Digital Elevation Models using Triangulated Irregular Networks for Interferometric Synthetic Aperture Sonar

Kristine Baluka Hein

Thesis submitted for the degree of
Master in Informatics: Technical and Scientific Applications
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

1st of August 2019

Digital Elevation Models using Triangulated Irregular Networks for Interferometric Synthetic Aperture Sonar

Kristine Baluka Hein

Acknowledgements

Throughout my studies, I have gotten tremendous amount of support that I am deeply grateful for. At the University of Oslo (UiO), especially in the group Digital Signal Processing and Image Analysis (DSB) at the Department of Informatics, and at the Norwegian Defence Research Establishment (FFI) I have met many wonderful and inspiring people. At the DSB group and at FFI I felt welcomed and had interesting discussions with several people.

My supervisors Prof. Roy Edgar Hansen, Prof. Andreas Austeng, and Assoc. Prof. Solveig Bruvoll has always been present to answer my questions, assisting me, and followed closely the development of this thesis by proofreading and providing feedback. Roy has always been positive and open to many of my questions. He has spent many hours sharing his knowledge of sonar imaging. His shared knowledge was truly valuable for me to understand the signal processing and physical limitations to construct the sonar images. Andreas has always been open to discussions and assisting me with the thesis. He introduced me to the DSB group in such way that I felt included. Solveig shared much of her knowledge of computational geometry and interesting and creative ideas to explore. She also showed genuine interest to discuss and help with proofreading this thesis.

I thank Prof. Michael S. Floater who invited me to many invaluable and inspiring discussions. He also helped me with a much better approach to an implementation that I would have spent many more days on, if not for his help.

Throughout the years at UiO, I have also been lucky to meet Prof. Morten Hjorth-Jensen. His enthusiasm and encouragement has given me the motivation to proceed in learning more about numerical computations.

I thank the students that I have met throughout my studies, and especially Ole-Christian Hagenes and Nirusan Tharmanathan. They made it enjoyable to study and prepare for exams together. They made an including and a motivating atmosphere that was inspiring to be a part of.

I am grateful for my mother and my grandparents that has showed great interest in my studies, especially my mother. She has spent countless of hours listened at me describe the ups and downs throughout the working of this thesis and showed support all the time.

This thesis would not have been finished without all of the fantastic people that have been a part of this journey.
Thank you.

Abstract

Interferometric synthetic aperture sonar (InSAS) produce bathymetric maps with large amounts of data. This introduces challenges in how the data should be stored and represented.

The bathymetric maps are in raster format that can be converted to triangular irregular networks (TINs). A TIN representation of the map can reduce the amount of storage needed. However, it is not straight forward how the conversion should be performed while preserving the main characteristics of the mapped surfaces.

We consider two decimation algorithms and one refinement algorithm that performs a raster to TIN conversion. They are point selection algorithms and based on Delaunay triangulations. We have implemented the algorithms and adapted them such that they also base their point selection on the coherence from an InSAS.

The algorithms were applied on three classes of data: synthetic data without coherence, synthetic data with coherence, and real bathymetric data from an InSAS. The algorithms returned reduced data sets that contained less than ten percent of the input data and at the same time preserved essential features of their inputs. We found that the refinement algorithm performed the best to convert a raster to TIN in overall.

List of abbreviations

AUV	Autonomous underwater vehicle
CGAL	Computational Geometry Algorithms Library
DEM	Digital elevation model
GIS	Geographical information systems
InSAR	Interferometric synthetic aperture radar
InSAS	Interferometric synthetic aperture sonar
SAS	Synthetic aperture sonar
SNR	Signal-to-noise ratio
TIN	Triangular irregular network
(w)RMSE	(weighted) Root mean squared error
(w)AT1	(weighted) Adaptive thinning 1
(w)AT3	(weighted) Adaptive thinning 3

Contents

1 Introduction

Today, underwater technology is capable to produce maps of seabeds at an impressive level of detail and at the same time cover large areas. However, there are still many areas that are not completely mapped (Mayer et al., 2018, p. 2). Since the surface of the Earth consist of more than 70 percent of water (Lurton, 2010, p. 1) and only a fraction of it has been mapped, there is still a lot of research and surveillance that remains to get a better insight of the nature to our planet.

An interferometric synthetic aperture sonar (InSAS) can produce maps, or digital elevation models (DEMs), that covers large areas and at the same time represent finer details of the imaged seabed (Hansen, 2011, p. 3). The maps can be used for applications such as analysis of the seabed and assist autonomous underwater robots that performs surveillance of the seabeds to navigate through its surroundings (Leonard and Bahr, 2016, p. 349). The amount of data in the maps of the seabeds can be massive due to their high level of detail and large area coverage. This imposes challenges to how the maps should be stored. Depending on which application the data are used for, the high level of detail in the data can be redundant. For instance, if an underwater robot needs the data to navigate through its environment, it is sufficient for it to keep a simplified map that captures the essential characteristics of its surroundings.

Surface simplification algorithms can be used to reduce the amount of data that contains measurements over a surface such as a seabed. Triangular irregular networks (TINs) are commonly encountered in geographical information systems (GIS) as the preferred approach to efficiently represent terrain surfaces (Heywood et al., 2011, p. 94). Many algorithms that performs surface simplification based TINs has been proposed throughout the years such as (Fowler and Little, 1979), (Garland and Heckbert, 1995), (Soommart and Paitoonwattanakij, 1999), (Demaret et al., 2005), (Zhou and Chen, 2011), (Sun et al., 2018). Such algorithms has also been applied to seabed data (Canepa et al., 2003), (Dokken et al., 2015), (Maleika et al., 2018). Surface simplification is still an ongoing research, where for instance the popular and well-established C++ library The Computational Geometry Algorithms Library, also known as CGAL, recently got extended by a surface simplification algorithm, see (Alliez et al., 2019). Similar ideas that follows the general ideas of surface simplification, is also considered in other research fields such as solving partial differential equations, see e.g (Quarteroni, 2009, Sec. 4.6), (Nochetto and Veeser, 2011).

1.1 Problem statement

We consider a map of a seabed, a *bathymetric map*, obtained from an InSAS. An InSAS can return detailed bathymetric maps that spans over hundreds of meters. In addition, the data in the bathymetric maps could also be of various quality. A possible quality assessment of the measurements is the *coherence* produced by the InSAS. Data associated with low coherence can reduce the overall quality of the bathymetric map and should ideally be removed and compensated for.

Our main focus has been to apply a selection of TIN-based simplification algorithms on a bathymetric map produced by an InSAS. In addition, we have adapted the algorithms such that they take into account the associated coherence to its input map. To our knowledge, this has not been done before. The desired end product is a simplified map that is significantly reduced in amount of data but still preserve the main features of the input map.

First, we describe how bathymetric maps are produced, how the chosen simplification algorithms and how we evaluated the simplified models. We applied the simplification algorithms at three classes of data; synthetic data not associated with coherence, synthetic data associated with coherence, and a bathymetry with an associated coherence map from an InSAS. The algorithms managed to return point sets that contained on less than ten percent of the input data and at the same time resembled the given surface when triangulated. One of the algorithms managed to simplify a real bathymetric map such that the simplified surface was formed by approximately one percent of data from the map. Approximately 73 percent of the surface deviated from the input map by less than or equal to 0.1 meters, which showed that most of the important characteristics to the input surface was preserved after the simplification.

1.2 Outline

We have organized the thesis as follows:

Chapter 2 - Background
In this chapter, we present and describe the main concepts of sonar imaging and digital terrain modeling. Thereafter, we present the basic theory of triangulations and Delaunay triangulations. We end this chapter by describing which error assessments we have chosen to evaluate our simplified surfaces.

Chapter 3 - Surface simplification
The surface simplification algorithms that we consider are described in this chapter. We present how we have adapted the algorithms to consider an associated quality measure of the input data.

Chapter 4 - Results
This chapter contains the results that we have obtained from our studies. We describe how we tested our implementations and examined how the algorithms performs applied to synthetic data without associated coherence, synthetic

data with associated coherence, and a bathymetry and its associated coherence acquired from a synthetic aperture sonar.

Chapter 5 - Discussion
We discuss our main results in this chapter.

Chapter 6 - Conclusion
We summarize our main results and conclude our research. Several proposals for future work are presented in this chapter.

