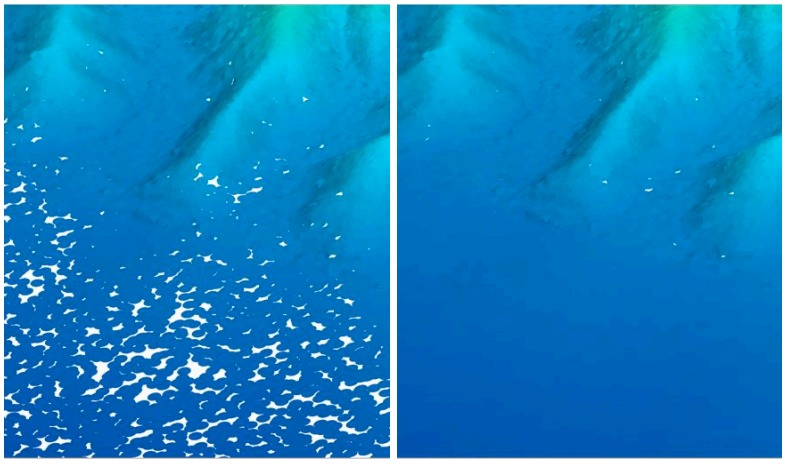


Figure 6.7: MAT-based simplification and splat-radii. The splat radii in this image are reduced for illustrative purposes.



(a) fixed-sized splats **(b)** lfs-sized splats

Figure 6.8: Simplified point cloud ($\epsilon = 0.4$) of mountain dataset.

effectively compensate for the coarser point distribution, leading to a virtually hole-free visualisation.

6.2.3 Conclusions

In this section I have made two main contributions. First, I have applied the unstructured MAT, as described in Chapter 3, aerial point clouds. Second, I introduce the idea to use the local feature size for determining point splat radii.

As a result I am able to effectively perform visualisation in which it is easier to perceive depth and structure in the rendered aerial point cloud, while rendering only a fraction of the full point cloud.

6.3 MAT-based Visibility analysis in point clouds

Visibility analysis is a prominent use case of 3D GIS data, since this provides information about spatial relations and potential obstacles in the line of sight between two points in space. For instance, such analyses have been done in estimating the visibility of a landmark Bartie et al. [2010], and in finding the optimal location to place a surveillance camera Yaagoubi et al. [2015]. An important variant of the visibility analysis is the estimation of shadows, since the position of the sun is variable and it is located at a practically infinite distance Biljecki et al. [2016]. Shadow analysis has gained importance in several disciplines. For instance, shadows are important to account for the loss of the photovoltaic potential Eicker et al. [2015]; Nguyen and Pearce [2012], for determining solar envelopes Knowles [2003], for assessing the value of real estate Helbich et al. [2013], for estimating the thermal comfort Hwang et al. [2011]; Yezioro and Shaviv [1994], and for geovisualisation Lovett [2003].

Visibility analysis is usually performed on a 3D city model, i.e. a boundary representation model that is reconstructed from elevation information, e.g. an airborne LiDAR point cloud, and sometimes combined with 2D datasets (e.g. building footprints from a topographic map). The visibility analysis involves testing if a line of sight (ray) intersects a face in the dataset, usually with algorithms developed in the computer graphics domain, e.g. Möller and Trumbore [1997]. However, the creation and maintenance of 3D city models often involves manual labour and typically results in a generalised version of the city that only contains the terrain and the buildings Alexander et al. [2009]; Biljecki et al. [2014]; Rottensteiner [2003], and in only in some cases other man-made objects such as roads, overpasses, and trees Oude Elberink [2010]. Despite the fact that many 3D city models are reconstructed from very dense points clouds, which

practically contain all urban features, many of these details are lost in the final city model. One cause is that many, especially automatically generated city models, are in fact 2.5D, which means it is not possible to model truly 3D features such as balconies and trees. And even though a number of algorithms for 3D surface reconstruction have been proposed and implemented (see for instance Amenta et al. [2001], Kolluri et al. [2004] and Dey and Sun [2006a]), these have several assumptions on the input point cloud, which usually come from close range laser-scanners and are therefore not suitable for e.g. airborne LiDAR point clouds that are sampled sparsely, have irregular sampling density and often contain significant noise and holes. Hence, despite the availability of highly detailed airborne point clouds, visibility analysis on a derived city model typically deviates significantly from reality.

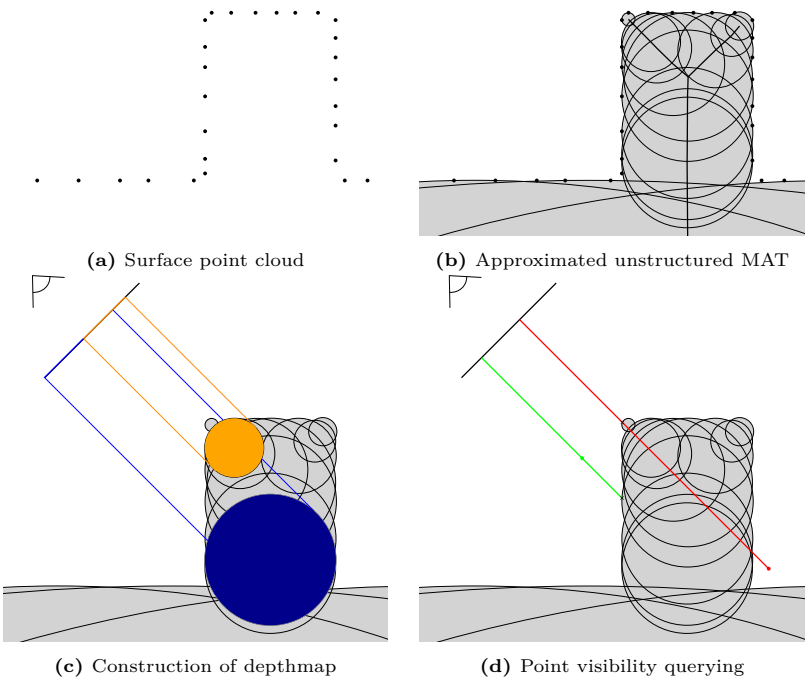
Here I attempt to bypass the generation of a 3D city model, and conduct shadow analysis directly on the geometry of the point cloud by using the 3D MAT. Apart from not having to first generate and store a city model, it yields a more realistic visibility analysis because it includes scanned objects in their true 3D appearance.

Figure 6.9 illustrated my approach to visibility analysis using the MAT. Medial balls are obtained using the ball shrinking algorithm with denoise heuristic as introduced in Chapter 3, and used to construct a view-dependent depthmap, which is then used to perform fast point visibility queries. To investigate the viability of this approach I present experiments on artificial and real-world datasets using a prototype implementation.

6.3.1 Related work

Katz et al. [2007] introduced the hidden point removal operator to determine the visible points in a point cloud as viewed from a given viewpoint by first performing a spherical flipping on the point cloud and then a convex hull computation. The algorithm does not require point normals, and is shown to be useful for shadow mapping and view-dependent surface reconstruction. Mehra et al. [2010] extend the algorithm from Katz et al. [2007] for handling noisy point clouds. However, unlike the algorithm that I present in this chapter, the hidden point removal operator can only determine the visibility of points that are part of the point cloud itself, which limits its potential applications.

Pfister et al. [2000], Sainz and Pajarola [2004] and Kobbelt and Botsch [2004] compute visibility for well-sampled and oriented point clouds as part of a point-based rendering pipeline. They render points as splats; disks that are aligned with the point normals. With these splats a depthmap is computed for the viewport to determine point visibility. However, when the sampling density of



(a) Surface point cloud

(b) Approximated unstructured MAT

(c) Construction of depthmap

(d) Point visibility querying

Figure 6.9: Visibility analysis using the MAT.

the point cloud is low and non-uniform it becomes non-trivial to choose optimal splat radii. My method does not have this problem. Another significant difference with the approach I present in this section is that I compute a volumetric representation of the sampled surface, whereas a splatting approach can represent only the boundary of the sampled surface. Holes are therefore handled quite differently (see also Figure 6.14).

Wald and Seidel [2005] perform ray-tracing in a point cloud based on an implicit surface representation. It works well for high quality point clouds that are densely sampled.

Finally, Jalba et al. [2012] implement a rendering pipeline that performs on-screen surface reconstruction by directly rendering interior medial balls. This is somewhat comparable to my approach, however they only implemented that for visualising the MAT of small point clouds and not for the purpose of visibility analysis.

6.3.2 MAT-based visibility computation

The key idea of MAT-based visibility analysis is to use interior medial balls to volumetrically represent objects and 'block' lines-of-sight from a user-defined viewport to a given set of query points as illustrated in Figure 6.9. This idea has not been applied earlier to perform generic visibility analysis on point clouds. Here I present one possible implementation to this approach, but note that other implementations, e.g. based on ray-tracing using a KD-tree, are possible.. My implementation here is based on the computation of a depthmap, i.e. by rasterising all visible medial balls to a user-defined viewport. Whether the line-of-sight to a query point is blocked or not is then determined by the use of a depthmap that encodes the distances from the viewport to all visible medial balls.

For the sake of simplicity an orthographic projection is assumed and only point visibility queries are considered. This is sufficient to be able to asses the general viability of the idea of MAT-based visibility analysis. Future more comprehensive implementations can address this limitation, as I explain in Section 6.3.6.

6.3.3 Depthmap computation

Prior to performing the visibility queries I generate a depthmap of the interior medial balls. The depthmap is computed for a viewport that is described by a point $\mathbf{p_0}$ to fix its position, two vectors $\vec{v}_x$ and $\vec{v}_y$ to fix its orientation and size in model space and a scalar s that scales model units to the pixels on the screen (see Figure 6.10).

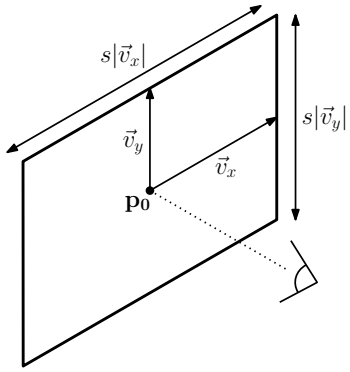


Figure 6.10: Parameters that define the viewport

Algorithm 7: The `projPoint` algorithm.

Input : a point in model coordinates $\mathbf{p_m}$ and the viewport parameters $\mathbf{p_0}, \vec{v}_x, \vec{v}_y, s$

Output: $\mathbf{p_m}$ in screen coordinates denoted as $\mathbf{p_s}$

1 $\vec{v} \leftarrow \mathbf{p_0} - \mathbf{p_m}$
2 $l_{\vec{v}_x} \leftarrow \frac{\vec{v} \cdot \vec{v}_x}{|\vec{v}_x|} \vec{v}_x$
3 $l_{\vec{v}_y} \leftarrow \frac{\vec{v} \cdot \vec{v}_y}{|\vec{v}_y|} \vec{v}_y$
4 $\mathbf{p_s}.x \leftarrow (l_{\vec{v}_x} + |\vec{v}_x|) \frac{s}{|\vec{v}_x|}$
5 $\mathbf{p_s}.y \leftarrow (l_{\vec{v}_y} + |\vec{v}_y|) \frac{s}{|\vec{v}_y|}$
6 $\vec{n} \leftarrow (\vec{v}_y \times \vec{v}_x)$
7 $\mathbf{p_s}.z \leftarrow \frac{\vec{v} \cdot \vec{n}}{|n|}$

Computing the depthmap is a fairly straightforward process that involves first projecting each medial ball centre, rasterising the ball to the viewport and finally performing a depth test for each pixel of the rasterised ball. Figure 6.9b illustrates this process. Algorithm 7 is used to project a point's model coordinates to the viewport and to compute its depth (i.e. its distance perpendicular to the viewport).

Algorithm 8, which updates the depthmap for one medial ball, first projects the ball centre to the viewport. Then for each pixel in the ball's projected image it computes the depth, and performs a depthtest. When a depth test succeeds (i.e. the depth of the ball is smaller than the current pixel depth), the pixel in the depthmap is updated.

Prior to calling `writeBall` for each medial ball, the depthmap is initialised with

Algorithm 8: The `writeBall` algorithm

Input : a ball with centre $\mathbf{c}$ and radius r,the viewport parameter s and depthmap D
Output: D is updated with the depths of ball $(\mathbf{c}, r)$

```
1 c_s ← projPoint (c)
2 for integer x in range −rs to +rs do
3     for integer y in range −rs to +rs do
4         h ← √(x² + y²)
5         if h smaller than rs then
6             d′ ← c_s.z − (rs − h)
7             d ← D[c_s.x + x, c_s.y + y]
8             if d′ smaller than d then
9                 D[c_s.x + x, c_s.y + y] ← d′
```

Algorithm 9: The `queryPoint` algorithm

Input : a querypoint $\mathbf{q_m}$ in model coordinates, depthmap D
Output: whether $\mathbf{q_m}$ is visible or not

```
1 q_s ← projPoint (q_m)
2 d ← D[q_s.x, q_s.y]
3 if q_s.z smaller than d then
4     q_m is visible
5 else
6     q_m is not visible
```

an infinite depth for each pixel.

6.3.4 Point visibility queries

After the depthmap has been computed, Algorithm 9 is used to perform the point visibility queries. Query points are projected onto the viewport, and their depth is compared with the depthmap (similar to Williams [1978]). As illustrated in Figure 6.9c the query point is visible only if its depth test succeeds. The depth test will fail for query points that are behind any medial ball as seen from the viewport.

6.3.5 Implementation and results

I have implemented the method described above using Python. OpenCL [2] is used for parallel execution of the algorithms listed in Section 6.3.3 and 6.3.4.

I ran experiments on two datasets:

1. `Dataset 8`: A simple artificially generated point cloud with its points and normals derived from a triangular mesh (2 690 points), and
2. `Dataset 7`: an airborne LiDAR dataset of a housing block in Zagreb, Croatia (24 647 points).

For the latter dataset the normals were approximated using principal component analysis of the 6 nearest neighbours of each point. For a good separation of the interior and exterior MAT it is important that the normals are properly oriented. This can be achieved by flipping the normals with respect to the scanner position at the time a point is acquired. However, because this information is not present in the LiDAR dataset a city model was used to properly orient the point normals.

For the visibility queries I randomly generated 1 million query points that are uniformly distributed inside the bounding box of the respective dataset. The time complexity for the computation of the depthmap is $O((rs)^2 N)$, with N the number of medial balls, r the ball radius and s the number of pixels per model unit. This is the most expensive algorithm of my approach, and it needs to be recalculated for every new viewport. However, once the depthmap is computed point visibility queries are extremely fast, since they run in constant time (thus independent of depthmap resolution or size of the dataset).

Figure 6.11 shows the results for the artificial dataset. From the depthmap (6.11a) it is clear that three-dimensional features in the point cloud (6.11b) are accurately modelled by the medial balls, given a sufficiently dense and complete sampling. The invisible or 'shadowed' points (inside the bounding box of the point cloud) as seen from the viewport (6.11a) are depicted in Figure 6.11c.