1.3 Own contributions

During the writing of these thesis, we have developed new methods and algorithms. These are:

- **Implementations in MATLAB**
  To our knowledge, no program of the algorithms that we considered were written in MATLAB. We have implemented the algorithms, without the usage of any pre-developed code except for the functions available in MATLAB. Our implementations constitutes 1127 lines of codes together, not considering the scripts that were made for testing and generating the results.

- **Weighted adaption of the selected algorithms** The algorithms that we consider, are based on Delaunay triangulations. There exists methods for constructing triangulations with respect to the weighting of the data, which are called *data-dependent* triangulations. However, we could not find any algorithms that simultaneously considered the weighting to the data and its corresponding Delaunay triangulation. Our weighted adaption of the presented algorithms in this thesis are thus novel, as far as we can see.

- **Using coherence in the simplification algorithms**
  We could not find any research where coherence has been incorporated into a simplification process.

- **Feature extraction based on variance and quadtree split**
  We could not find any literature that presented the feature extraction method as described in Section 3.2 that we considered in this thesis.

- **Scatter plot over the longest edges and smallest angles to the triangles in a triangulation**
  To our knowledge, there is no research where the longest edges and the smallest angles to the triangles in a triangulation has been found and considered in an analysis of the occurrence of long and thin triangles in a triangulation.

2 Background

Interferometric synthetic aperture sonar can produce bathymetric maps in centimeters resolution. Delaunay triangulations are commonly encountered in geographical information systems as a preferred way of constructing digital elevation models. They have properties that tend to avoid badly shaped triangles and can be efficient to use since vertex insertion and deletion requires local updates of the triangulations. In this chapter, we present what an interferometric synthetic aperture sonar is and how it produces a bathymetric map. We also present what properties makes Delaunay triangulation popular in geographical information systems and how we calculate the error between the approximated surface and the given surface.

2.1 Synthetic aperture sonar

Sonar is a device that uses sound to gather information of its environment (Rossing, 2014, p. 6). In this context, sound conveys important information and therefore often referred to as a signal or an acoustic signal. Sonar is widely used in the ocean, as sound propagates far better than electromagnetic waves in seawater (Lurton, 2010, pp. 1 - 3).

There are two different kinds of sonar systems; passive and active (Lurton, 2010, p. 9). Passive sonars only measures the sound emitted from its environment. Active sonars emits and measures the echoed sound from its environment. The objects that reflects the transmitted sounds, are usually referred to as *targets* or *scatterers*. Active sonars use the reflected signals to retrieve information from the scatterers.

Synthetic aperture sonar (SAS) is an active imaging sonar. An *aperture* is a group of sensors that measures any incoming signal (Johnson and Dudgeon, 1993, p. 59). Its physical construction, along with the waveform of the transmitted signal, is crucial in an imaging system. How the imaging system is affected is described in e.g (Johnson and Dudgeon, 1993, Chap. 3). A SAS can use the received signals to produce a sonar image over a desired region. The sonar image contains estimated reflectivity of the seabed. An *interferometric* SAS (InSAS) is a mapping sonar that can use its sonar images to produce seabed depth maps, *bathymetric maps*, of the imaged scene. This will be discussed further in Section 2.1.3.

There are other types of seabed mapping sonars, including multibeam echosounders and interferometric side-scan sonars (Kuperman and Roux, 2014, p. 197). We refer to (Lurton, 2010, pp. 351 - 363) for a description of multibeam echosounders.

Side-scan sonars and SAS are closely related (Hansen, 2011, p. 7). The main difference is how the images are constructed. An interferometric side-scan sonar constructs thin strips of an imaged seabed and thereafter stacks the strips on top of each other to construct a sonar image of the seabed (Blondel, 2009, pp. 37 - 38). A SAS constructs the sonar images by combining all of the received signals into one image. We will discuss the image formation of a SAS in more detail in Section 2.1.2 and refer to (Blondel, 2009) for a comprehensive description of side-scan sonars.

SAS can produce images with spatial resolution independent of the distance between the seabed and the sonar, whereas side-scan sonars cannot. The spatial resolution of an image is defined as the lower bound on how close two scatters can be to each other without being imaged as a single scatterer (Franceschetti and Lanari, 1999b, p. 14). SAS can produce sonar images with resolution in centimeters. This is possible because of its synthetic aperture. The physical aperture of the SAS that has produced the map over a seabed that we have considered (see Section 2.1.1) consists of three arrays; two receiver arrays and one transmitter array. The receiver arrays are separated by a vertical distance. The transmitter array can be placed between the receiver arrays. The transmitter array repeatedly transmits acoustic signals that are often referred to as *pings* during a survey. The transmitted signals are reflected from the seabed and recorded by the receiver arrays.

In SAS, it is possible to construct sonar images *as if* the data was gathered from a physical aperture having similar imaging capabilities as its synthetic aperture. The synthetic aperture is constructed on how the pings are processed, see Section 2.1.2 and the references therein.

2.1.1 The imaging system

An InSAS can be mounted on an *autonomous underwater vehicle* (AUV). An AUV is a robot that can gather data underwater and navigate by itself based on the information it has gathered during a survey. In (Leonard and Bahr, 2016, p. 341) it described in more detail what an AUV is and how it manages to navigate under water. We consider data that has been collected by a HUGIN AUV developed by Kongsberg Maritime and Norwegian Defence Research Establishment (FFI). The InSAS that was mounted on the HUGIN AUV is a HISAS 1032. The data that we consider are from the same survey that is described in (Hansen et al., 2018, p. 344, p. 346). Figure 2.1 shows a HUGIN with an InSAS.

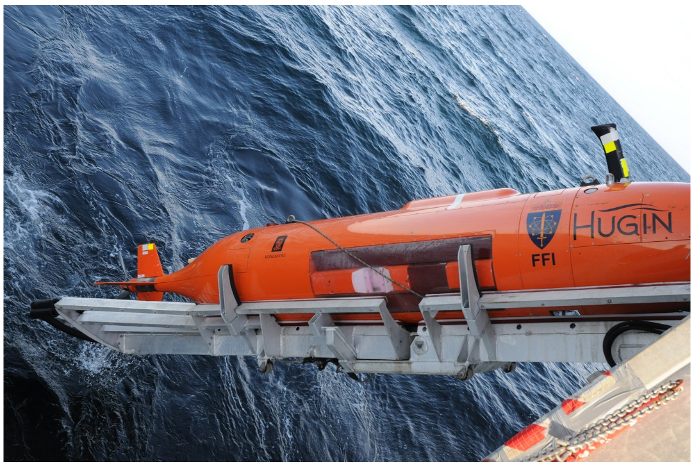


Figure 2.1: Image of a HUGIN AUV equipped with HISAS 1032 interferometric SAS from the FFI research vessel H. U. Sverdrup II during the *Marex17* scientific trials outside Bergen, Norway, in March 2017. The HISAS is the brown rectangular objects forming a "H"-looking shape mounted on HUGIN. Photo courtesy of FFI and used with permission.

The theoretical resolution of the sonar images acquired from HISAS 1032 is approximately 3×3 centimeters and the bathymetric maps 18×18 centimeters (Hansen et al., 2018, p. 346). The bathymetric map has poorer resolution than the sonar images as a consequence of how the bathymetric map is generated, see Section 2.1.3 and the references therein. For the technical specifications of the HUGIN AUV, we refer to (Hansen et al., 2019, p. 10).

2.1.2 Image construction

The samples of the received signals are processed and stored as complex values. More details about the processing of the acquired signals can be found in (Richards, 2005, pp. 115 - 158). The amplitude represents the amount of the reflected signal. The phase contains information that one can use to retrieve information of the geometry to the object that scattered the transmitted signal (Massonnet and Souyris, 2008, p. 60). It is deterministic in the sense that it will remain the same if the target is located at the same position, the properties of the medium does not change, and it is observed from the exact same location (Hanssen, 2001, p. 35).

We describe how the sonar constructs an image with pixels $\gamma[i, j]$ that represents the signals from a small area at the imaged seabed. A generated map over a seabed is based on sonar images from the seabed. To understand how an InSAS forms maps over seabeds, we first need to consider how sonar images are constructed.

A signal $s_k^{(p)}(t)$ hits the k-th sensor element at ping p. The sensor samples the signal over a time. The signal is then *pulse compressed*. Pulse compression is a processing step that takes the waveform of the transmitted signal and correlates it with the received signal, see (Richards, 2005, Chap. 4). This is done to increase the *signal-to-noise ratio* (SNR) (Massonnet and Souyris, 2008, pp. 125 - 126). Noise can be present due to other signals in the environment, disturbances by the sensors and the vehicle or interference between the signals (Lurton, 2010, p. 123).