Figures 6.12, 6.13 and 6.14 illustrate the results for the LiDAR dataset. The following observations can be made.

1. Despite the low number of samples on the vertical segments, the building facades are still modelled without holes (6.12d,e).
2. Sparsely sampled details on the buildings such as dormers and chimneys are represented with only a small number of medial balls (6.12b and 6.13).

[2] https://www.khronos.org/opencl/

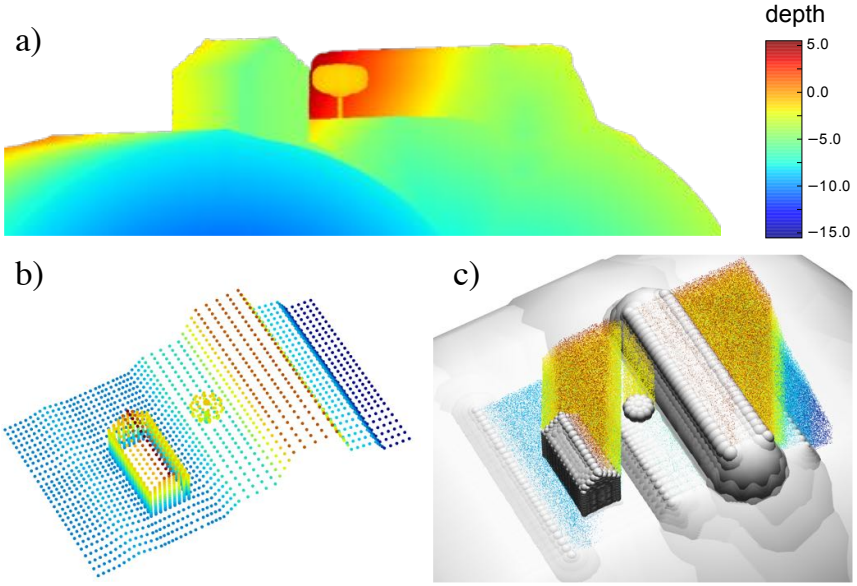


Figure 6.11: Artificially generated dataset. (a) Depthmap for viewport, (b) top-down view of point cloud and (c) point visibility from viewport with medial balls

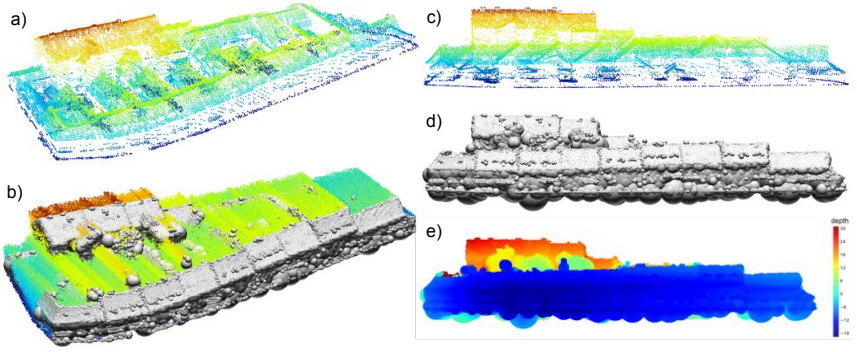


Figure 6.12: Aerial LiDAR point cloud dataset. Top-down view of point cloud (a) and point visibility with medial balls (b). Viewport view with point cloud (c), medial balls (d) and depthmap (e).

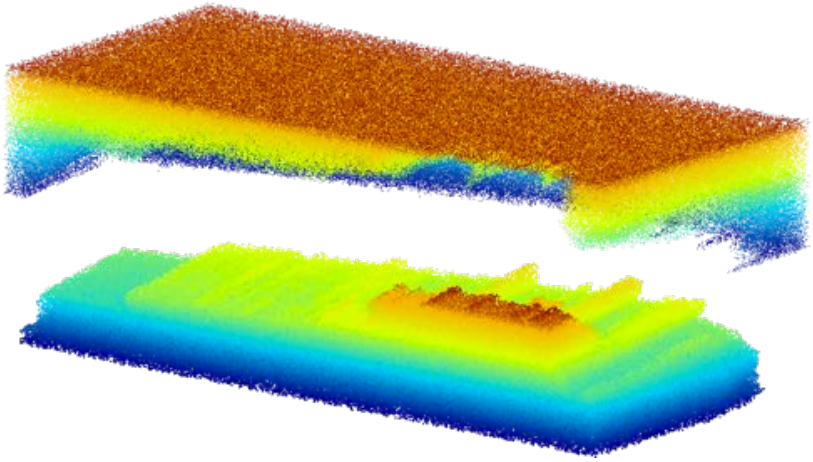


Figure 6.13: Visible (top) and invisible (bottom) points for viewport and LiDAR dataset of Figure 6.12

3. In case of a complete lack of samples for surfaces such as the right side of the roofs in Figure 6.14 an object may be wrongly represented due to protruding medial balls.

4. In the case of trees the orientation of point normals becomes rather ambiguous, which leads to a fuzzy definition of what is an interior or an exterior medial ball, which can lead to medial balls that protrude the tree canopy (see Figure 6.14).

6.3.6 Conclusions

I propose a new approach to visibility analysis in urban scenes directly from a point cloud, thus without the need of an overly simplified intermediate 3D city model. Experimental results using a prototype implementation show that the basic idea of the approach is valid and works on real world data.

There are a number of limitations in the current implementation of the approach:

1. The computation of the depthmap, which needs to happen once for every viewport, is computationally expensive.

2. Only the orthogonal projection is supported.

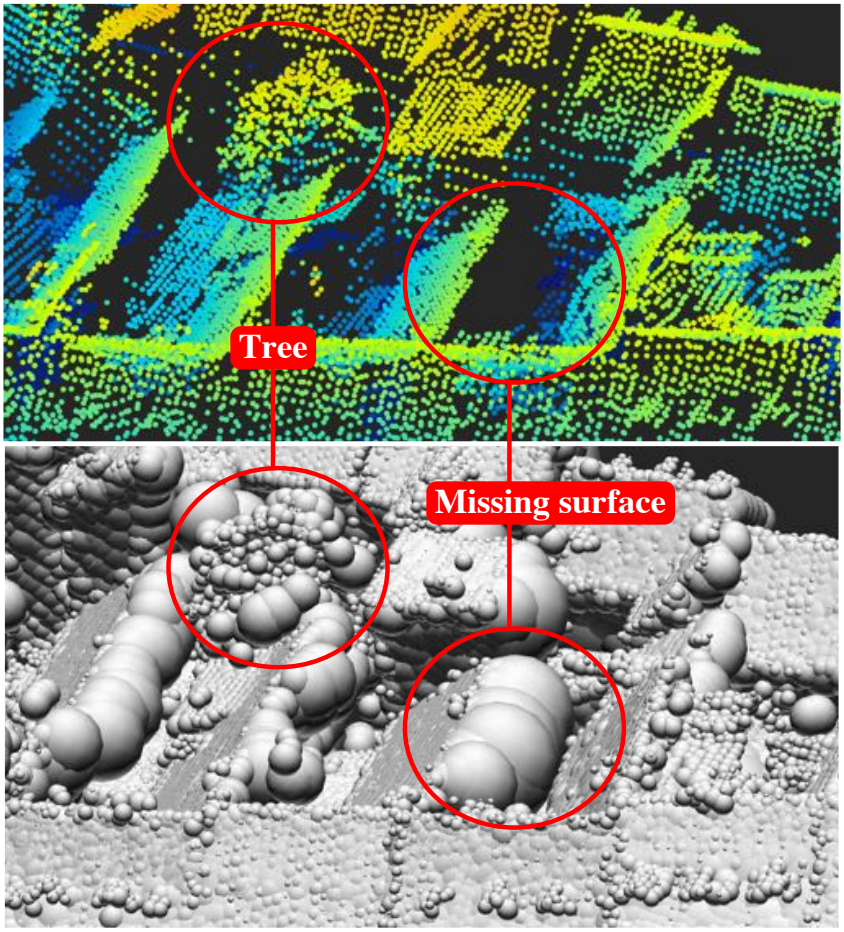


Figure 6.14: Detail view of LiDAR dataset for point cloud (top) and medial balls (bottom)

3. A good separation between the interior and exterior MAT is required. This requires consistently oriented normals that are not typically available for geographical point clouds.

In Section 8.2 I give some ideas on how to overcome these limitations.

7 Applications of the structured MAT

In Chapter 4, I introduced the structured MAT, a robust organisation of the medial atoms of the unstructured MAT into a set of connected medial sheets. In this chapter I demonstrate and explore how to make use of the unstructured MAT mainly for object detection.

Section 7.1 is about the detection of watercourses in the Dutch landscape. This case study uses the medial sheet segmentation that I introduced in Section 4.2.2. Single medial sheets are used to detect watercourses and to reconstruct a 2D centreline for them. This work is based on the paper:

> **Automatic identification of watercourses in flat and engineered landscapes by computing the skeleton of a LiDAR point cloud** . Tom Broersen, Ravi Peters and Hugo Ledoux. *Computers & Geosciences* 106, September 2017, pp. 171–180, doi: 10.1016/j.cageo.2017.06.003

Section 7.2 focuses on the detection of building-like structures using medial clusters, i.e. connected sets of medial sheets, as explained in Section 4.2.1.

7.1 Watercourse detection

Several areas around the world, such as the Netherlands, are characterised by low lying, flat, and engineered agricultural lands. The drainage network of these areas—which is artificial—consists of *connected linear features* such as channels, culverts, and reshaped gullies [Bailly et al., 2011]; we refer to these hereafter as "watercourses". These form a network that transits water from the fields into larger canals [Bouldin et al., 2004]. Typically, these areas have very little variation in elevation, see in Figure 7.7b how the elevation varies only by about 1m over an area of more than 2 km^2. Because engineered lands are sensitive to flooding [Parry et al., 2007], it is of the utmost importance to have an up-to-date and accurate model of the watercourses [Cavalli et al., 2013]. Such a model will consist of the planimetric geometry of the watercourses (their centreline), their connectivity, but also of other characteristics such as the width and the shape of the banks (which is useful to calculate the storage capacity). This information can help us design measures to avoid floods [Cazorzi et al., 2013], and can play

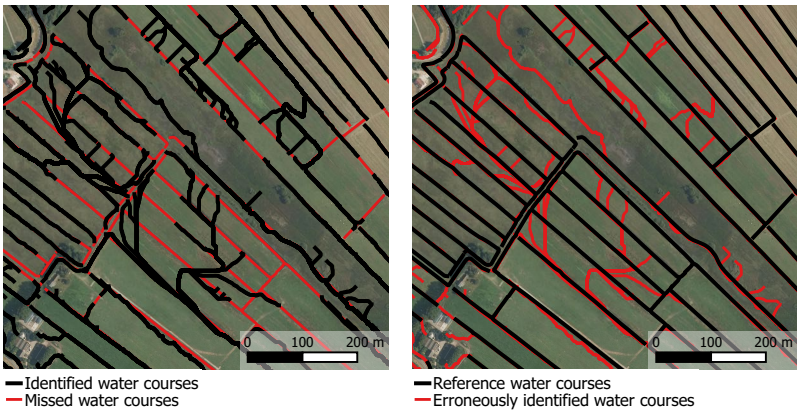
an important role in designing drainage channels and pumping stations [Malano and Hofwegen, 1999].

This application is about the *automatic* detection of the network of watercourses in flat and engineered landscapes. Currently, such networks are typically identified with a semi-automatic methodology using LiDAR point clouds, aerial imagery and field surveys. This is labour-intensive, and subjective [Gandolfi and Bischetti, 1997]. The vast majority of methods and algorithms developed in interdisciplinary studies have not been designed for our case, but for natural landscapes. These usually assume that the slope along a watercourse is always positive [Costa-Cabral and Burges, 1994; Lohani and Mason, 2001], or that the curvature of the terrain is higher than a certain threshold [Brzank et al., 2005; Meisels et al., 1995; Passalacqua et al., 2010], which may not be true for flat landscapes. Another problem with existing methods is that, when LiDAR datasets are used, usually a derived product of the original dataset is used as input, e.g. a 5mX5m gridded digital elevation model (DEM) seems standard. This is a problem since they contain missing data where the water is located (due to the absorption of LiDAR signals by water), and because the conversion to grids inevitably implies a certain decrease of accuracy, due to the interpolation process and the resampling [Brzank et al., 2008; Fisher, 1997; Gold and Edwards, 1992]. Furthermore, a 5 m resolution DEM is insufficient for our case because watercourses can be less than 1 m wide, and several ones can be closer than 5m to each other.

In Section 7.1.2 two skeleton-based approaches are introduced for the automatic detection of water course in flat and engineered landscapes. Both approaches, respectively called the 2D-skeleton method and the 3D-skeleton method, work on the basis of a high resolution LiDAR point cloud. The first is based on conventional 2D GIS operations, and the second is based on the 3D MAT. Both approaches are tested on three study areas in the Netherlands of soil types—this affects the shape of the watercourse profile—and use the national AHN3 point cloud dataset.

7.1.1 Related work

Bailly et al. [2008] identifies watercourses in agricultural areas using LiDAR by analysing the profile at defined locations perpendicular to field boundaries, and choosing a threshold for the curvatures (those above the threshold are ditches). They achieved ditch omissions of around 50%, and ditch commissions of around 15%. They attribute the poor performance to insufficient density of LiDAR points (while they have 10 points/m^2), and to vegetation coverage along the ditches. Their method only works if the boundaries of fields are given as input, and may not perform well for watercourses which are largely filled with water.



(a) Error of omission **(b)** Error of commission

Figure 7.1: Identification of watercourses by Passalacqua et al. [2012] (with GeoNet software) for an area with peat soil near Utrecht in the Netherlands. The dataset is compared to a reference dataset provided by the HDSR (background aerial photo courtesy of `www.pdok.nl`).

Passalacqua et al. [2012] argue that watercourses can typically be characterised by positive curvature, and by high values of flow accumulation. Their method was designed specifically for flat and engineered landscapes. They successfully extracted the network using a 3m DEM for the low-relief human-impacted landscape of an area along a basin in the USA. However, their study area has elevation differences of up to 60 m, and therefore seems to be less flat than the study areas used in this study (see Figure 7.7 and Section 7.1.3). Their method is freely available in the package GeoNet [Sangireddy et al., 2016]. Figure 7.1 shows that it performs poorly for the peat study area used in this study (many errors of omission and commission), although it performed slightly better for the clay area. It struggles in places with very low relief since there is little surface curvature, and thus picks the slightest change.

Cazorzi et al. [2013] extracts local low-relief features from a 1 m DEM, and extract the network by labelling peak values based on a threshold value that is taken as the standard deviation of the local relief. Their results proved to be more reliable than their outdated cartography-based reference data, and a median distance of reference points to the extracted watercourse network was registered to be about 1 m. The usage of a threshold on the local relief can have implications on the ability of the method to identify watercourses of different forms, and as stated above, is less suitable for low-relief watercourses.