After the pulse compression, each sensor element k has a discrete time series $s_k^{(p)}[:] = \left\{ s_k^{(p)}[i], s_k^{(p)}[1], \ldots, s_k^{(p)}[n-1] \right\}$ of n samples. Every i-th sample is such that $s_k^{(p)}[i] \in \mathbb{C}$ and $s_k^{(p)}[i] = s_k^{(p)}(i \cdot \Delta t)$ for $i = 0, \ldots n-1$, where $\Delta t = \frac{T}{n-1}$ and T is the duration of the received signal. An image can be formed by *backprojection*[1] (Hansen, 2011, p. 4). The image is formed on a rectangular grid that represents the true region being imaged. Consider a grid $\mathcal{I}$ that is $I_M \times I_N$ large where I_M and I_N are positive integers and $\mathcal{I} = \big\{(i,j) : i \in \{0, 1, \ldots, I_M - 1\}$ and $j \in \{0, 1, \ldots, I_N - 1\}\big\}$. A set of signals that covers the region of interest are then chosen. Let $\mathcal{P} = \{p_1, p_2, \ldots, p_S\}$ be the set of S integers, each representing a ping. The pings are such that all sensor elements has a time series sampled from the imaged region. These signals can be used to estimate the reflected signal at each coordinate $(i,j) \in \mathcal{I}$. Each coordinate (i,j) from $\mathcal{I}$ represents the average reflection of a small area centered at coordinate (x_i, y_j), a *resolution cell* (Richards, 2005, p. 21), in the measured region of interest. Figure 2.2 illustrates how the grid $\mathcal{I}$ represents the imaged region.

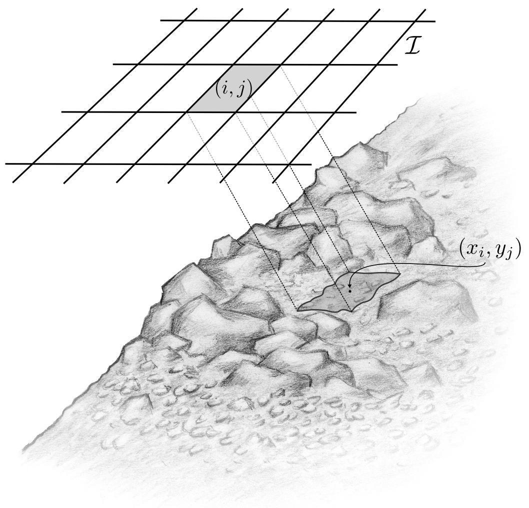


Figure 2.2: The value at position (i,j) in the grid $\mathcal{I}$ represents the average of the reflected acoustic waves from a resolution cell centered at position (x_i, y_j) at the seabed.

For all $p_m \in \mathcal{P}$ the sensor elements has sampled information of the region of interest. The signal $s_k^{(p_m)}[:]$ can be used to obtain an estimate on how much the true signal has been scattered from each resolution cell. Each resolution cell is centered at (x_i, y_j) within the desired region to image. To obtain the estimated reflectivity of the imaged scene, first the time $t_k^{(p_m)}(x_i, y_j)$ the transmitted wave spent from the k-th sensor mounted on the platform to (x_i, y_j) and back has to be found. This is possible using the relation

$$t_k^{(p_m)}(x_i, y_j) = s/c, \tag{2.1}$$

where s is the distance the wave traveled and c the speed of the wave. The value $s_k^{(p_m)}(t_k^{(p_m)}(x_i, y_j))$ contains thus information about the amount of the reflected wave.

[1]Also referred to as *Delay-And-Sum beamforming* (Hansen, 2011, p. 5).

Since each k-th sensor only has a discrete representation $s_k^{(p_m)}[:]$ of the signal $s_k^{(p_m)}(t)$, it is not certain that the element has a sample recorded at $t_k^{(p_m)}(x_i, y_j)$ exactly. It is possible to get an exact representation of $s_k^{(p_m)}\left(t_k^{(p_m)}(x_i, y_j)\right)$, given that the samples are well-sampled (Manolakis and Ingle, 2011, pp. 298 - 299). Due to practical limitations such as computational time, a simpler interpolation method such as linear interpolation can used be to approximate $s_k^{(p_m)}\left(t_k^{(p_m)}(x_i, y_j)\right)$.

After having obtained a representation $\hat{s_k}^{(p_m)}[i,j]$ of $s_k^{(p_m)}\left(t_k^{(p_m)}(x_i, y_j)\right)$, a summation over the elements is performed. This gives an estimate of the reflected acoustic wave $\gamma^{(p_m)}$ at (i,j):

$$\gamma^{(p_m)}[i,j] = \sum_k w_k \hat{s_k}^{(p_m)}[i,j],$$

where w_k is some weight chosen accordingly on how much the recorded signal at the k-th element should contribute to the final estimate. The summation ensures to amplify the dominant trends in the measured signals, thus yielding a more probable estimate of the reflected acoustic wave at (x_i, y_j) at ping p_m (Johnson and Dudgeon, 1993, p. 112).

The final image γ that represents the reflected signals over a region of interest, is constructed by summing over all pings from $\mathcal{P}$,

$$\gamma[i,j] = \sum_{m=1}^{S} \gamma^{(p_m)}[i,j]. \tag{2.2}$$

The synthetic aperture is constructed when the pings are combined as in Equation (2.2) (Hayes and Gough, 2009, p. 207). The synthetic aperture is therefore dependent on how the pings were sampled. The platform has to move such that it manages to sample enough pings to fulfill a spatial sampling criterion (Hansen, 2011, pp. 8 - 10). This implies several restrictions to the platform, see (Bruce, 1992), (Lurton, 2010, pp. 347 - 351), (Hansen, 2011, pp. 12 - 14).

2.1.3 Estimation of height using synthetic aperture interferometry

SAS interferometry uses the phase difference between two images formed as described in Section 2.1.2. The images are of the same scene, but observed from two or more slightly different positions (Franceschetti and Lanari, 1999a, p. 167). The phase difference between such images can convey important information of the geometry to the area that otherwise could be difficult to measure (Hanssen, 2001, p. 35).

Consider the case where only one receiver is mounted on a platform and only one image is constructed of a scene. Assume also there are two targets in the scene that are located at the same range, separated by a horizontal distance and have different elevations. Then the receiver would perceive the targets as equal (Hanssen, 2001, p. 35), since the reflected acoustic waves will spend the same amount of time from the target to the sensor arrays. This will cause an ambiguity of the measurement, as we do not have information that reveals exactly where the wave was reflected from (Franceschetti and Lanari, 1999a, pp. 167 - 170).

To overcome the ambiguity of where the acoustic wave was reflected from, we need at least two images of the same scene observed from slightly different positions vertically (Jakowatz et al., 1999, p. 285). In addition, one has to ensure that the imaging conditions are the same. For simplicity, we discuss the case where only two images γ_1 and γ_2 are available. The images are first co-registered. This is done by taking either γ_1 or γ_2 as a reference image and move the pixel positions in the other image such that each pixel represents the same resolution cell relative to the reference image (Franceschetti and Lanari, 1999a, pp. 177 - 185). Without the loss of generality, we let γ_1 be the reference image and $\widehat{\gamma_2}$ the transformed image. We can now compute the phase difference between them to obtain an *interferogram* of the scene. The phase differences in the interferogram can thereafter be used to estimate the elevations to the imaged surface.

Let

$$\gamma_{1,2} = (\gamma_1 \odot \widehat{\gamma_2}^*) \star h, \tag{2.3}$$

where $\odot$ is the Hadamard product[2], $\widehat{\gamma_2}^*$ be the complex conjugate of $\widehat{\gamma_2}$, $\star$ the convolution operator, and h some chosen two-dimensional lowpass filter. An interferogram φ of γ_1 and $\widehat{\gamma_2}$ is defined as (Hanssen, 2001, p. 92):

$$\varphi = \operatorname{Arg}\{\gamma_{1,2}\},$$

where $\operatorname{Arg}\{\cdot\}$ takes the principal phase of its argument.

The lowpass filter h of a given size is applied to the product $\gamma_1 \odot \widehat{\gamma_2}^*$ in Equation (2.3) to dampen the variability of SNR in γ_1 and $\widehat{\gamma_2}$. The variability of SNR is damped within a local neighborhood defined by h to every pixel in $\gamma_1 \odot \widehat{\gamma_2}^*$. Hence the estimated becomes more probable, but at the cost of poorer spatial resolution related to the size of the filter, see (Sæbø, 2010, pp. 121 - 127). This is why InSAS systems, like the one described in Section 2.1.1, provides sonar images having better resolution than the bathymetric maps.

The interferogram can provide us insight in the imaging geometry between the platform and the scene during the data acquisition. As phase is a deterministic quantity, the phase difference is related to at which position the two images were observed from (Massonnet and Souyris, 2008, p. 179). There will be a difference in path length the reflected acoustic waves has traveled as a consequence of the different locations of observation. By knowing how far apart each platform has been during the two different acquisitions, one can derive geometrically how the height affects the difference in path length, see (Sæbø et al., 2013, p. 4451).

Since phase is 2π-periodic, there is no guarantee that it truly represents the changed phase reflected from a target. A phase $\widehat{\phi}$ sampled from the incoming signal and

[2] The *Hadamard product* between two matrices A and B having same dimensions is a matrix $A \odot B = C$ with the same dimension as A and B where each element C_{ij} in C is such that $C_{ij} = A_{ij}B_{ij}$ (Horn and Johnson, 2013, p. 477).

represented in the interval $(-\pi, \pi]$ (Franceschetti and Lanari, 1999a, p. 172) could actually be equal to $\widehat{\phi} + 2\pi n$ for a non-zero integer n. The interferogram might therefore not represent the differences between the sampled signals. Thus it has to be further processed to find an estimate of the true difference between the phases. This is called *phase-unwrapping* and is thoroughly covered in e.g (Ghiglia and Pritt, 1998).

2.1.4 Coherence as a quality assessment

The interferogram φ is found from the coregistered images γ_1 and $\widehat{\gamma_2}$. Each pixel in γ_1 and $\widehat{\gamma_2}$ are supposed to represent the same resolution cell. Under ideal circumstances, the only difference between the pixels is in their phase. The difference in phase should only convey information of the angle difference between the receiver arrays. However, the pixels might differ in both phase and amplitude due to several reasons. There could for instance be unwanted disturbances in the acquired signals, wrong assumptions of the traveled paths to the signals, and faulty sensor elements. We want to know at which pixel locations (i, j) in γ_1 and $\widehat{\gamma_2}$ those signals contributed to, as they will contribute to erroneous estimates in elevation at $\varphi[i, j]$. One common way to quantify the quality of φ is through the *coherence* of γ_1 and $\widehat{\gamma_2}$ (Franceschetti and Lanari, 1999a, p. 173), (Massonnet and Souyris, 2008, p. 186).

The coherence can be derived from the complex coherence by taking the absolute value of it. The complex coherence has to be estimated, as the definition of coherence requires the expectation over all possible realizations of γ_1 and $\widehat{\gamma_2}$ (Hanssen, 2001, p. 96). Let $\gamma_{1,2}$ be defined as in Equation (2.3). The estimated complex coherence $\hat{\chi}$ at position (i, j) can be found by (Ghiglia and Pritt, 1998, p. 66):

$$\hat{\chi}[i,j] = \frac{\sum_{s=-n}^{n}\sum_{t=-m}^{m}\gamma_{1,2}[i+s,j+t]}{\sqrt{\sum_{s=-n}^{n}\sum_{t=-m}^{m}|\gamma_1[i+s,j+t]|^2}\sqrt{\sum_{s=-n}^{n}\sum_{t=-m}^{m}|\widehat{\gamma_2}[i+s,j+t]|^2}}, \tag{2.4}$$

where the summations are performed over a neighborhood of size $(2n+1)\times(2m+1)$ and $|\gamma_1|^2 = \gamma_1\gamma_1{}^*$ with $\gamma_1{}^*$ being γ_1 complex conjugated (similar definition holds for $|\widehat{\gamma_2}|^2$ also).

From the estimated complex coherence $\widehat{\chi}$, the estimated coherence for every $(i, j) \in \mathcal{I}$ can be found by $\widehat{C}[i, j] = |\widehat{\chi}[i, j]|$. By using $\widehat{C}[i, j]$, an approximation of the SNR$[i, j]$ to each pixel at every $(i, j) \in \mathcal{I}$ can be found by using the relation (Zebker and Villasenor, 1992, p. 951):

$$\widehat{C}[i,j] = \frac{\text{SNR}[i,j]}{\text{SNR}[i,j]+1},$$

which gives that

$$\text{SNR}[i,j] = \frac{\widehat{C}[i,j]}{1-\widehat{C}[i,j]}. \tag{2.5}$$

The SNR$[i, j]$ can provide us information on how certain the measurement at (i, j) is. The SNR$[i, j]$ is related to the variance $\sigma_\tau^2[i, j]$ of the estimated time from Equation (2.1) (Sæbø, 2010, p. 65) by

$$\sigma_\tau^2[i,j] \propto \frac{2\,\mathrm{SNR}[i,j]+1}{2\,(\mathrm{SNR}[i,j])^2}.$$

The σ_τ^2 is proportional to the variance σ_z^2 in estimated depth (Sæbø, 2010, p. 37), as depth estimation is dependent on correct time measurement. In weighted least squares approximation, the optimal weighting of the data that are being approximated are one divided by their variance (Kay, 1993, pp. 225 - 226). Inspired by this, we have decided to use the weights

$$\begin{aligned} w[i,j] &= \left(\frac{2\,\mathrm{SNR}[i,j]+1}{2\,(\mathrm{SNR}[i,j])^2}\right)^{-1} \\ &= \frac{2\,(\mathrm{SNR}[i,j])^2}{2\,\mathrm{SNR}[i,j]+1} \end{aligned} \tag{2.6}$$

in our chosen error assessment and algorithms described in Section 2.4 and Chapter 3 respectively.

2.1.5 Characteristics of the sonar images

The principles and ideas in InSAS are similar to its radar counterpart, interferometric synthetic aperture radar (InSAR) (Gough and Hawkins, 1997, p. 344), (Lurton, 2010, p. 347). However, there are some differences that separates the images they produce (Hayes and Gough, 2009, p. 213), (Hansen et al., 2011, p. 3677). We present a short description of what characterizes InSAS and refer to (Griffiths, 1997), (Sæbø and Hansen, 2010), (Hansen et al., 2011, pp. 3680 - 3685) for a more detailed description of the differences between InSAS and InSAR.

Navigation and the speed of sound

Navigation under water is difficult since geographical position system is unavailable (Gough and Hawkins, 1998, p. 212), (Hansen et al., 2011, p. 3680). InSAS is also more prone to errors in estimated position of the platform compared to InSAR. This is because the speed of the platform with an InSAS relative to the speed of sound, is much higher that the speed of platform with an InSAR relative to the speed of light (Lurton, 2010, p. 351). Wrong estimations of the whereabouts to an InSAS leads to images, and thereby bathymetric maps, where the imaged region can suffer of geometric distortions (Hansen et al., 2011, pp. 3682 - 3683).

Acoustic shadows

The side-scan sonars and SAS transmits pings that forms a low grazing angle between the emitted wave and the surface. There is therefore no guarantee that the transmitted signals will be reflected back to the sensor. In the resolution cells where this happens, the sensors collects little to no data that are representative of the seabed and therefore form acoustic shadows, see Figure 2.3. The shadows can be used for instance to search for man-made objects such as mines at the seabed (Lurton, 2010, p. 341).

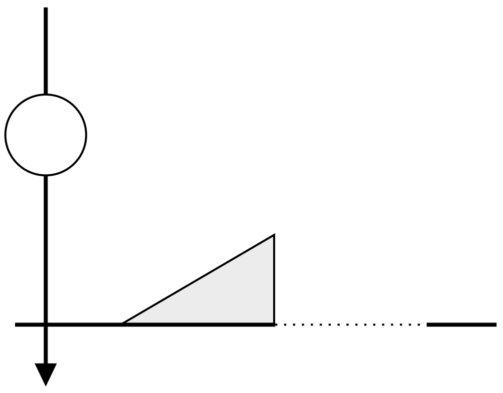


Figure 2.3: There could be some areas that the emitted signal from the platform is not reflected from. Here, the dotted horizontal lines is the area where the emitted signal is not reflected from the seabed.