Cho et al. [2011] detect stream channels in very low-relief landscapes, based on

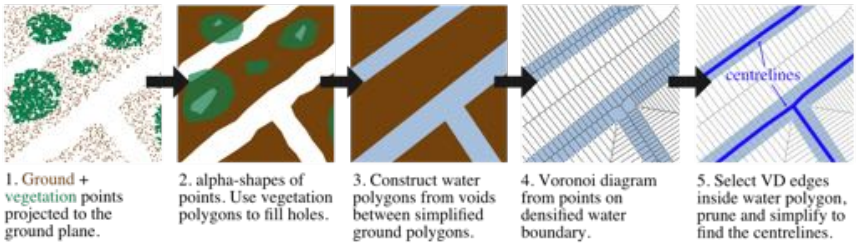


Figure 7.2: Workflow to obtain watercourses from the 2D skeleton.

local minima and maxima in elevation values from a 1m DEM, but comment that the method requires significant training and computation.

Possel et al. [2010] try to detect very wide (100 m) buried channels in an area in the Netherlands from a 2 m DEM with a maximum likelihood classifier based on slope, curvature and relative elevation.

Höfle et al. [2009] extract the edges of a water body by modelling the locations of laser shot drop-outs along with the surface roughness, after which potential water regions are detected by using a region growing algorithm. Toscano et al. [2014] proposes a similar method that requires less pre-processing, but the method uses a DEM. The original LiDAR samples are converted (pixels having no LiDAR signal get a low value) and then an analysis of the height histogram allows them to identify low area (which should be water). Histogram analysis may not suffice for small water bodies found in the Netherlands, since these do probably not generate high enough peaks in the elevation data. Both Höfle et al. [2009] and Toscano et al. [2014] are unable to classify dry watercourses or those completed covered by canopy.

7.1.2 Methods

Two skeleton-based approaches to automatically detect watercourses from a classified aerial LiDAR point cloud are introduced here. The first one (see Figure 7.2) uses the alpha-shape (also commonly called "concave hull") of ground and vegetation points to compute water polygons from which the centrelines are derived using a 2D skeleton. The second one (see Figure 7.3) computes centrelines as the lower envelope of the 3D skeleton (used here as a synonym for the 3D MAT) of ground points and aims at detecting concave profiles, which means it can also detect dry watercourses and watercourses covered partly by canopy.

Note that both methods assume the input point cloud to be classified. The 2D skeleton uses the ground and vegetation classes, whereas the 3D skeleton uses

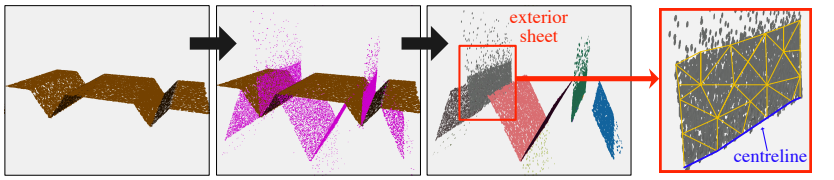


Figure 7.3: Workflow to obtain watercourses from the 3D skeleton.

solely the ground class.

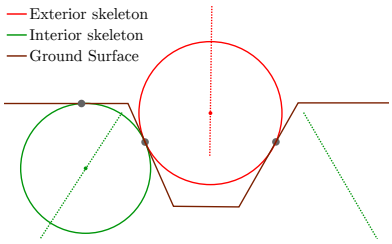
2D skeleton-based watercourse detection

The 2D skeleton-based approach takes advantage of a key property of red laser-based LiDAR datasets above open water bodies: it is almost completely absorbed, only LiDAR signals emitted at or near nadir are reflected strong enough to be detected by the sensor. The few LiDAR measurements which did reflect on the water bodies can be filtered out [Höfle et al., 2009].

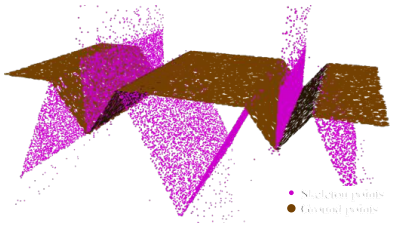
As input for this method we use two sets of points: 1) ground points (which includes the building points to fill the voids in the ground class where buildings are), 2) vegetation points. What remains is a dataset with separate disconnected groups of ground points, with voids in between these groups representing the waterbodies.

As illustrated in Figure 7.2 the 2D skeleton-based comprises of 5 steps:

1. Projecting the points to the ground plane, i.e. $z = 0$.
2. Seperately converting the ground points and vegetation points into multiple disconnected ground and vegetation polygons using *alpha-shapes* (see Edelsbrunner et al. [1983]). Then use the vegetation polygons to fill holes in the ground polygons.
3. Constructing water polygons from the voids in between the filled ground polygons.
4. Compute the Voronoi diagram from the densified boundaries of the water polygons.
5. Construct the 2D skeleton from the Voronoi edges that are completely contained by the water polygons. Prune and simplify. The remaining edges are the centrelines found by this approach.



(a) Cross section of a watercourse. The 3D skeleton is obtained by reconstructing medial balls that touch the ground surface in two points.



(b) Perspective view of the 3D skeleton of simple watercourses.

Figure 7.4: Profile view of 3D skeleton of the terrain

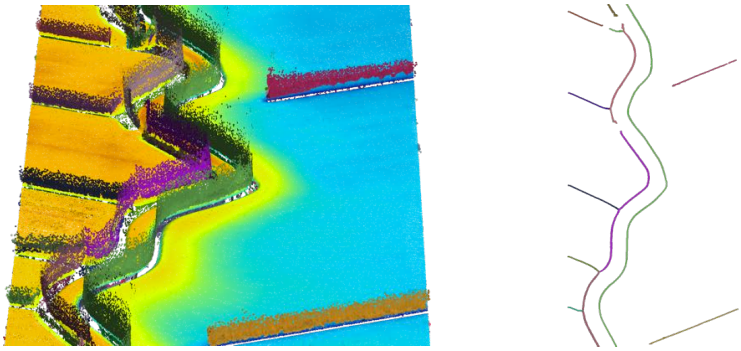
See Broersen et al. [2016] for more details on the construction of the 2D skeleton-based approach that is used here.

3D skeleton-based watercourse detection

The 3D skeleton, i.e. the 3D MAT, is used to detect watercourses based on the three-dimensional morphology of the landscape. The 3D skeleton-based method, illustrated in Figure 7.3, computes centrelines as the lower envelope of the 3D skeleton of ground points and aims at identifying concave profiles, which means it can also identify dry watercourses and watercourses covered partly by canopy.

As depicted in Figure 7.4b, the 3D skeleton of a typical watercourse results in three medial sheets: one exterior (above ground), and two interior (below ground). The exterior medial sheet of a watercourse thus forms a 'centre plane' that contains the centreline of the watercourse. The centreline of a watercourse is defined as the projection to the xy-plane of the lower envelope of its exterior medial sheet (see also Figure 7.3).

For the computation of the 3D skeleton the extended shrinking ball algorithm that is given in Chapter 3 is used. The 3D skeleton is computed for all surface points that are classified as ground. The result is a point approximation (i.e. a point cloud) of the 3D skeleton. Next, a segmentation of the point cloud into distinct medial sheets is performed using a region-growing segmentation algorithm based on the medial bisector as described in Section 4.2.2. An added benefit of this approach is that outliers in the 3D skeleton point cloud can be omitted, since they are not part of any medial sheets and are therefore not assigned to a segment. Figure 7.5 shows the segmentation result for a LiDAR dataset of watercourses. Observe that each watercourse is delineated by a medial



(a) Perspective view of skeleton and ground points.

(b) Plan view of exterior skeleton sheets.

Figure 7.5: Segmentation of 3D skeleton sheets. Each distinct sheet was assigned a random colour. The surface points are coloured by elevation (yellow = low; blue = high).

sheet.

Prior to deriving the 2D centreline representation of each sheet, the medial sheets are triangulated using the ball pivoting algorithm from Bernardini et al. [1999]. A centreline of the water surface can then be derived by selecting the lower edges on the boundary of the triangulated sheet (see also step 4 in Figure 7.3). These lower edges are found by walking around the boundary edges of the triangulation (i.e. those edges that are only incident to one triangle), and then selecting the edges whose two endpoints are below (i.e. having a lower z coordinate) the opposite vertex in the triangle to which that edge belongs. Finally, a 2D representation of the resulting polyline is obtained simply by omitting the z-coordinates.

7.1.3 Experiments & results

In the following experiments we compare the 3D skeleton method with a 2D skeleton method. The 2D skeleton method was implemented using primarily QGIS[1] and LAStools[2]. See Chapter 5 for details on the implementation of the 3D skeleton method. Only for the ball pivoting algorithm, Meshlab[3] was used.

[1] QGIS: http://www.qgis.org.
[2] LAStools: https://rapidlasso.com/lastools/.
[3] MeshLab: http://meshlab.sourceforge.net.

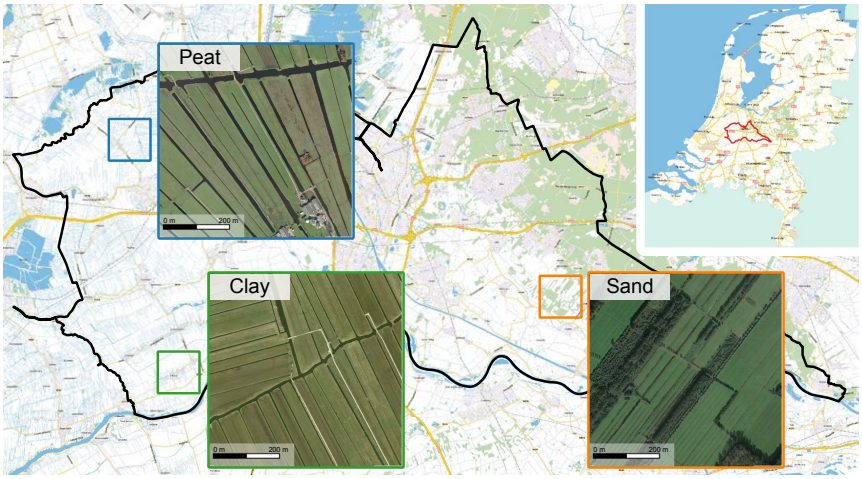


Figure 7.6: The three study areas.

Study area & experiments

The study areas for these experiments are all 3x3 km and are situated around the city of Utrecht, the Netherlands (see Figure 7.6). This area consists for the most part of flat (elevation typically ranges between -2 m to +6 m) and engineered landscapes (see Figure 7.7). We have selected three different types of environments with different characteristics that can be classified according to their subsoils; clay, peat, and sand:

Clay: little vegetation and fairly wide watercourses with a very clear concave profile.

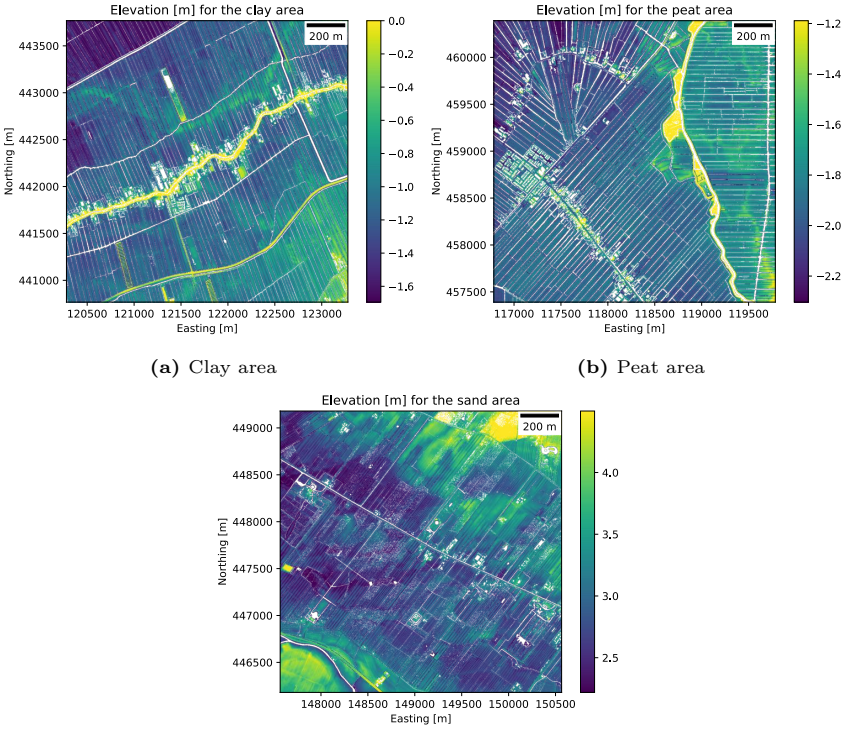
Peat: little vegetation and very wide watercourses with a less clear concave profile.

Sand: a lot of vegetation and narrow watercourses with a clear concave profile.

More detailed information on the study areas can be found in Table 7.1.

The publicly available AHN3[4] aerial LiDAR point cloud data was used, which is the most current version of the national elevation dataset of the Netherlands, it has a point density of around 10 points per square meter. For validation purposes, an existing centreline dataset was used, which was obtained with a semi-automatic method, from HDSR (Hoogheemraadschap De Stichtse Rijnlanden, i.e. the water board responsible for water management in the study areas).

[4] www.ahn.nl



(a) Clay area

(b) Peat area

(c) Sand area

Figure 7.7: Terrain elevation of the study areas. Interpolated from ground points. No data pixels are white.

Table 7.1: Details on study areas. The specified location in EPSG:28992 is the lower left coordinate of the area (they are all 3x3 km in size). The percentage of vegetation coverage is based on the relative number of vegetation points in the point cloud. The percentage of water coverage is computed by taking the total surface area of all water polygons in the HDSR reference dataset and dividing it by the total surface area of the study area.

	Area		
Characteristic	Clay	Peat	Sand
Location (EPSG:28992)	(120279 , 440768)	(116785 , 457391)	(147565 , 446180)
Location (city / village)	Cabauw	Zegveld	Langbroek
Vegetation coverage (%)	5	8	47
Water coverage (%)	9	14	5
Elevation range (cm)	-250/+300	-250/+150	+150/+600

For the experiments, first the two skeleton-based approaches were tested separately on all three study areas. In addition a combined approach is tested, where the centrelines of both skeleton-based appraoches are merged by computing the 2D skeleton of the union of the buffers of both approaches.

The resulting sets of centrelines were compared to the HDSR dataset of watercourses, i.e. our reference dataset. The following error metrics are used (see Lillesand et al. [2008]):

- *Positional accuracy*: Refers to the extent to which the actual position of the watercourses is correctly indicated. It can be estimated by calculating the average positional deviation for multiple watercourses in the generated dataset with respect to the reference dataset.
- *Error of omission*: The percentage of watercourses in the reference dataset that are *not* in the generated dataset.
- *Error of commission*: The percentage of watercourses in the generated dataset that are *not* in the reference dataset.