Multipath

An acoustic wave might change its direction multiple times before arriving at the receivers (Lurton, 2010, p. 30). If not compensated for, it could be difficult to find at where the acoustic wave was reflected from the seabed. This phenomena, called *multipath*, occurs especially in shallow waters, as the interface between water and air can reflect acoustic signals from the seabed (Gough and Hawkins, 1998, pp. 214 - 215), (Hansen et al., 2011, pp. 3684 - 3685). The emitted signals may also be reflected multiple times by objects at the seabed. In Figure 2.4 we show how multipath can affect the acoustic waves.

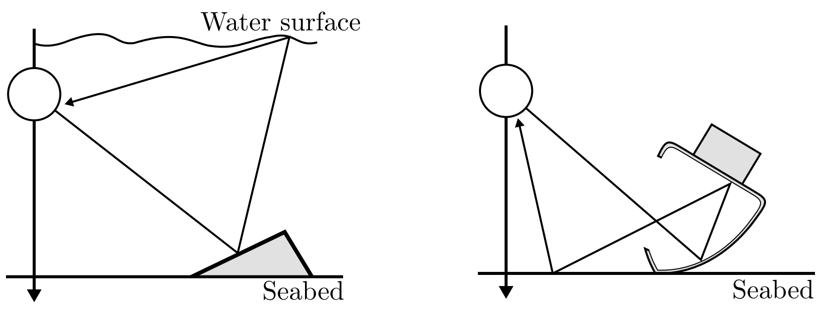


Figure 2.4: Possible scenarios where the acoustic wave does not travel directly back to the platform. *Left:* Multiple path to the signal due to the shallow water. *Right:* Multiple paths to the signal due to complex objects at the seabed such as ship wrecks.

Layover and foreshortening

Layover and foreshortening are geometric distortions related to the topography of the imaged surface. Layover occurs when the grazing angle α to the platform is greater than the angle β to the normal vector to the surface normal (Richards, 2006, pp. 10 - 11). This is the case for the platform at elevation A in Figure 2.5. The scatterers are imaged in reverse order spatially, as some are closer to the platform than others. Foreshortening occurs when α is less than β (Richards, 2006, pp. 10), which is the case for the platform at elevation B in Figure 2.5. The spatial distribution of the scatterers gets compressed in the image.

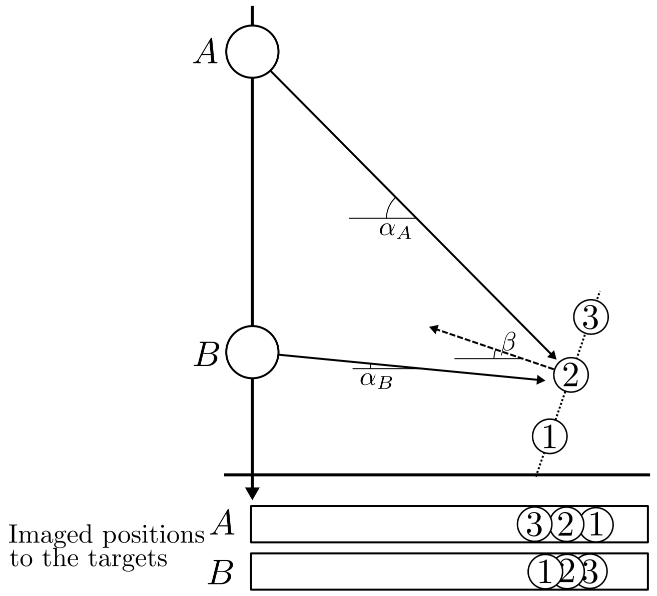


Figure 2.5: Layover and foreshortening. Three targets located at an inclination, shown as the dotted line through the targets 1, 2, and 3. Layover occur in the imaged region from position A since $\alpha_A > \beta$. Foreshortening occur in the imaged region from position B since $\alpha_B < \beta$ This figure is inspired by Fig. 6 in (Richards, 2006, p. 10).

Variation in speed of sound and refraction

The speed of sound under water can change as a function of depth, as the physical properties could be different (Lurton, 2010, p. 13). By Snell-Descartes law (Lurton, 2010, pp. 47 - 49), this will make the acoustic waves “bend” an therefore refracted, see Figure 2.6. The acoustic waves will spend more or less time traveling compared to if it traveled in a straight path.

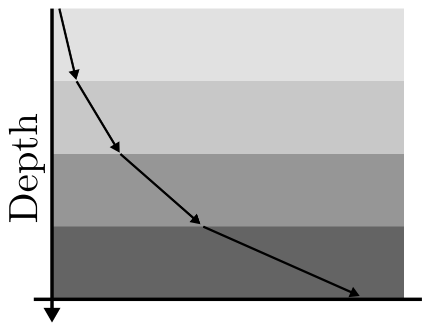


Figure 2.6: How the path of the signal, illustrated as black arrows, can change in water with different physical properties. The colored boxes illustrates water with different properties.

2.2 Digital elevation modeling

A digital elevation model (DEM) represents a set of measured elevations of a surface (Hengl and Evans, 2009, p. 40). The measurements can be acquired from remote sensing devices such as sonars and radars. The elevation data consists of points in $\mathbb{R}^3$ where each point p is a triplet (x, y, z) that represents a measured elevation z at the planar coordinates x and y. The representation of the measured surface is dependent on which applications the data are supposed to be used for. It is possible to convert the represented data from one representation to an another representation. The most common representations of elevation data are grid-based or based on a collection of triangles (Li et al., 2004, p. 68), (Haverkort and Toma, 2014, p. 3).

A set of measurements that are represented on a grid is a *raster* DEM (Heywood et al., 2011, p. 92). A raster DEM is constructed by a collection of regularly spaced quadrilaterals of equal shape and size. Each of the quadrilaterals contains one value that represents a measured elevation. Typically, squares are used (Wilson, 2018, p. 25) as they can simply be stored as matrices (Li et al., 2004, p. 70). Any data set of points that are regularly spaced in the plane is a raster DEM. If a data set contains points that are irregularly spaced in the plane, interpolation of the elevations to the data points has to be performed to yield a gridded representation of the point set (Li et al., 2004, pp. 82 - 83). A raster DEM is therefore strict in the sense that it requires to represent the elevations to each point in a regular manner. This requirement can also make a raster DEM store more points than necessary.

A *triangular irregular network* (TIN) can represent a surface that consists of regions that varies in complexity without storing more points than necessary (Heywood et al., 2011, p. 94). A TIN forms a collection of triangles based on the data set such that it produces a *triangulation* of it. The triangles in a triangulation has to satisfy some criteria, see Section 2.3. It is common that TINs are constructed using *Delaunay triangulations* (Van Kreveld, 2017, p. 1566). We consider TINs constructed by Delaunay triangulations in this thesis and describe what a Delaunay triangulation is in Section 2.3.1. The data points that forms a TIN can be irregularly spaced without further processing, as opposed to grids where interpolation between the data points has to be performed. This is why the network is irregular. A TIN is also a *vector* based representation of the data set. A vector based representation consist only of information on how the points are spatially related (Li et al., 2004, p. 66). In the case of TINs, only the triangles that are formed by the data points are stored. Because of this, a TIN could be more convenient to use for storage and further analysis of a point set (Van Kreveld, 1997, p. 46). We obtain a *raster* TIN when we interpolate the elevations associated with the vertices to each triangle in the network. We present in Section 2.3 how we chose to compute the interpolated elevations.

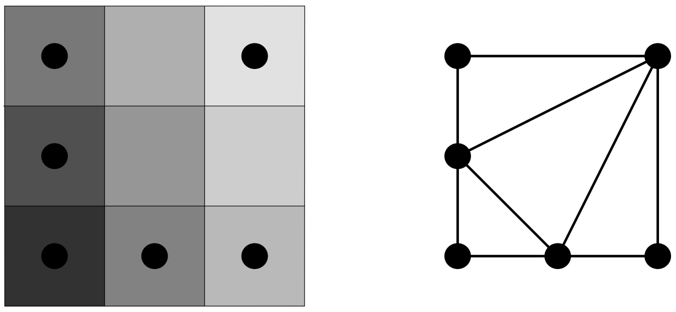


Figure 2.7: A raster representation and TIN representation of a point set where the points are drawn as black dots. *Left:* A raster representation of the point set. The raster is based on a square grid. Each point in the grid is assigned a value that is visualized as a gray level intensity. In the squares without points, values are assigned based on e.g an interpolation scheme. *Right:* A vector TIN representation of the point set.