To compute the error metrics the centrelines are uniformly discretised into points, and for these points the shortest Euclidean distance to the centreline in the reference dataset is found. By aggregating and averaging these point distances per centreline, an estimate is obtained of the generated dataset's positional accuracy. To obtain the mapping accuracies we use threshold distances, e.g. if a threshold distance of 2 m is set, and the distance between a point on the generated centreline and the reference centreline is larger than this distance, then this point counts as an error of commission. Similarly, points can be selected on the generated centreline to find the error of omission. The metrics are

Table 7.2: This table lists the metrics which were computed for the centrelines generated by the 2D skeleton, the 3D skeleton, and the combined approach, for clay, peat and sand areas.

		dataset		
Error metric		Clay	Peat	Sand
Positional accuracy (m)	2D skeleton	0.5	0.7	0.6
	3D skeleton	0.6	0.8	0.8
	Combined	0.6	0.7	0.9
Error of omission (%)	2D skeleton	5	5	58
	3D skeleton	4	15	26
	Combined	2	3	24
Error of commission (%)	2D skeleton	1	2	4
	3D skeleton	8	8	17
	Combined	8	8	17

computed by taking the number of points omitted or committed, relative to the total number of points.

Notice that this evaluation method is comparable to the one proposed by Heipke et al. [1997]. The main difference is that Heipke et al. [1997] use a buffer to match line features between the reference and the generated data instead of the shortest distance between the points on the line features. A benefit of the approach used here is that it can do partial matching, i.e. a parts of the same line feature are matched separately, whereas Heipke et al. [1997] match only the complete line features. This is an advantage especially in the case where the topology of the networks of the reference and the generated datasets are different, which is very likely since they are generated using different methods.

7.1.4 Results

The outcome of the experiments, for all combinations of methods and study areas, are shown in Table 7.2 and summarised below.

Both skeleton-based methods perform very well for the clay area with only 5% of watercourses missing (error of omission) and a low error of commission. The 2D skeleton method works based on the assumption that water surfaces can be detected from the point cloud because water typically is the only surface type that does not reflect red laser. The 3D skeleton method on the other hand works based on the assumption of concavity, i.e. watercourses often have significant surface curvature at the water banks. It is therefore not surprising that both

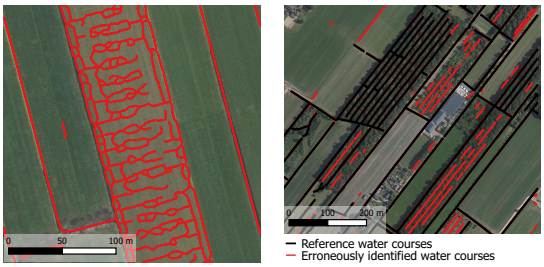
methods perform well on the clay area that has significant concavity and wide watercourse surfaces with little vegetation covering it.

For the peat area, the 2D skeleton performed nearly equally well, since water surfaces are even wider and again there is little vegetation covering these surfaces. However, the watercourses show less clear concave profiles, since relative water levels are higher here (thus many watercourses have only very small banks), which impedes the effectiveness of the 3D skeleton. Although the 3D skeleton performs less for this area, it still manages to identify some of the watercourses which were not identified by the 2D skeleton method. This is indicated by the fact that the combined method identifies roughly 97% of the watercourses, which is more than the 2D skeleton method identified by itself.

The sand area clearly stands out in Table 7.2 since the errors for both methods are significantly worse than for the two other areas. Especially the 2D skeleton method does a poor job at identifying the watercourses with a 58% error of omission. The main problems here are: (1) the fact that water is not well visible in this landscape, (2) water surfaces are often narrower than 1m, and (3) many patches of forest are present. The 3D skeleton is much more effective with only 26% error of omission, but it also struggles with the detection of the narrower watercourses that are naturally also represented with relatively few points in the point cloud. The combined method for the sand area raises the commission error only marginally to 24%, indicating that the 3D skeleton method identified almost all of the watercourses identified by the 2D skeleton, and is clearly the better performing method for this area.

7.1.5 Discussion

The 2D- and 3D-skeleton methods introduced here both have different strengths and limitations. Summarising, it can be said that the 2D skeleton method is particularly efficient with open water watercourses of sufficient width. The (lack of) surface curvature does not affect its effectiveness. And it is characterised by a low error of commission. Its limitations are watercourses with a width of less than 1 m (i.e. depending on the parameters used while creating the alpha-shapes, which itself depends on the density of the LiDAR point cloud; for a denser point cloud, this parameter could be lowered significantly), the presence of large patches of vegetation that cover water bodies (due to the cleaning method of vegetation artefacts), and voids in the LiDAR point cloud that are not the result of waterbodies. The former also means that the 2D skeleton method is ineffective in case of low water levels at time of LiDAR measurements. Thus, the 2D skeleton method is particularly suited for use in areas where water levels are high and water is a predominant feature of the landscape. The 3D skeleton on the other hand does not depend on the presence of water or voids in general, and is effective as long as the canopy is not too dense and allows the LiDAR



(a) Artefact centrelines in the 3D skeleton caused by local relief.

(b) Error of commission of the sand area for the 3D skeleton.

Figure 7.8: Remarkable cases in the results of our experiments.

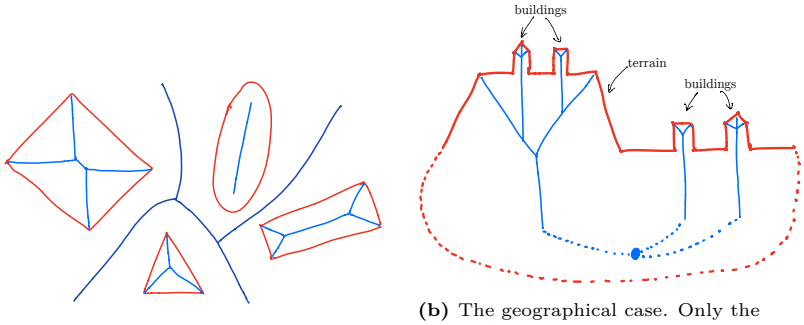
signal to pass through. The limitations of the 3D skeleton are its dependence on surface curvature (high water levels may obfuscate this) an its tendency to find concavities in the landscape where one would not immediately expect a watercourse, e.g. levees or piles of earth or dirt (see Figure 7.8a). The 3D skeleton method is therefore mostly suited for areas with watercourses that have a low water level, clearly concave profiles and may be covered with large patches of vegetation.

7.2 Building detection detection based on the structured MAT

The structured MAT is very promising for applications like object detection and object classification.. There are two main reasons for this. First, the structure of the MAT, i.e. its organisation into medial clusters and medial sheets, implies an intuitive decomposition of shapes into meaningful parts. And second, with the local medial geometry of medial atoms, we have a powerful way to describe and characterise the geometry of those parts. These two properties together make the MAT very appealing for applications that require the detection and effective characterisation of objects in geographical point clouds.

This section describes initial attempts to use the structured MAT for building detection, based on experiments on real-world datasets. Building detection is a form of object detection, i.e. a way to decompose a point cloud into subsets that correspond to meaningful objects. It is an important step (see e.g. Sun and Salvaggio [2013]) toward more advanced and automated applications like object classification, object matching and object reconstruction.

In the previous section, i.e. the detection of watercourses, the decomposition of the MAT into medial sheets was sufficient, since one watercourse typically



(a) A scene of closed manifold objects. Each interior medial cluster by definition corresponds to exactly one object. In addition there is one exterior cluster.

(b) The geographical case. Only the surface that is visibly above the ground is represented. Conceptially the entire terrain is one object (as indicated by the dashed lines) and should correspond to one interior medial cluster. However, due to the incomplete surface information, medial sheets do not converge and we obtain multiple interior clusters.

Figure 7.9: The correspondence between medial clusters and objects in a scene.

corresponds to one medial sheet. However, the MAT of a building is likely a composition of multiple medial sheets, i.e. a medial cluster, as the geometry of a building is more complex. In Chapter 4 I explained how to obtain medial clusters from the unstructured MAT. In this Section, I investigate how buildings can be detected using these medial clusters in two steps. First, I discuss the correspondence between medial clusters and objects in the point cloud. Then, I discuss how to identify which medial clusters are buildings.

7.2.1 The medial cluster of a building

As explained in Section 4.1 the MAT of a scene, e.g. an urban landscape, can be decomposed into medial clusters. The central idea of using the MAT for building detection is that each interior medial cluster corresponds to a distinct object in the landscape. This is certainly true if we consider the theoretical case of a collection of watertight manifold shapes embedded in $\mathbb{R}^3$ as depicted in Figure 7.9a. If one would compute the MAT of this scene, indeed each object would by definition result in one separate set of connected medial sheets, i.e. a medial cluster.

The geographical case is different because objects are not be closed manifolds in geographical scenes, since—as depicted in Figure 7.9b—only the surface visible from above the ground is represented in the point cloud (I also discuss this in

Section 2.1). In theory this should entail that the MAT of all objects in the terrain is connected. In practice, however this is not a problem for flat terrains, since in that case the point where the clusters of two nearby buildings would connect is infinitely far away and will not be present in the approximation of the MAT that I compute. Buildings in hilly terrain may be part of the same cluster.

On the other hand, due to the manner of acquisition of a geographical point cloud as described in Section 1.1, the MAT that we compute in practice is deformed. This may cause

1. connecting sheets between the interior (e.g. in the earth) and the exterior (e.g. in the earth) parts of the MAT (see Figure 7.10a), I call this a supercluster, and
2. single buildings that are composed out of multiple medial clusters (see Figure 7.10b), I call this a fragmented cluster.

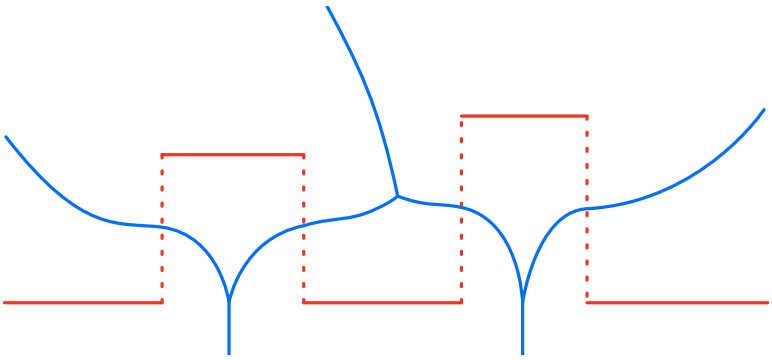
As a result the one-to-one correspondence between a medial cluster and object in the urban landscape can not be taken for granted.

Experiments with simple medial clustering

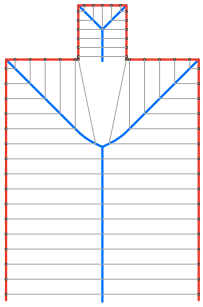
I have performed a number of experiments to roughly demonstrate the effectiveness of the structured MAT for building detection. For each dataset I have simply computed the interior medial clusters form the structured MAT using the methods described in Sections 4.2.1 and 4.4. For visual purposes medial each cluster is a assigned a random colour in the following figures.

Figure 7.11 shows the interior medial clusters for `Dataset` 2. The terrain is flat and all buildings correspond to one or more medial clusters and are thus effectively detected. Some building structures are fragmented clusters. In particular building parts with flat roofs are often in separate clusters, very similar to the case illustrated in Figure 7.10b. See for example Figure 7.11d; two of the dormers on the gabled roof are separate clusters, while the two dormers on the other side of the roof are not. Other examples of fragmented clusters are the flat roofed buildings in the back of the scene.

In Figure 7.12 the results for `Dataset` 5 are shown. A few observations can be made. First, the majority of buildings is detected with the interior medial clusters in Figure 7.12b, despite the dataset's low point density and the lack of virtually all vertical surfaces (e.g. the walls of the buildings). This confirms the idea that the MAT can be effective even for incompletely sampled objects. Notable are the flat roofed buildings on the right side of the dataset. The medial sheets of these buildings are not in separate clusters. The case of Figure 7.10a

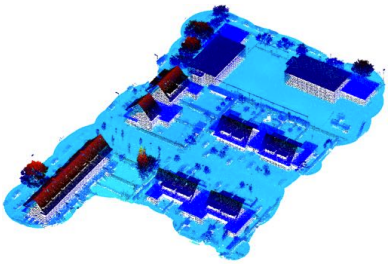


(a) Building objects may contain multiple interior medial clusters or be part of a larger exterior medial cluster.

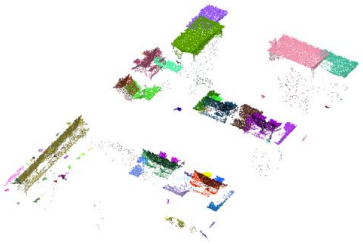


(b) Due to the discrete nature of the unstructured MAT, some connecting medial sheets (ligatures) will not be reconstructed. As a result some building structures will be represented using more than one medial cluster. Spoke vectors shown with grey lines.

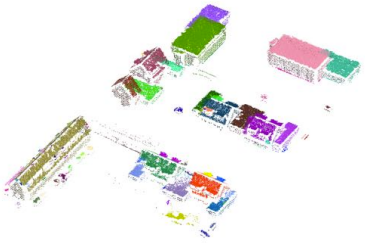
Figure 7.10: Due to missing vertical surfaces (dashed lines) the interior and exterior parts of the MAT may get connected.



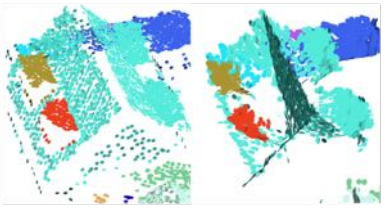
(a) Full surface point cloud.



(b) Interior MAT clusters.

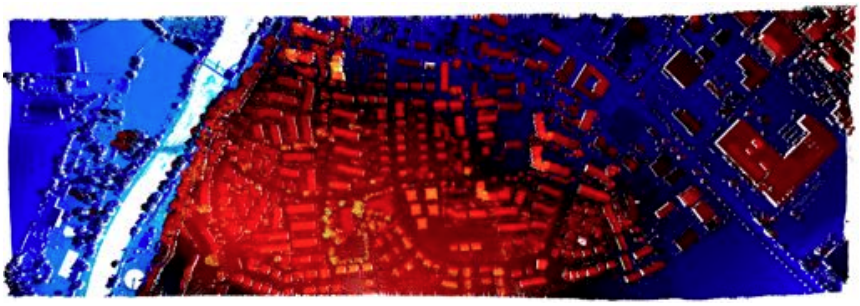


(c) Surface points corresponding to interior MAT clusters. Each segment has a random colour.



(d) Fragmented building cluster. Surface points (left) and MAT points (right)

Figure 7.11: Building detection in `Dataset` 2.