We have already encountered a raster DEM in Section 2.1.3. The bathymetric map from an InSAS is a raster DEM since the two sonar images γ_1 and γ_2 the map is constructed of are defined on a grid $\mathcal{I}$. The bathymetric map from the InSAS that we consider is stored as a matrix, hence the grid consist of squares. The process of acquiring data and construct the bathymetric map constitutes Step 1, Step 2, and Step 3 in Figure 2.8. The bathymetric maps produced by the imaging system described in Section 2.1.1 can have centimeters in resolution and at the same time cover areas that spans tenths or hundreds of meters. Hence, a the maps can easily contain millions of data points. We chose to look at a selection of methods that converts the raster bathymetric map into a TIN representation. This is Step 4 in Figure 2.8. In general, it is not given how the construction from a raster DEM to TIN should be performed and is still an ongoing research. We therefore consider Step 4 throughout this thesis.

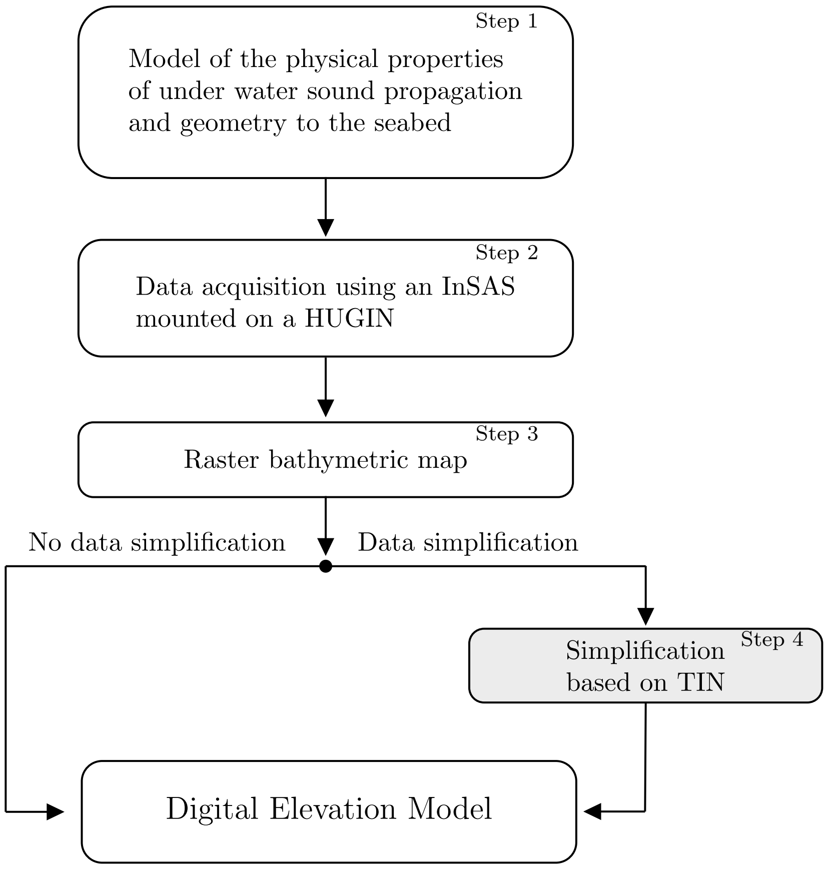

Step 1
Model of the physical properties of under water sound propagation and geometry to the seabed
Step 2
Data acquisition using an InSAS mounted on a HUGIN
Step 3
Raster bathymetric map
No data simplification
Data simplification
Step 4
Simplification based on TIN
Digital Elevation Model

Figure 2.8: The processing chain that we consider to represent a sampled seabed. Our emphasis is on Step 4.

2.3 Triangulations

In GIS it is common to have a set of discrete samples that represents a surface. In most cases, it is desirable to use the samples to get a representation of the sampled surfaces for applications such as visualization, analysis of the sampled terrain, and efficient representation of the data. Triangulations based on the acquired samples are often encountered in GIS. They are flexible in representing point sets that varies in density and at the same time preserve important features of the terrain (Wilson, 2018, p. 26).

Consider a set P of n points. Each point $p_i \in P$ has a location in the plane given by the coordinates (x_i, y_i) where x_i, $y_i \in \mathbb{R}$. Each coordinate pair (x_i, y_i) is associated with an elevation z_i as illustrated in Figure 2.9. Each point p_i in P can be represented as a triplet, $p_i = (x_i, y_i, z_i)$. We assume that there is only one point in P that is located at (x_i, y_i) such that z_i is unique.

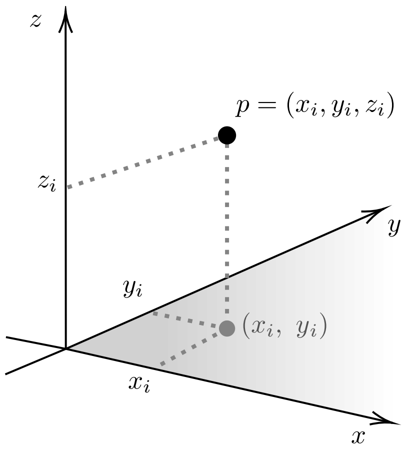


Figure 2.9: A point $p_i = (x_i, y_i, z_i)$ and its position (x_i, y_i) in the plane. The point represents a measured elevation z_i at the coordinates (x_i, y_i).

A triangulation is composed of triangles. A triangle T is formed by connecting three points. These points are called *vertices*. In this thesis, however, we will interchangeably use points and vertices. If the points are collinear[3], then T is *degenerate* (Cheng et al., 2012, p. 26). The points are connected by straight lines that are called *edges*. A triangulation can therefore be regarded as a collection of vertices and edges. Figure 2.10 shows a triangulation of a point set.

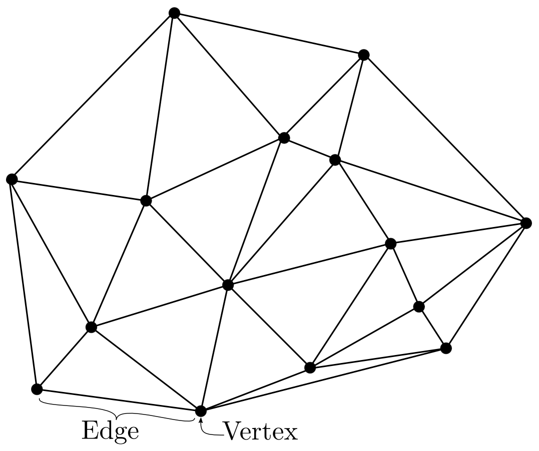


Figure 2.10: A triangulation of a point set. Each point in the point set is illustrated as the black markers.

Let $P' = \{(x_i, y_i) : \forall p_i \in P\,,\, p = (x_i, y_i, z_i)\}$ be the set P projected in the xy-plane as illustrated in Figure 2.9. A triangulation of P' is the set of non-degenerate triangles formed connecting the points in P' with non-overlapping edges (Preparata and Shamos, 1985, p. 19). The union of the triangles forms the convex hull $\text{conv}(P')$ to P'. The set $\text{conv}(P')$ is the smallest set in $\mathbb{R}^2$ that contains $P' \subset \mathbb{R}^2$ and the line segment between every possible pair of points in $\text{conv}(P')$ (De Berg et al., 2008,

[3]A set of three or more points are *collinear* if they lie on the same straight line (Farin, 1997, p. 19).

p. 2). Furthermore, the triangulations that we consider are generated based on the planar coordinates to each point in P. The triangulation is therefore generated based on P' only. It is also possible to obtain a *data dependent* triangulation where the elevation associated to each point is also considered during the construction of a triangulation. Data dependent triangulations are described in e.g (Dyn et al., 1990), (Brown, 1991), (Hjelle and Dæhlen, 2006, pp. 95 - 112).

After finding a triangulation $\mathcal{T}(P')$ of P', each vertex in the triangulation can then be "lifted up" such that the elevation to each vertex at (x_i, y_i) corresponds to the elevation z_i as shown in Figure 2.11. This is possible since there is only one value z_i at each position (x_i, y_i).