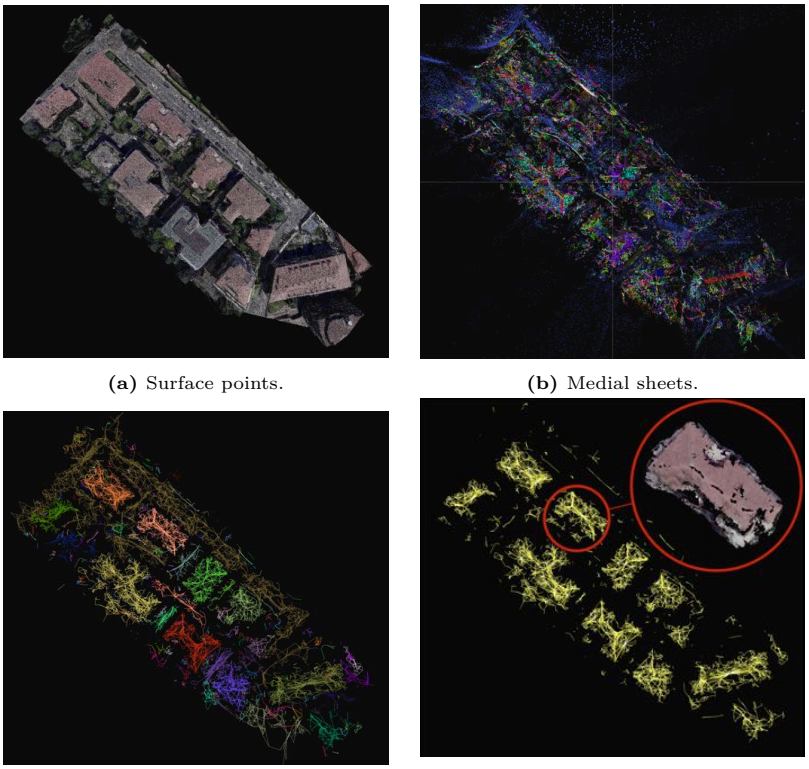


(a) Full surface point cloud with elevation colourmap.

(b) Surface points corresponding to interior medial clusters. Each cluster has a random colour.

(c) Surface points corresponding to interior MAT clusters with an average separation angle of at least 70° and an atom count of at least 50.

Figure 7.12: Building detection in `Dataset` 5. Some of the big buildings structures with flat roofs on the right side are not detected.



(a) Surface points.

(b) Medial sheets.

(c) Interior medial clusters.

(d) Interior clusters with an average separation angle greater than 50°.

Figure 7.13: Building detection in `Dataset` 4. This point cloud was created using dense image matching from oblique imagery.

applies here.Due to the missing wall surfaces the clusters of these buildings are not well separated from the exterior clusters.

Finally, in Figure 7.13 results for a point cloud obtained from dense image matching are shown.

7.2.2 Filtering unwanted interior medial clusters

In Figures 7.11c and 7.12b most interior medial clusters correspond to building structures. However, some of the interior clusters in Figure 7.12b correspond to sloped or hilly parts in the terrain. In addition there are many small clusters

that do not correspond to a building. These clusters can be filtered out based on properties such as the number of medial atoms in a cluster, the minimum or maximum medial ball radius or the average separation angle. Figure 7.12c demonstrates that such simple thresholds can be effective in eliminating many of the unwanted medial clusters.

A richer way to characterise a medial cluster is by examining in more detail its medial sheets. Figure 7.14 shows how this can be achieved. Here, each medial sheet is characterised in terms of the relation between the medial radii and the separation angles. Evidently, this relation is characterised by the type of sheet, i.e. it depends on the surface geometry that generated the sheet. Two simple examples are given: 1) the case of a sheet between to parallel surface patches (Figure 7.14a) and 2) the case of a sheet between to converging linear surface patches (Figure 7.14b). This is a compact way to describe and potentially classify a medial sheet. Furthermore, the combination of sheets in a medial cluster may give a good indication of the type of object represented by the medial cluster (Figure 7.14c). While I have not implemented this, I believe it could be the basis for an effective object classification approach based on the structured MAT.

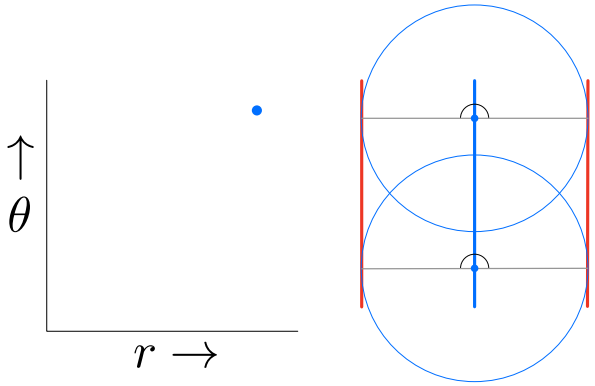
7.2.3 Conclusions

Based on these initial experiments the MAT is indeed appropriate for building detection in geographical point clouds.

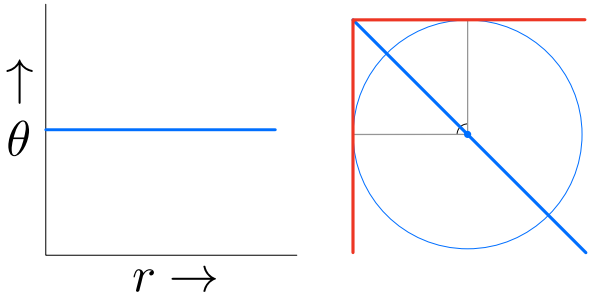
First, the segmentation of the unstructured MAT into medial clusters is straightforward and it immediately gives us the ability to detect entire objects and only requires a raw point cloud. However, the correspondence between a building and a medial cluster is not always one-to-one, which means the decomposition is not always consistent across comparable buildings.

Second, I have applied simple thresholding to remove medial clusters that are not corresponding to buildings. This method can be effective, but requires manual fine-tuning and may inadvertently remove building clusters as well.

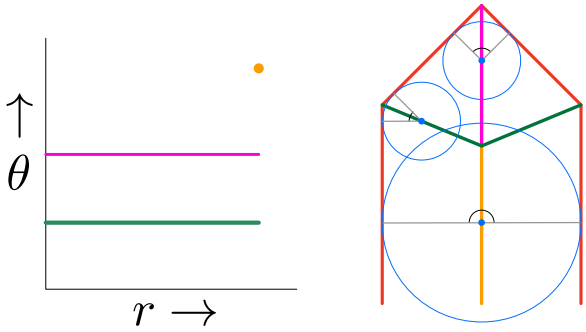
Thus, for both the clustering and the filtering part, future research should be focused on making the method more robust and accurate. I give some suggestions for future work in Chapter 8. Ultimately, the experiments have shown that the MAT can be used to detect building-like objects in geographical points clouds, and therefore potentially other types of objects for which well-defined interior medial clusters.



(a) Constant θ and constant r.



(b) Constant θ and increasing r.



(c) A combination of sheets.

Figure 7.14: Characterisation of a medial cluster by the medial radii (r) and separation angles (θ) of its sheets. Given a medial sheet, the plot of the medial radius against the separation angle of its medial atoms forms a compact description of its geometry.

8 Conclusions and future work

With this thesis I have investigated the 3D MAT as a new approach for geographical point cloud modelling. Key benefits of the MAT for geographical point cloud modelling are 1) it is fully 3D, 2) it can be used to intuitively structure and decompose a point cloud into objects, 3) it clearly separates a point cloud into interior and exterior volumes, and 4) it is able to compactly characterise geometrical properties of a shape though its local medial geometry.

Prior to this thesis the 3D MAT was never applied to geographical point clouds, because of their relatively poor quality in terms of point distribution and acquisition noise (to which the MAT is particularly sensitive). Previous methods in geographic point cloud modelling often relied on the boundary representation of shape and often used 2D or 2.5D techniques. I have shown that the 3D MAT is a viable alternative in geographical point cloud modelling.

My main contributions are:

1. I have shown that the 3D MAT can be robustly computed even for typical geographical point clouds using the shrinking ball algorithm [Ma et al., 2012] with my novel denoising heuristic. The resulting approximation of the MAT, i.e. the unstructured MAT, in itself is a point cloud without any form of organisation.

2. I have proposed a novel method to decompose this point cloud into its constituent medial sheets and to explicitly obtain the topology of those medial sheets, i.e. the structured MAT. Together, these two contributions form the foundation of a usable geographical point cloud modelling framework based on the 3D MAT.

3. I demonstrated the feasibility of this MAT-based geographical point cloud modelling framework, by implementing a number of applications, namely point cloud simplification, visualisation, visibility analysis and object detection.

The 3D MAT offers a new way to analyse and semantically enrich such point clouds, so that they can be used in fields like asset management, crisis management, city and landscape planning, and environmental simulations [Axelsson, 1999; Snyder, 2013]. However, I believe that with this research I have merely scratched the surface of what is possible with the MAT for geographical point

clouds. In Section 8.2 I have therefore listed what I think are the most promising topics for future research, most of which is focused on the further development of the structured MAT. But first, I will get back to the research questions that I presented in Chapter 1 in Section 8.1.

8.1 Research questions

Can the MAT be robustly computed or approximated for geographical point clouds?

Yes, robust and efficient MAT approximation of geographical point cloud, i.e. the unstructured MAT, is possible using the ball shrinking algorithm from Ma et al. [2012] extended with my novel denoising heuristic. The denoising heuristic does come at the expense of a slight reduction in sensitivity of the MAT for small features, as is the case for all known noise handling methods for the MAT.

After approximation of the MAT using the extended ball-shrinking algorithm, the topology of medial sheets in the MAT, i.e. the structured MAT, can be obtained explicitly. This can be achieved by applying a region-growing segmentation based on the medial bisector to decompose the MAT into its medial sheets, followed by the construction of a sheet adjacency graph. With the adjacency graph a geographical point cloud can be decomposed into constituent objects through graph operations like connected component analysis and further classified (e.g. into the interior and exterior MAT) by analysing the aggregated medial geometry of the resulting MAT components, i.e. medial clusters.

An important aspect of working with the MAT for geographical point clouds is the balance between feature size and the local noise levels and point density. This is especially critical for particularly small or thin objects and near sharp edges. In these cases the medial atoms may get distorted in their position and separation angle due to poorly estimated surface normals and medial balls that protrude through the boundary surface. For this reason the effective applications of the MAT are limited to cases where the objects of interest have sufficient thickness and have sufficient surface point density. In practice these are objects that have a clearly defined volume in the point cloud such as for example houses and landscape features but not trees and thin street furniture.

What applications benefit most from MAT-based geographical point cloud modelling?

I show that the unstructured MAT, that lacks an organisation into medial sheets, by itself is already useful for applications such as point cloud simplification. How-

ever, I believe most value lies in the application of the structured MAT. In this thesis I have shown that using fairly simple methods, it is already possible to detect objects such as watercourses and buildings in a geographical point cloud. Consequently, the MAT proves to be a powerful tool for decomposing a geographical point cloud into separate objects. Furthermore, using the local medial geometry, medial clusters can be further characterised. This opens the door to applications such as object classification (I already classify interior and exterior clusters) and shape matching. I further elaborate on these in Section 8.2.

8.2 Future work

8.2.1 Core methodology

Improve robustness of unstructured MAT In this thesis I have shown that the MAT can in fact be robustly approximated for geographical point clouds. Nonetheless, I believe that further improvements to the robustness of the MAT approximation should be explored. A very interesting research direction would be to improve the point normals that are used during approximation. As discussed in Section 3.5.1, the MAT approximation can be distorted due to point normals that are estimated in a 'smoothened' fashion around sharp edges. To improve, one might for instance consider changing the position of a medial atom and the related point normals in such a way that point normals are aligned with all touching medial balls. This might be achieved by e.g. considering that nearby medial atoms in the same medial sheet should have similar medial geometry and could be implemented as a post-processing step to the current method to approximate the unstructured MAT. Note that this would not only improve the quality of the MAT approximation, but also give us a better way to estimate point cloud normals, which is a useful result by itself. In addition, one could consider extending the current denoising heuristic by using additional criteria to detect instabilities such as a ball-emptiness measure as used by Berger and Silva [2012]. This should also improve the accuracy of the unstructured MAT e.g. by better preserving small surface features, that may now be missed in noisy point clouds that require relatively aggressive denoising parameters.

Further development of structured MAT In Chapter 4 I developed a method to decompose the MAT into medial clusters and medial sheets using a region-growing segmentation approach. This is a new result in MAT research that could help in the problem of structuration [Delame et al., 2016]; explicit reconstruction of the medial surfaces as e.g. a triangulation. Conventional structuration methods are not aware of such a decomposition into different

sheets of a medial point cloud and often produce invalid geometries even for non-geographical point clouds. Structuration is hard because one needs to deal with difficult cases such as the intersection of multiple sheet surfaces at one curve. However, when the sheet decomposition is known, the structuration can be performed on a sheet by sheet basis which is much easier since sheets typically do not intersect themselves. In addition, one can use the local medial geometry in the sheet to aid in the reconstruction of the sheet surfaces, something that has not been done before as far as I know.

8.2.2 Application of the MAT

Object detection In Section 7.2 I discuss the use of the MAT for building detection in geographical point clouds. The approach is promising but needs further development. It has difficulties with missing vertical surfaces in the surface point cloud, because in that case medial clusters are not properly separated and the interior and exterior parts of the MAT may get connected. This may be improved e.g. by using a a graph-cut segmentation (see e.g. Sun and Salvaggio [2013]) of problematic sheets that aims to optimise the separation of the interior and exterior MAT based on properties of the medial geometry.

Object classification After object detection follows object classification or shape matching between objects. To this end, it would be interesting to investigate whether an effective feature descriptor can be designed that is based on the medial sheet decomposition of an object and its local medial geometry, e.g. by correlating properties of medial geometry for each sheet (see also Section 7.2.2). Ideally, this feature descriptor could then be used for computing a similarity measure between different objects and for object classification.

Feature aware terrain surface reconstruction This would be an extension of the feature aware point cloud simplification that was described in Section 6.1. Researchers have shown before that it is possible to use a MAT with triangulated medial surfaces for surface reconstruction of high quality point clouds (see e.g. [Amenta et al., 2001] or [Sun et al., 2013]). If one is able to obtain such explicit medial surfaces for the MAT of geographical point clouds, a logical next step would be to attempt surface reconstruction of e.g. terrains in a feature aware manner so that different levels of detail could be generated.