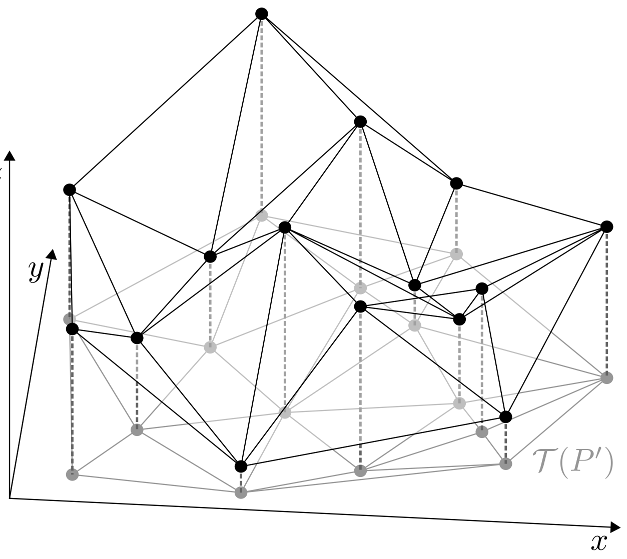


Figure 2.11: A lifted triangulation. The gray points belong to P'. Each point in P' are a projection of the point set P. In P, each point is associated with a height z_i. Each vertex in the triangulation $\mathcal{T}(P')$ can be lifted to its corresponding height in P.

When we discuss triangulations of a point set $P \subset \mathbb{R}^3$ in this thesis, we actually refer to the triangulation of the projected point set $P' \subset \mathbb{R}^2$ of P. We have decided to let a triangulation of P be understood as the planar triangulation of P since we do not consider triangulations in other dimensions. We say that a surface is *triangulated* when the planar triangulation of a point set is lifted up.

One can then find the height to each point from conv(P') in the lifted triangulation. For applications in GIS, using a linear interpolant over every triangle is usually sufficient, as it does not introduce oscillating effects in the interpolation (Li et al., 2004, p. 74). It is possible to obtain curved surfaces over triangulations by using for instance Bernstein polynomials, see e.g (Lai and Schumaker, 2007, pp. 20 - 85), (Akenine-Möller et al., 2018, pp. 740 - 743), but at the cost of more computations. Suppose p_1, p_2, and p_3 are the vertices of a triangle $T \in \mathcal{T}(P')$ and associated with the elevations z_1, z_2, and z_3, respectively. If we perform a linear interpolation over the triangle, the interpolated height $z_T(x, y)$ at every position (x, y) inside T can be found by (Hjelle and Dæhlen, 2006, p. 4):

$$z_T(x, y) = b_1(x, y)z_1 + b_2(x, y)z_2 + b_3(x, y)z_3, \tag{2.7}$$

where the values of $b_i : \mathbb{R}^2 \to \mathbb{R}$ for $i = 1, 2, 3$ are determined by how (x, y) is located relative to p_1, p_2, and p_3 and $\sum_{i=1}^{3} b_i(x, y) = 1$. The values $b_i(x, y)$ for $i = 1, 2, 3$ are called the *barycentric coordinates* of (x, y) relative to the triangle T. The computation of $b_i(x, y)$ for $i = 1, 2, 3$ is described in (Lai and Schumaker, 2007, pp. 18 - 19). Figure 2.12 shows how a point is mapped onto a triangle.

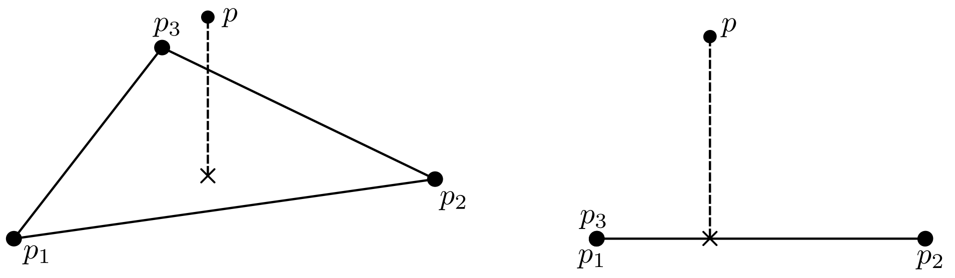


Figure 2.12: *Left:* The interpolated elevation, draw as the cross, to the point p with respect to the triangle formed by the vertices p_1, p_2, p_3. *Right:* The triangle and the point viewed from another angle. The figures are inspired by Figure 5 in (Soommart and Paitoonwattanakij, 1999, p. 70).

There always exists a triangulation of a point set (Cheng et al., 2012, pp. 31 - 32). There could also exist several triangulations for the same point set (De Berg et al., 2008, p. 192). These triangulations could be considered as equally appropriate for an application. Choosing the most appropriate or best triangulation is therefore not always straight forward. Generally, numerical errors in Equation (2.7) can occur if the triangles are long and thin (Shewchuk, 2002). There are some special cases long and thin triangles can be advantageous, see e.g (Rippa, 1992). In general, we would like to avoid long and thin triangles for our application. A type of triangulations that tend to avoid using such triangles in their construction, are *Delaunay triangulations* (Hjelle and Dæhlen, 2006, pp. 47 - 50). They are often encountered in surface modeling since there are well established theory about their properties and efficient algorithms for constructing the triangulations.

2.3.1 Delaunay triangulation

Delaunay triangulations are often encountered in GIS since they are commonly used to construct TINs. What separates a Delaunay triangulation from other triangulations, is that a Delaunay triangulation constructs triangles that are as close to being equiangular as possible. A Delaunay triangulation satisfies the *circle criterion*. The circle criterion states that the circumscribed circle to every triangle T in a triangulation cannot include any vertices from the triangulation in its interior (Hjelle and Dæhlen, 2006, p. 57). Finding the triangulation of a point set P that satisfies the circle criterion is equivalent to find the triangulation that maximizes the smallest angle of a triangle over all possible triangulations of P. A proof of this property can be found in e.g (Lawson, 1977, Sec. 3 pp. 1 - 8).

A Delaunay triangulation $\mathcal{D}(P)$ of a point set P has the property that removing and inserting a point p requires only a local update of $\mathcal{D}(P)$ to satisfy the circle criterion. In (Lawson, 1977, Sec. 3.4) it is proved that an insertion of a point requires a local retriangulation for the triangulation to satisfy the circle criterion. Since a point insertion requires a local retriangulation, then a point deletion will also require a local retriangulation. This is because a point deletion can be regarded as the reverse operation to a point insertion. This is discussed further in (Midtbø, 1994). We discuss these properties further in Section 2.3.2.1 and Section 2.3.2.2.

The retriangulation after a point deletion is performed within the *cell* $\mathcal{C}(p)$ of a vertex p, see Figure 2.13. A cell $\mathcal{C}(p)$ is the set of triangles in a Delaunay triangulation that has p as vertex (Lai and Schumaker, 2007, p. 234).

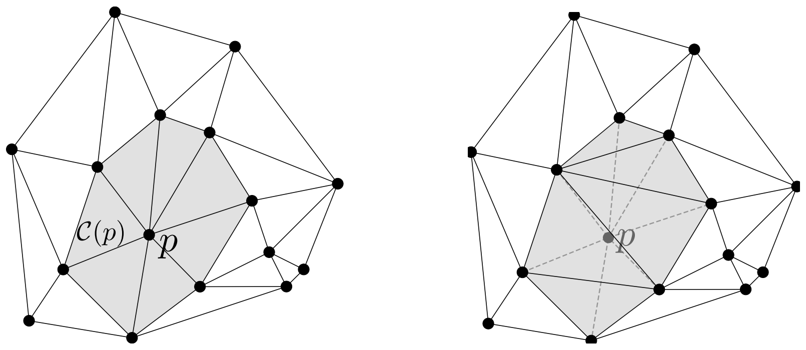


Figure 2.13: *Left:* The gray area is the cell $\mathcal{C}(p)$ to the point p. *Right:* After p is removed, only the triangles in $\mathcal{C}(p)$ are retriangulated.

A point set containing more than three points where the entire set is not collinear can be triangulated. A point set that can be triangulated, will have at least one Delaunay triangulation (Cheng et al., 2012, p. 39). The Delaunay triangulation is unique if no four or more points in a point set P lies on the same circle that does not contain any points from P in its interior.

As a consequence of the circle criterion, a Delaunay triangulation of a point set P has the property of being dual to the *Voronoi diagram* of P. They are dual in the sense that the Delaunay triangulation can be derived from the Voronoi diagram and vice versa. This duality is discussed in (Okabe et al., 2000, pp. 52 - 54).

2.3.2 Voronoi Diagrams

The Voronoi diagram of a point set $P \subset \mathbb{R}^2$ can give information about which points in P are spatially closest to each other. The *Voronoi polygon*[4] $\mathrm{V}(p_i)$ of a point $p_i \in P$ is the set of points from $\mathbb{R}^2$ that are closer to p_i than any other point from P (Okabe et al., 2000, p. 45),

$$\mathrm{V}(p_i) = \{x : d(x, p_i) \leq d(x, p_j) \; \forall p_j \in P, \; j \neq i\},$$

[4]Also called *Dirichlet regions*, *Thiessen polygons* or *Wigner-Setiz cells* (Preparata and Shamos, 1985, p. 204).

where $d(p, q) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ is the Euclidean distance[5] between the points $p = (x_1, y_1)$ and $q = (x_2, y_2)$.