Object reconstruction The reconstruction of individual objects in the terrain, e.g. the polyhedral reconstruction of buildings, can also benefit from the MAT. In particular the MAT's ability to separate between the interior and

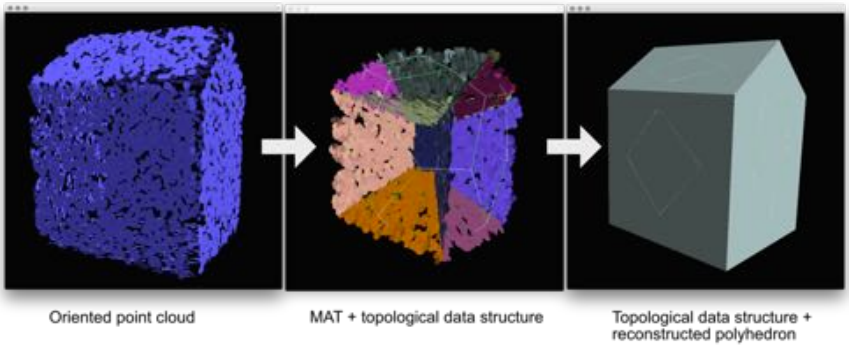


Figure 8.1: From point cloud to polyhedron using the MAT topology

exterior of an object is relevant here. For instance, the approach of Verdie et al. [2015] derives the interior/exterior separation from a photogrammetric mesh and use it to drive a 3D arrangement of planes in an optimisation procedure from which a watertight polyhedral mesh is computed. With the MAT, such method could be adapted for the case an input surface mesh is not available, i.e. when only a point cloud is available, by deriving the interior/exterior separation from the MAT instead of the surface mesh. Another approach could be based on mapping the topology of medial surfaces to the boundary faces of an object. I explored this idea and developed a prototype implementation (see Figure 8.1) that works for carefully prepared artificial datasets. The main challenge in making such an approach work for real-world geographical point clouds is to robustly find the topology between medial sheets.

Breakline detection The MAT has been used before for edge detection in polyhedral meshes [Hisada et al., 2002; Kustra et al., 2016]. A similar method, i.e. based on detecting the surface points that correspond to the outer boundaries of medial sheets, could be applied for breakline detection in the context of terrain analysis. In Section 7.1 I already explained a method for the detection of 2D centrelines of watercourses using medial sheets. The detection of 3D breaklines is a natural extension of this work. In the MAT a breakline should occur where the medial atoms have a zero radius. Unfortunately, those atoms are rarely constructed in practice due to surface noise and the application of the denoising heuristic. A possible solution to this problem would be to extrapolate medial sheets in the direction of decreasing medial radius, after which the explicit 3D linear geometry of the breakline would have to be computed. The problem would be easier to solve if the medial sheet surfaces are available, e.g. as a triangulation.

Watercourse analysis In Section 7.1 I showed how the MAT can be used for the detection of watercourses as 2D centrelines. This work can be possibly extended in a number of ways. For example, one could investigate if the MAT can be used for computing cross-sections of a watercourse, for reconstructing the entire 3D surface of a watercourse, and for the estimation of the volume of a watercourse (i.e. the so-called *storage-capacity*).

Visibility analysis In Section 6.3 I introduced a method for visibility analysis based on the MAT. Some of the limitations of that approach, i.e. the high computational cost and that only the orthogonal projection can be used, may be addressed by using a different implementation based on ray-tracing a KD-tree of medial balls (similar to e.g. [Wald and Havran, 2006]). Such an implementation should yield exactly the same visibility results, but in a more efficient and flexible way. In addition the approach should be extended to make use of the interior/exterior separation of the MAT as described in Section 4.4.

A Datasets

Table A.1: Overview of datasets and their details

#	Acquisistion	Point count	Point density	Source
1	ALS	746351	30-100 m^{-2}	Hoogtebestand Rotterdam 2012
2	ALS	708416	30-100 m^{-2}	Hoogtebestand Rotterdam 2012
3	ALS	93483	30-100 m^{-2}	Hoogtebestand Rotterdam 2012
4	DIM	784673	50+ m^{-2}	FBK, Bergamo/Italy
5	ALS	3776182	5 m^{-2}	ISPRS Benchmark, Vaihingen/Germany
6	ALS	1632040	2 m^{-2}	OpenTopography, Dragons Back Ridge
7	ALS	24647	10 m^{-2}	private
8	AH	2690	n/a	n/a
9	AH	69634	n/a	n/a
10	AH	186577	n/a	n/a
11	AS	1751	n/a	n/a

ALS = Aerial Laser Scanning
DIM= Dense Image Matching
AH = Artificially generated with homogeneous sampling
AS = Artificially generated with ALS simulated sampling

Bibliography

LAS SPECIFICATION VERSION 1.4 – R13. The American Society for Photogrammetry & Remote Sensing, 2013.

P. K. Agarwal, L. Arge, T. Mølhave, and B. Sadri. I/O-efficient algorithms for computing contours on a terrain. In *Proceedings 24th Annual Symposium on Computational Geometry*, pages 129–138, College Park, MD, USA, 2008. ACM Press.

A. Aggarwal and S. Vitter, Jeffrey. The input/output complexity of sorting and related problems. *Communications of the ACM*, 31(9), 1988.

M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. Silva. Point set surfaces. In *Visualization, 2001. VIS '01. Proceedings*, pages 21–29, 537, Oct 2001.

C. Alexander, S. Smith-Voysey, C. Jarvis, and K. Tansey. Integrating building footprints and LiDAR elevation data to classify roof structures and visualise buildings. *Computers, Environment and Urban Systems*, 33(4):285–292, 2009.

N. Amenta and S. Choi. Voronoi methods for 3D medial axis approximation. In *Medial Representations*, pages 223–239. Springer, 2008.

N. Amenta and R. K. Kolluri. The medial axis of a union of balls. *Computational Geometry*, 20(1-2):25 – 37, 2001.

N. Amenta, M. Bern, and D. Eppstein. The crust and the β-skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing*, 60(2): 125 – 135, 1998a.

N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 415–421, New York, NY, USA, 1998b. ACM.

N. Amenta, S. Choi, and R. K. Kolluri. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, SMA '01, pages 249–266, New York, NY, USA, 2001. ACM.

B. Ameri and D. Fritsch. Automatic 3D building reconstruction using plane-roof structures. *ASPRS, Washington DC*, 2000.

L. Arge, A. Danner, H. Haverkort, and N. Zeh. I/O-efficient hierarchical watershed decomposition of grid terrain models. In A. Reidl, W. Kainz, and G. Elmes, editors, *Progress in Spatial Data Handling—12th International Symposium on Spatial Data Handling*, pages 825–844. Springer-Verlag, 2006.

D. Attali and J.-O. Lachaud. Delaunay conforming iso-surface, skeleton extraction and noise removal. *Computational Geometry*, 19(2–3):175 – 189, 2001.

D. Attali and A. Montanvert. Modeling noise for a better simplification of skeletons. In *Image Processing, 1996. Proceedings., International Conference on*, volume 3, pages 13–16. IEEE, 1996.

D. Attali and A. Montanvert. Computing and simplifying 2D and 3D continuous skeletons. *Computer Vision and Image Understanding*, 67(3):261 – 273, 1997.

D. Attali, J.-D. Boissonnat, and H. Edelsbrunner. Stability and computation of medial axes-a state-of-the-art report. *Mathematical foundations of scientific visualization, computer graphics, and massive data exploration*, pages 109–125, 2009.

M. Awrangjeb and C. S. Fraser. Automatic segmentation of raw lidar data for extraction of building roofs. *Remote Sensing*, 6(5):3716–3751, 2014.

P. Axelsson. Processing of laser scanner data—algorithms and applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2):138–147, 1999.

J. S. Bailly, P. Lagacherie, C. Millier, C. Puech, and P. Kosuth. Agrarian landscapes linear features detection from lidar: application to artificial drainage networks. *International Journal of Remote Sensing*, 29(12):3489–3508, 2008.

J. S. Bailly, F. Levavasseur, and P. Lagacherie. A spatial stochastic algorithm to reconstruct artificial drainage networks from incomplete network delineations. *International Journal of Applied Earth Observation and Geoinformation*, 13 (6):853–862, 2011.

G. Barequet, D. Eppstein, M. T. Goodrich, and A. Vaxman. Straight skeletons of three-dimensional polyhedra. In *Algorithms-ESA 2008*, pages 148–160. Springer, 2008.

P. Bartie, F. Reitsma, S. Kingham, and S. Mills. Advancing visibility modelling algorithms for urban environments. *Computers, Environment and Urban Systems*, 34(6):518–531, Nov. 2010.

J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

M. Berger and C. T. Silva. Medial kernels. *Computer Graphics Forum*, 31(2pt4): 795–804, 2012.

F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *Visualization and Computer Graphics, IEEE Transactions on*, 5(4):349–359, Oct 1999.

S. Biasotti, D. Attali, J.-D. Boissonnat, H. Edelsbrunner, G. Elber, M. Mortara, G. Baja, M. Spagnuolo, M. Tanase, and R. Veltkamp. Skeletal structures. In L. Floriani and M. Spagnuolo, editors, *Shape Analysis and Structuring*, Mathematics and Visualization, pages 145–183. Springer Berlin Heidelberg, 2008.

F. Biljecki, H. Ledoux, J. Stoter, and J. Zhao. Formalisation of the level of detail in 3D city modelling. *Computers, Environment and Urban Systems*, 48:1–15, Nov. 2014.

F. Biljecki, J. Stoter, H. Ledoux, S. Zlatanova, and A. Çöltekin. Applications of 3D City Models: State of the Art Review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889, Dec. 2015.

F. Biljecki, H. Ledoux, and J. Stoter. Does a finer level of detail of a 3D city model bring an improvement for estimating shadows? In *Advances in 3D Geoinformation*, pages 1–15. 2016.

H. Blum. A transformation for extracting new descriptors of shape. *Models for the perception of speech and visual form*, 19(5):362–380, 1967.

H. Blum. Biological shape and visual science (part i). *Journal of Theoretical Biology*, 38(2):205 – 287, 1973.

H. Blum. Discussion paper: A geometry for biology. *Annals of the New York Academy of Sciences*, 231(1):19–30, 1974.

G. Borgefors, I. Nyström, and G. S. di Baja. Discrete skeletons from distance transforms in 2d and 3d. In *Medial Representations*, pages 155–190. Springer, 2008.

M. Botsch and L. Kobbelt. High-quality point-based rendering on modern gpus. In *Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference on*, pages 335–343. IEEE, 2003.

J. Bouldin, J. Farris, M. Moore, and C. Cooper. Vegetative and structural characteristics of agricultural drainages in the mississippi delta landscapes. *Environmental Pollution*, 132(3):403 – 411, 2004.

A. Boulch and R. Marlet. Fast and robust normal estimation for point clouds with sharp features. *Computer Graphics Forum*, 31(5):1765–1774, 2012.

J. W. Brandt and V. R. Algazi. Continuous skeleton computation by voronoi diagram. *CVGIP: Image Understanding*, 55(3):329–338, 1992.

C. Brenner. Towards fully automatic generation of city models. *International Archives of Photogrammetry and Remote Sensing*, 33(B3/1; PART 3):84–92, 2000.

T. Broersen, F. W. Fichtner, E. J. Heeres, I. de Liefde, O. B. P. M. Rodenberg, E. Verbree, and R. Voûte. Project pointless. identifying, visualising and pathfinding through empty space in interior point clouds using an octree approach. In *AGILE 2016; 19th AGILE Conference on Geographic Information Science, 14-17 June, 2016; Authors version*, 2016.

A. Brzank, J. Göpfert, and P. Lohmann. Aspects of lidar processing in coastal areas. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36 (Part 1/W3):6, 2005.

A. Brzank, C. Heipke, J. Goepfert, and U. Soergel. Aspects of generating precise digital terrain models in the wadden sea from lidar-water classification and structure line extraction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 63(5):510–528, 2008.

T.-T. Cao, K. Tang, A. Mohamed, and T.-S. Tan. Parallel banding algorithm to compute exact distance transform with the gpu. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pages 83–90. ACM, 2010.

M. Cavalli, S. Trevisani, B. Goldin, E. Mion, S. Crema, and R. Valentinotti. Semi-automatic derivation of channel network from a high-resolution dtm: the example of an italian alpine region. *European Journal of Remote Sensing*, 46:152–174, 2013.

F. Cazorzi, G. D. Fontana, A. De Luca, G. Sofia, and P. Tarolli. Drainage network detection and assessment of network storage capacity in agrarian landscapes. *Hydrological Processes*, 27:541–553, 2013.

J. Chaussard, M. Couprie, and H. Talbot. Robust skeletonization using the discrete λ-medial axis. *Pattern Recognition Letters*, 32(9):1384 – 1394, 2011.

F. Chazal and A. Lieutier. The lambda-medial axis. *Graphical Models*, 67(4): 304–331, 2005.

H. Cho, K. C. Slatton, C. R. Krekeler, and S. Cheung. Morphology-based approaches for detecting stream channels from alsm data. *International Journal of Remote Sensing*, 32(24):9571–9597, 2011.

H. I. Choi, S. W. Choi, and H. P. Moon. Mathematical theory of medial axis transform. *pacific journal of mathematics*, 181(1):57–88, 1997.

C. Constantin, S. Brown, and J. Snoeyink. Implementing streaming simplification for large labeled meshes. In *Proceedings Algorithm Engineering and Experiments (ALENEX10)*, pages 149–158, Austin, Texas, USA, 2010.

M. C. Costa-Cabral and S. J. Burges. Digital elevation model nnetwork (demon): A model of flow over hillslopes for computation of contributing and dispersal areas. *Water Resources Research*, 30(6):1681–1692, 1994.

T. Culver, J. Keyser, and D. Manocha. Exact computation of the medial axis of a polyhedron. *Computer Aided Geometric Design*, 21(1):65–98, 2004.

J. Damon. Smoothness and geometry of boundaries associated to skeletal structures i: sufficient conditions for smoothness (la lissité et géométrie des bords associées aux structures squelettes i: conditions suffisantes pour la lissité). In *Annales de l'institut Fourier*, volume 53, pages 1941–1985, 2003.

M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithm and Applications*, volume Third Edition. Springer, 2000.

T. Delame, J. Kustra, and A. Telea. Structuring 3D Medial Skeletons: A Comparative Study. In *Symposium on Vision, Modeling and Visualization*, Bayreuth, Germany, Oct. 2016.

T. Dey and J. Sun. Normal and feature approximations from noisy point clouds. In S. Arun-Kumar and N. Garg, editors, *FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science*, volume 4337 of *Lecture Notes in Computer Science*, pages 21–32. Springer Berlin Heidelberg, 2006a.

T. K. Dey and S. Goswami. Tight cocone: a water-tight surface reconstructor. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, SM '03, pages 127–134, New York, NY, USA, 2003. ACM.

T. K. Dey and J. Sun. Defining and computing curve-skeletons with medial geodesic function. In *Symposium on Geometry Processing*, pages 143–152, 2006b.

T. K. Dey and W. Zhao. Approximate medial axis as a Voronoi subcomplex. *Computer-Aided Design*, 36(2):195–202, 2004.

T. K. Dey, J. Giesen, and J. Hudson. Decimating samples for mesh simplification. In *Proc. 13th Canadian Conf. Comput. Geom*, pages 85–88, 2001.

P. Dorninger and C. Nothegger. 3D segmentation of unstructured point clouds for building modelling. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 35(3/W49A):191–196, 2007.

P. Dorninger and N. Pfeifer. A comprehensive automated 3D approach for building extraction, reconstruction, and regularization from airborne laser scanning point clouds. *Sensors*, 8(11):7323–7343, 2008.

J. Dykes, A. MacEachren, and M. Kraak. Beyond tools: Visual support for the entire process of giscience. *Exploring Geovisualization*, page 83, 2005.

H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *Information Theory, IEEE Transactions on*, 29(4):551–559, 1983.

U. Eicker, D. Monien, E. Duminil, and R. Nouvel. Energy performance assessment in urban planning competitions. *Applied Energy*, 155:323–333, Oct. 2015.

S. O. Elberink and G. Vosselman. Building reconstruction by target based graph matching on incomplete laser data: Analysis and limitations. *Sensors*, 9(8): 6101–6118, 2009.

J. Elseberg, D. Borrmann, and A. Nüchter. One billion points in the cloud – an octree for efficient processing of 3D laser scans. *ISPRS Journal of Photogrammetry and Remote Sensing*, 76(0):76 – 88, 2013. Terrestrial 3D modelling.

N. Faraj, J.-M. Thiery, and T. Boubekeur. Progressive medial axis filtration. *ACM SIGGRAPH 2013, Technical Briefs*, 2013.

S. Filin and N. Pfeifer. Neighborhood systems for airborne laser data. *Photogrammetric Engineering & Remote Sensing*, 71(6):743–755, 2005.

M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

P. F. Fisher. The pixel: A snare and a delusion. *International Journal of Remote Sensing*, 18(3):679–685, 1997.

G. Forlani, C. Nardinocchi, M. Scaioni, and P. Zingaretti. Complete classification of raw lidar data and 3d reconstruction of buildings. *Pattern Analysis and Applications*, 8(4):357–374, Feb 2006.

M. Foskey, M. C. Lin, and D. Manocha. Efficient computation of a simplified medial axis. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, SM '03, pages 96–107, New York, NY, USA, 2003. ACM.

C. Gandolfi and G. B. Bischetti. Influence of the drainage network identification method on geomorphological properties and hydrological response. *Hydrological Processes*, 11:353–375, 1997.

M. Garland and P. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 209–216, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

M. Garland and P. S. Heckbert. Fast polygonal approximation of terrain and height fields. Technical Report CMU-CS-95-181, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, 1995.

P. Giblin and B. Kimia. On the local form and transitions of symmetry sets, medial axes, and shocks. *International Journal of Computer Vision*, 54(1-3): 143–157, 2003.

J. Giesen, B. Miklos, M. Pauly, and C. Wormser. The scale axis transform. In *Proceedings of the 25th annual symposium on Computational geometry*, pages 106–115. ACM, 2009.

C. Gold and J. Snoeyink. A one-step crust and skeleton extraction algorithm. *Algorithmica*, 30(2):144–163, 2001.

C. M. Gold and G. Edwards. The Voronoi spatial model: Two- and three-dimensional applications in image analysis. *ITC Journal*, 1:11–19, 1992.

A. Golovinskiy and T. Funkhouser. Min-cut based segmentation of point clouds. In *IEEE Workshop on Search in 3D and Video (S3DV) at ICCV*, Sept. 2009.

M. F. Goodchild. Geographical data modeling. *Computers & Geosciences*, 18 (4):401–408, 1992.

G. Gröger and L. Plümer. CityGML—interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 71:12–33, 2012.

M. Gross and H. Pfister. *Point-based graphics*. Morgan Kaufmann, 2011.

M. Gschwandtner, R. Kwitt, A. Uhl, and W. Pree. BlenSor: Blender Sensor Simulation Toolbox Advances in Visual Computing. volume 6939 of *Lecture Notes in Computer Science*, chapter 20, pages 199–208. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2011.

N. Haala and C. Brenner. Interpretation of urban surface models using 2d building information. In *Automatic extraction of man-made objects from aerial and space images (II)*, pages 213–222. Springer, 1997.

N. Haala and M. Kada. An update on automatic 3D building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(6):570–580, 2010.

C. Heipke, H. Mayer, C. Wiedemann, and O. Jamet. Evaluation of automatic road extraction. *International Archives of Photogrammetry and Remote Sensing*, 32(3 SECT 4W2):151–160, 1997.

M. Helbich, A. Jochem, W. Mücke, and B. Höfle. Boosting the predictive accuracy of urban hedonic house price models through airborne laser scanning. *Computers, Environment and Urban Systems*, 39(C):81–92, May 2013.

M. Heller. Triangulation algorithms for adaptive terrain modeling. In *Proceedings 4th International Symposium on Spatial Data Handling*, pages 163–174, Zurich, Switzerland, 1990.

A. Henn, G. Gröger, V. Stroh, and L. Plümer. Model driven reconstruction of roofs from sparse lidar point clouds. *ISPRS Journal of photogrammetry and remote sensing*, 76:17–29, 2013.

W. Hesselink, M. Visser, and J. Roerdink. Euclidean skeletons of 3D data sets in linear time by the integer medial axis transform. In C. Ronse, L. Najman, and E. Decencière, editors, *Mathematical Morphology: 40 Years On*, volume 30 of *Computational Imaging and Vision*, pages 259–268. Springer Netherlands, 2005.

M. Hisada, A. G. Belyaev, and T. L. Kunii. A skeleton-based approach for detection of perceptually salient features on polygonal surfaces. *Computer Graphics Forum*, 21(4):689–700, 2002.

B. Höfle, M. Vetter, N. Pfeifer, G. Mandlburger, and J. Stötter. Water surface mapping from airborne laser scanning using signal intensity and elevation data. *Earth Surface Processes and Landforms*, 34:1635–1649, 2009.

J. Hopcroft and R. Tarjan. Algorithm 447: Efficient algorithms for graph manipulation. *Commun. ACM*, 16(6):372–378, June 1973.

P. V. C. Hough. Method and means for recognizing complex patterns, Dec. 18 1962. US Patent 3,069,654.

H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. R. Zhang. Edge-aware point set resampling. *ACM Trans. Graph.*, 32(1):9:1–9:12, Feb. 2013.

R.-L. Hwang, T.-P. Lin, and A. Matzarakis. Seasonal effects of urban street shading on long-term outdoor thermal comfort. *Building and Environment*, 46(4):863–870, Apr. 2011.

M. Isenburg, Y. Liu, J. R. Shewchuk, and J. Snoeyink. Streaming computation of Delaunay triangulations. *ACM Transactions on Graphics*, 25(3):1049–1056, 2006.

A. C. Jalba, J. Kustra, and A. C. Telea. Surface and curve skeletonization of large 3d models on the gpu. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(6):1495–1508, June 2012.

E. M. Joao. *Causes and Consequences of Map Generalization*. CRC Press, 1998.

M. Kada and L. McKinley. 3D building reconstruction from lidar based on a cell decomposition approach. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38(Part 3):W4, 2009.

R. A. Katz and S. M. Pizer. Untangling the Blum medial axis transform. *International Journal of Computer Vision*, 55(2):139–153, 2003.

S. Katz, A. Tal, and R. Basri. Direct visibility of point sets. *ACM Trans. Graph.*, 26(3), July 2007.

M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pages 61–70, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

J. Kilian, N. Haala, M. Englich, et al. Capture and evaluation of airborne laser scanner data. *International Archives of Photogrammetry and Remote Sensing*, 31:383–388, 1996.

R. L. Knowles. The solar envelope: its meaning for energy and buildings. *Energy and Buildings*, 35(1):15–25, Jan. 2003.

L. Kobbelt and M. Botsch. A survey of point-based techniques in computer graphics. *Comput. Graph.*, 28(6):801–814, Dec. 2004.

R. Kolluri, J. R. Shewchuk, and J. F. O'Brien. Spectral surface reconstruction from noisy point clouds. In *Proceedings Symposium on Geometry Processing*, pages 11–21, Nice, France, 2004.

B. Kovač and B. Žalik. Visualization of LIDAR datasets using point-based rendering technique. *Computers & Geosciences*, 36(11):1443 – 1450, 2010.

K. Kraus and N. Pfeifer. Determination of terrain models in wooded areas with airborne laser scanner data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 53(4):193–203, 1998.

K. Kraus and N. Pfeifer. Advanced dtm generation from lidar data. *International Archives Of Photogrammetry Remote Sensing And Spatial Information Sciences*, 34(3/W4):23–30, 2001.

O. Kreylos, G. Bawden, and L. Kellogg. Immersive visualization and analysis of lidar data. *Advances in Visual Computing*, pages 846–855, 2008.

M. P. Kumler. An intensive comparison of triangulated irregular networks (TINs) and digital elevation models (DEMs). *Cartographica*, 31(2), 1994.

J. Kustra, A. Jalba, and A. Telea. Robust segmentation of multiple intersecting manifolds from unoriented noisy point clouds. *Computer Graphics Forum*, 33 (1):73–87, 2014.

J. Kustra, A. Jalba, and A. Telea. Computing refined skeletal features from medial point clouds. *Pattern Recognition Letters*, 76:13 – 21, 2016. Special Issue on Skeletonization and its Application.

F. Lafarge and C. Mallet. Creating large-scale city models from 3D-point clouds: a robust approach with hybrid representation. *International journal of computer vision*, 99(1):69–85, 2012.

M. Lam. Creating the medial axis transform for billions of LiDAR points using a memory efficient method. Master's thesis, MSc thesis in Geomatics, Delft University of Technology, 2016.

D.-T. Lee. Medial axis transformation of a planar shape. *IEEE Transactions on pattern analysis and machine intelligence*, (4):363–369, 1982.

J. Lee. *A drop heuristic conversion method for extracting irregular network for digital elevation models*. ASPRS/ACSM, 1989.

F. Leymarie and B. Kimia. The medial scaffold of 3D unorganized point clouds. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(2):313–330, 2007.

B. Li, R. Schnabel, R. Klein, Z. Cheng, G. Dang, and S. Jin. Robust normal estimation for point clouds with sharp features. *Computers & Graphics*, 34 (2):94 – 106, 2010.

P. Li, B. Wang, F. Sun, X. Guo, C. Zhang, and W. Wang. Q-mat: Computing medial axis transform by quadratic error minimization. *ACM Transactions on Graphics (TOG)*, 35(1):8, 2015.

Z. Li, Q. Zhu, and C. M. Gold. *Digital Terrain Modeling—Principles and Methodology*. CRC Press, 2005.

T. Lillesand, R. Kiefer, and J. Chipman. *Remote Sensing and Image Interpretation*. John Wiley & Sons, Ltd, 6th edition edition, 2008.

B. Lohani and D. C. Mason. Application of airborne scanning laser altimetry to the study of tidal channel geomorphology. *ISPRS Journal of Photogrammetry and Remote Sensing*, 56:100–120, 2001.

A. Lovett. GIS-based visualisation of rural landscapes: defining 'sufficient' realism for environmental decision-making. *Landscape and Urban Planning*, 65 (3):117–131, Oct. 2003.

J. Ma, S. W. Bae, and S. Choi. 3D medial axis point approximation using nearest neighbors and the normal field. *The Visual Computer*, 28(1):7–19, 2012.

H. Malano and P. V. Hofwegen. *Management of Irrigation and Drainage Systems: A Service Approach*. A. A. Balkema Publishers, Rotterdam, Netherlands, January 1999.

C. Mallet and F. Bretar. Full-waveform topographic lidar: State-of-the-art. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(1):1–16, 2009.

J. N. Mather. Distance from a submanifold in euclidean-space. In *Proceedings of symposia in pure mathematics*, volume 40, pages 199–216, 1983.

K. Matuk, C. Gold, and Z. Li. Skeleton based contour line generalization. *Progress in Spatial Data Handling*, pages 643–658, 2006.

R. Mehra, P. Tripathi, A. Sheffer, and N. J. Mitra. Visibility of noisy point cloud data. *Computers & Graphics*, 34(3):219–230, 2010.

A. Meisels, S. Raizman, and A. Karnieli. Skeletonizing a dem into a drainage network. *Computers & Geosciences*, 21(1):187–196, 1995.

B. Miklos, J. Giesen, and M. Pauly. Discrete scale axis representations for 3d geometry. *ACM Trans. Graph.*, 29:101:1–101:10, July 2010.

T. Möller and B. Trumbore. Fast, minimum storage ray-triangle intersection. *Journal of Graphics Tools*, 2(1):21–28, 1997.

P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. van Gool, and W. Purgathofer. A survey of urban reconstruction. *Computer Graphics Forum*, 32 (6):146–177, 2013.

H. T. Nguyen and J. M. Pearce. Incorporating shading losses in solar photovoltaic potential assessment at the municipal scale. *Solar Energy*, 86(5):1245–1260, May 2012.

R. Ogniewicz and M. Ilg. Voronoi skeletons: Theory and applications. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pages 63–69. IEEE, 1992.

Open Geospatial Consortium. City Geography Markup Language (CityGML) Encoding Standard, version: 2.0.0, 2012.

S. J. Oude Elberink. *Acquisition of 3D topography: automated 3D road and building reconstruction using airborne laser scanner data and topographic maps.* University of Twente, 2010.

J. Overby, L. Bodum, E. Kjems, and P. Iisoe. Automatic 3D building reconstruction from airborne laser scanning and cadastral data using hough transform. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34:296–301, 2004.

M. Parry, O. Canziani, J. Palutikof, P. van der Linden, and C. Hanson. Contribution of working group ii to the fourth assessment report of the intergovernmental panel on climate change. *IPCC Fourth Assessment Report: Climate Change 2007*, 2007.

P. Passalacqua, T. Do Trung, E. Foufoula-Georgiou, G. Sapiro, and W. E. Dietrich. A geometric framework for channel network extraction from lidar: Nonlinear diffusion and geodesic paths. *Journal of Geophysical Research: Earth Surface*, 115(F1), 2010. F01002.

P. Passalacqua, P. Belmont, and E. Foufoula-Georgiou. Automatic geomorphic feature extraction from lidar in flat and engineered landscapes. *Water Resources Research*, 48(3), 2012.

M. Pauly, M. Gross, and L. Kobbelt. Efficient simplification of point-sampled surfaces. In *Visualization, 2002. VIS 2002. IEEE*, pages 163–170. IEEE, 2002.

H. Pfister, M. Zwicker, J. Van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 335–342. ACM Press/Addison-Wesley Publishing Co., 2000.

S. M. Pizer, K. Siddiqi, G. Székely, J. N. Damon, and S. W. Zucker. Multiscale medial loci and their properties. *International Journal of Computer Vision*, 55(2-3):155–179, 2003.

B. M. Possel, R. C. Lindenbergh, and J. E. Storms. *Automatic detection of buried channel deposits from dense laser altimetry data.* International Society of Photogrammetry and Remote Sensing (ISPRS), 2010.

T. Rabbani, F. Van Den Heuvel, and G. Vosselmann. Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5):248–253, 2006.

R. Richter and J. Döllner. Out-of-core real-time visualization of massive 3d point clouds. In *Proceedings of the 7th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, pages 121–128. ACM, 2010.

R. Richter and J. Döllner. Concepts and techniques for integration, analysis and visualization of massive 3d point clouds. *Computers, Environment and Urban Systems*, 45:114–124, 2014.

F. Rottensteiner. Automatic generation of high-quality building models from lidar data. *IEEE Computer Graphics and Applications*, 23(6):42–50, 2003.

F. Rottensteiner. Consistent estimation of building parameters considering geometric regularities by soft constraints. In *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2006.

F. Rottensteiner, G. Sohn, M. Gerke, J. D. Wegner, U. Breitkopf, and J. Jung. Results of the ISPRS benchmark on urban object detection and 3d building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 93: 256 – 271, 2014.

P. K. Saha, G. Borgefors, and G. S. di Baja. A survey on skeletonization algorithms and their applications. *Pattern Recognition Letters*, 76:3–12, 2016.

M. Sainz and R. Pajarola. Point-based rendering techniques. *Computers & Graphics*, 28(6):869 – 879, 2004.

H. Sangireddy, C. P. Stark, A. Kladzyk, and P. Passalacqua. GeoNet: An open source software for the automatic and objective extraction of channel heads, channel network, and channel morphology from high resolution topography data. *Environmental Modelling & Software*, 83:58–73, 2016.

C. Scheiblauer. *Interactions with Gigantic Point Clouds*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, 2014.

R. Schnabel, R. Wessel, R. Wahl, and R. Klein. Shape recognition in 3D point-clouds. In *The 16-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, volume 8. Citeseer, 2008.

J. Shan and C. K. Toth. *Topographic laser ranging and scanning: principles and processing*. CRC press, 2008.

D. J. Sheehy, C. G. Armstrong, and D. J. Robinson. Computing the medial surface of a solid from a domain delaunay triangulation. In *Proceedings of the third ACM symposium on Solid modeling and applications*, pages 201–212. ACM, 1995.

K. Siddiqi and S. M. Pizer. *Medial representations: mathematics, algorithms and applications*, volume 37. Springer, 2008.

G. I. Snyder. The benefits of improved national elevation data. *Photogrammetric Engineering and Remote Sensing*, 79(2), 2013.

A. Sobiecki, H. C. Yasan, A. C. Jalba, and A. C. Telea. Qualitative comparison of contraction-based curve skeletonization methods. In *Mathematical Morphology and Its Applications to Signal and Image Processing*, pages 425–439. Springer, 2013.

G. Sohn and I. Dowman. Data fusion of high-resolution satellite imagery and lidar data for automatic building extraction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 62(1):43 – 63, 2007.

A. Sud, M. A. Otaduy, and D. Manocha. DiFi: Fast 3D distance field computation using graphics hardware. *Computer Graphics Forum*, 23(3):557–566, 2004.

A. Sud, M. Foskey, and D. Manocha. Homotopy-preserving medial axis simplification. *International Journal of Computational Geometry & Applications*, 17 (05):423–451, 2007.

F. Sun, Y.-K. Choi, Y. Yu, and W. Wang. Medial meshes for volume approximation. *arXiv preprint arXiv:1308.3917*, 2013.

S. Sun and C. Salvaggio. Aerial 3D building detection and modeling from airborne lidar point clouds. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(3):1440–1449, June 2013.

A. Tagliasacchi, T. Delame, M. Spagnuolo, N. Amenta, and A. Telea. 3D skeletons: A state-of-the-art report. *Computer Graphics Forum*, 35(2):573–597, 2016.

R. Tam and W. Heidrich. Shape simplification based on the medial axis transform. In *Visualization, 2003. VIS 2003. IEEE*, pages 481–488. IEEE, 2003.

F. Tarsha-Kurdi, T. Landes, and P. Grussenmeyer. Extended ransac algorithm for automatic detection of building roof planes from lidar data. *The photogrammetric journal of Finland*, 21(1):97–109, 2008.

G. Toscano, U. Gopalam, and V. Devarajan. Auto hydro break line generation using LiDAR elevation and intensity data. In *Proceedings ASPRS 2014 Annual Conference*, Louisville, Kentucky, USA, 2014.

Y. Verdie, F. Lafarge, and P. Alliez. LOD Generation for Urban Scenes. *ACM Trans. on Graphics*, 34(3), 2015.

V. Verma, R. Kumar, and S. Hsu. 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, 2006.

J.-P. Virtanen, A. Kukko, H. Kaartinen, A. Jaakkola, T. Turppa, H. Hyyppä, and J. Hyyppä. Nationwide point cloud—the future topographic core data. *ISPRS International Journal of Geo-Information*, 6(8), 2017.

G. Vosselman. Slope based filtering of laser altimetry data. In *Proceedings IAPRS, Vol. XXXIII*, Amsterdam, the Netherlands, 2000.

G. Vosselman, S. Dijkman, et al. 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, 2001.

R. Wahl, R. Schnabel, and R. Klein. From detailed digital surface models to city models using constrained simplification. In *Photogrammetrie, Fernerkundung, Geoinformation*. Citeseer, 2008.

I. Wald and V. Havran. On building fast kd-trees for ray tracing, and on doing that in o (n log n). In *Interactive Ray Tracing 2006, IEEE Symposium on*, pages 61–69. IEEE, 2006.

I. Wald and H.-P. Seidel. Interactive ray tracing of point-based models. In *Proceedings of the Second Eurographics / IEEE VGTC Conference on Point-Based Graphics*, SPBG'05, pages 9–16, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

M. Wand, A. Berner, M. Bokeloh, P. Jenke, A. Fleck, M. Hoffmann, B. Maier, D. Staneker, A. Schilling, and H.-P. Seidel. Processing and interactive editing of huge point clouds from 3D scanners. *Computers & Graphics*, 32(2):204 – 220, 2008.

L. Williams. Casting curved shadows on curved surfaces. In *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '78, pages 270–274, New York, NY, USA, 1978. ACM.

M. Wimmer and C. Scheiblauer. Instant points: Fast rendering of unprocessed point clouds. In *SPBG*, pages 129–136. Citeseer, 2006.

B. Xiong, S. O. Elberink, and G. Vosselman. A graph edit dictionary for correcting errors in roof topology graphs reconstructed from point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 93:227 – 242, 2014.

B. Xiong, S. O. Elberink, and G. Vosselman. Footprint map partitioning using airborne laser scanning data. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 3(3), 2016.

R. Yaagoubi, M. Yarmani, A. Kamel, and W. Khemiri. HybVOR: A Voronoi-Based 3D GIS Approach for Camera Surveillance Network Placement. *ISPRS International Journal of Geo-Information*, 4(2):754–782, May 2015.

A. Yezioro and E. Shaviv. Shading: A design tool for analyzing mutual shading between buildings. *Solar Energy*, 52(1):27–37, Jan. 1994.

K. Zhang, S.-C. Chen, D. Whitman, M.-L. Shyu, J. Yan, and C. Zhang. A progressive morphological filter for removing nonground measurements from airborne lidar data. *IEEE transactions on geoscience and remote sensing*, 41 (4):872–882, 2003.

Q. Zhou and A.-X. Zhu. The recent advancement in digital terrain analysis and modeling. *International Journal of Geographical Information Science*, 27(7): 1269–1271, 2013.

Q.-Y. Zhou and U. Neumann. *2.5D Dual Contouring: A Robust Approach to Creating Building Models from Aerial LiDAR Point Clouds*, pages 115–128. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

Summary

A geographical point cloud is a detailed three-dimensional representation of the geometry of our geographic environment. Using geographical point cloud modelling, we are able to extract valuable information from geographical point clouds that can be used for applications in asset management, crisis management, city and landscape planning, and environmental simulations. During this process the point cloud is semantically enriched, e.g. by performing classification, and structurally enriched, e.g. by performing segmentation or surface reconstruction.

In this thesis I propose a new approach to geographical point cloud modelling based on the 3D Medial Axis Transform (MAT), a skeleton-like representation of shapes that explicitly models both the topology and the geometry of shapes. While the 3D MAT has been used before in other fields, its application to geographical point clouds is novel. Advantages of the MAT over existing mostly 2.5D and boundary representation-based methods include that 1) it is fully 3D, 2) it can be used to intuitively structure and decompose a point cloud into objects, 3) it clearly separates a point cloud into interior and exterior volumes, and 4) it is able to compactly characterise geometrical properties of a shape though its local medial geometry.

I make three core contributions. First, I explain how to robustly approximate the 3D MAT for large real-world geographical point clouds. This is critical for geographical point clouds because they are inherently noisy due to the challenging acquisition conditions and the fact that the MAT in itself is highly sensitive to noise. Second, I show how to structure the MAT into a connected set of medial sheets that form so-called *medial clusters* that give us a natural decomposition of the point cloud into objects. Third, I demonstrate how the MAT can be applied for feature aware point cloud simplification and visualisation, visibility analysis, watercourse detection, and building detection.

Due to noise and limitations in the point density of geographical point clouds, the MAT performs best for objects that have a clearly defined volume in the point cloud such as for example houses and landscape features. It is less suitable for object like trees and thin street furniture.

The core result of this thesis is that I prove that the 3D MAT is a useful and practically viable tool for geographical point cloud modelling.

Samenvatting

Een geografische puntenwolk is een gedetailleerde drie-dimensionale representatie van de geometrie van onze geografische omgeving. Uit geografische puntenwolken kan waardevolle informatie worden verkregen die gebruikt kan worden voor toepassingen in asset management, crisismanagement, gebiedsontwikkeling en studies over milieu en natuur. Deze informatie wordt uit de puntenwolk verkregen door het inbrengen van semantiek en structuur, door middel van bijvoorbeeld classificatie, segmentatie en oppervlakte reconstructie.

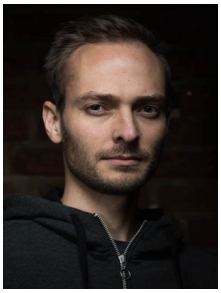
Ik introduceer een nieuwe aanpak voor het modelleren van geografische puntenwolken die gebaseerd is op de 3D Medial Axis Transform (MAT), een skeletachtige representatie van vormen waarin zowel de topologie als de geometrie expliciet wordt gemodelleerd. Alhoewel de 3D MAT al eerder in andere vakgebieden is gebruikt, is de toepassing op geografische puntenwolken nieuw. Voordelen van de MAT ten opzichte van bestaande methodes die veelal 2.5D zijn en op de *boundary*-representatie gebaseerd zijn: 1) het is een volwaardige 3D methode, 2) met behulp van de MAT kan een puntenwolk gestructureerd en opgedeeld worden in losse objecten, 3) er wordt een duidelijke scheiding aangebracht tussen volumes aan de binnen- en buitenkant van objecten, en 4) de MAT bevat attributen (de zgn. *local medial geometry*) waarmee de geometrische eigenschappen van een vorm op compacte wijze beschreven kunnen worden.

In dit proefschrift maak ik drie belangrijke bijdrages. Ten eerst leg ik uit hoe de 3D MAT op robuuste wijze berekend kan worden voor grote geografische puntenwolken. Dit is erg belangrijk aangezien geografische puntenwolken altijd een significante hoeveelheid ruis bevatten door de uitdagende inwinnings omstandigheden. Bovendien wordt de vorm van de MAT zelf in hoge mate beïnvloed door ruis. Ten tweede laat ik zien hoe de MAT georganiseerd kan worden tot een verzameling *medial sheets* die zogenaamde *medial clusters* vormen waarmee een puntenwolk op natuurlijke wijze opgesplitst kan worden in losse objecten. Ten derde demonstreer ik hoe de MAT toegepast kan worden voor slimme puntenwolk simplificatie en visualisatie, zicht analyses, en het detecteren van watergangen en gebouwen.

Door ruis en de beperkte punt dichtheid in geografische punten wolken, werkt de MAT het beste voor objecten met een goed gedefinieerd volume in de punten wolk. Dit zijn bijvoorbeeld gebouwen en vormen in het lanschap zoals dijken. Objecten zoals bomen en lantaarnpalen zijn minder geschikt.

Het belangrijkste resultaat uit dit proefschrift is dat ik bewezen heb dat de 3D MAT een nuttige en praktisch toepasbare methodiek is voor het modelleren van geografische puntenwolken.

Curriculum Vitae



Ravi Peters was born on June 6, 1989 in Rotterdam. In 2007 he graduated from the Marnix Gymnasium (secondary school) with the subject clusters Science & Technology (*'Natuur en Techniek'*) and Science & Health (*'Natuur en Gezondheid'*).

From 2007 to 2012, Ravi studied at the Delft University of Technology. There he obtained his Bachelor of Science degree in Electrical Engineering and his Master of Science degree in Geomatics. His MSc thesis titled *A Voronoi- and surface-based approach fot the automatic generation of depth-contours for hydrographic charts* was awarded second Best graduation thesis by the Hydrographic Society Benelux and led to a publication in the Marine Geodesy journal.

From 2013 to 2017, Ravi conducted his PhD research on *Geographical point cloud modelling with the 3D medial axis transform*, funded by the Netherlands Organisation for Scientific Research (*NWO*) and supervised by Hugo Ledoux and Jantien Stoter at TU Delft.

In 2017 Ravi was hired by the 3D geoinformation group at TU Delft to work as a geo-data scientist for the Amsterdam Institute for Advanced Metropolitan Solutions (*AMS*).

Publications

Automatic identification of watercourses in flat and engineered landscapes by computing the skeleton of a LiDAR point cloud . Tom Broersen, Ravi Peters and Hugo Ledoux. *Computers & Geosciences* 106, 2017, pp. 171–180, doi: 10.1016/j.cageo.2017.06.003

Robust approximation of the Medial Axis Transform of LiDAR point clouds as a tool for visualisation. Ravi Peters and Hugo Ledoux. *Computers & Geosciences* 90(A), 2016, pp. 123–133, doi: 10.1016/j.cageo.2016.02.019

Population estimation using a 3D city model: a multi-scale country-wide study in the Netherlands. Filip Biljecki, Ken Arroyo Ohori, Hugo Ledoux, Ravi Peters and Jantien Stoter. *PLOS ONE* 11(6), 2016, pp. e0156808, doi: 10.1371/journal.pone.0156808

Visibility Analysis in a Point Cloud Based on the Medial Axis Transform. Ravi Peters, Hugo Ledoux and Filip Biljecki. *Eurographics Workshop on Urban Data Modelling and Visualisation 2015*, 2015, pp. 7–12, doi: 10.2312/udmv.20151342

Automatic identification of watercourses in flat and engineered landscapes by computing the skeleton of a LiDAR point cloud . Tom Broersen, Ravi Peters and Hugo Ledoux. *Computers & Geosciences* 106, 2017, pp. 171–180, doi: 10.1016/j.cageo.2017.06.003

A Voronoi-based approach to generating depth-contours for hydrographic charts. Ravi Peters, Hugo Ledoux and Martijn Meijers. *Marine Geodesy* 37(2), 2014, pp. 145–166, doi: 10.1080/01490419.2014.902882

Medial Axis-based point cloud simplification for Digital Surface Models. Ravi Peters. *Workshop on Processing Large Geospatial Data*, Cardiff, UK, 2014.

Generation and generalization of safe depth-contours for hydrographic charts using a surface-based approach. Ravi Peters, Hugo Ledoux and Martijn Meijers. In D. Burghardt (eds.), *Proceedings 16th ICA Generalisation Workshop*, Dresden, Germany, 2013.

ISBN 978-94-6186-899-2