The Voronoi polygon $\mathrm{V}(p_i)$ of a point $p_i \in P$ can be regarded as the intersection of all *half-planes* between p_i and the other points p_j from P. The half-plane $H(p_i, p_j)$ is the set of points that are closer to p_i than p_j in Euclidean distance, including the perpendicular bisector[6] of the line segment that has p_i and p_j as endpoints (Cheng et al., 2012, p. 155). The half-plane can be expressed as (Preparata and Shamos, 1985, p. 204):

$$H(p_i, p_j) = \left\{x \in \mathbb{R}^2 : d(x, p_i) \leq d(x, p_j)\right\}.$$

Since $\mathrm{V}(p_i)$ consists of all points that are closer to p_i than any other point from P, we get that

$$\mathrm{V}(p_i) = \bigcap_{\substack{p_j \in P \\ j \neq i}} H(p_i, p_j).$$

Figure 2.14 shows a half-plane and how the intersection of all of the other half-planes forms $\mathrm{V}(p_i)$.

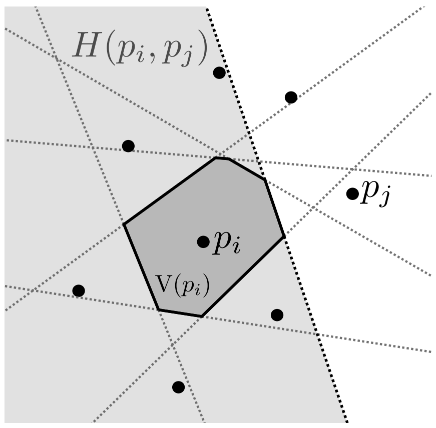


Figure 2.14: The half-plane $H(p_i, p_j)$, illustrated as the gray region. The smaller, gray dotted lines shows the bisector between the half-planes to the other points illustrated as black circles. The Voronoi polygon $\mathrm{V}(p_i)$ is obtained by taking the intersections of the half-planes between p_i and all of the other points. Note that we have defined $\mathrm{V}(p_i)$ to include its boundary, shown as the thick lines.

Two points p_i and p_j are *Voronoi neighbors* if the intersection between their Voronoi polygons is non-empty, i.e $\mathrm{V}(p_i) \cap \mathrm{V}(p_j) \neq \emptyset$ (Figure 2.15). Every point x in the non-empty set $\mathrm{V}(p_i) \cap \mathrm{V}(p_j)$ has the property $d(x, p_i) = d(x, p_j)$. They form a *Voronoi edge*.

[5]More generally, the Voronoi diagram could also be defined using another metric $d(u, v)$ than the Euclidean, see (Fortune, 2017, pp. 712 - 713).

[6]The *bisector* of a line l is the line that intersects l at its midpoint. (Cheng et al., 2012, p. 155)

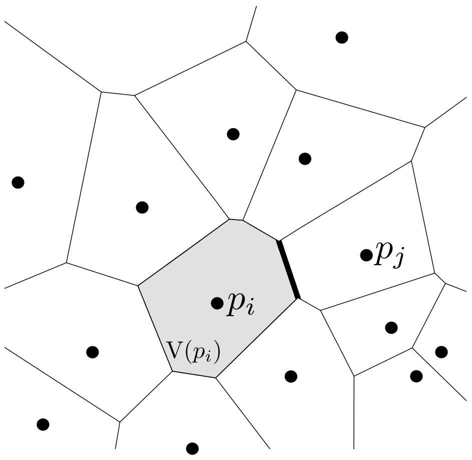


Figure 2.15: The Voronoi polygon $V(p_i)$ of the point p_i is shown as the gray area. The points p_i and p_j are Voronoi neighbours since they share a Voronoi edge. The Voronoi edge is illustrated as the thick line and consists of all the points that located equally far apart from p_i and p_j.

A *Voronoi vertex* is a point y where the intersection between three or more Voronoi polygons is non-empty. This means that $d(y, p_i) = d(y, p_j) = d(y, p_k)$ for at least one triplet of distinct points p_i, p_j, $p_k \in P$. A Voronoi vertex is thus the loci to the center of the circumscribed circle to the triangles formed by the points p_i, p_j, and p_k, see Figure 2.16. As described in Section 2.3.1, the circle does not contain any other point from P.

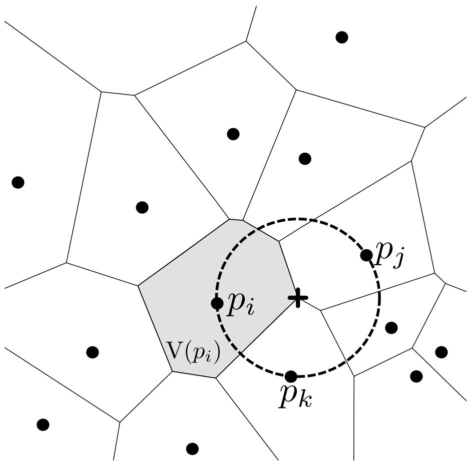


Figure 2.16: The plus is located at a Voronoi vertex that is the center to the dotted circle. The circle circumscribes the triangle in a Delaunay triangulation that has p_i, p_j, and p_k as vertices.

We get the *Voronoi diagram* Vor(P) of a point set P by collecting the Voronoi polygon to each point in P (Okabe et al., 2000, p. 45),

$$\text{Vor}(P) = \{\text{V}(p) : p \in P\}.$$

The Delaunay triangulation of a point set P can be derived from $\mathrm{Vor}(P)$ if we connect all points in P that are Voronoi neighbors in $\mathrm{Vor}(P)$, as shown in Figure 2.17.

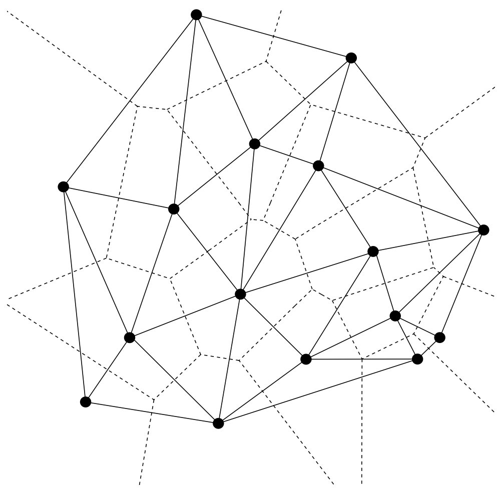


Figure 2.17: The Delaunay triangulation of a point set can be obtained by connecting the Voronoi neighbors to each point in the point set.

The duality makes it possible to consider the Delaunay triangulation and Voronoi diagram interchangeably. The duality also shows that the Delaunay triangulation connects the points that are closest to each other. Through a Delaunay triangulation of a point set, it is possible to extract and analyze points closest to each other.

2.3.2.1 Insertion of a point into the Voronoi diagram

As (Lawson, 1977) proved, an insertion into a Delaunay triangulation $\mathcal{D}(P)$ for a point set P requires only a local retriangulation. This is an important property of $\mathcal{D}(P)$ that we used in our implementations. We are only interested in the case where the inserted point belongs to the convex hull $\mathrm{conv}(P)$ of P. We use the duality between the Voronoi diagram and Delaunay triangulation to provide a short argument on why an insertion is local and how it is bounded.

Consider a point $p \notin P$ and $p \in \mathrm{conv}(P)$ that is inserted into $\mathrm{Vor}(P)$. The inserted point will be located in a Voronoi polygon $\mathrm{V}(q)$ to a point q where $q \neq p$. Let q_i be the N Voronoi neighbors to q and V_q the set containing q_i for $i = 1, \ldots, N$. Assume that p is contained in $M < N$ circles $\mathscr{C}_{i_j}$ that are centered at the Voronoi vertices shared by $\mathrm{V}(q)$, $\mathrm{V}\big(q_{i_j}\big)$, and at least one $\mathrm{V}(q_{i_k})$ where $q_{i_j}, q_{i_k} \in V_q$ and $j \neq k$ for $j = 1, \ldots, M$ as shown in Figure 2.18.

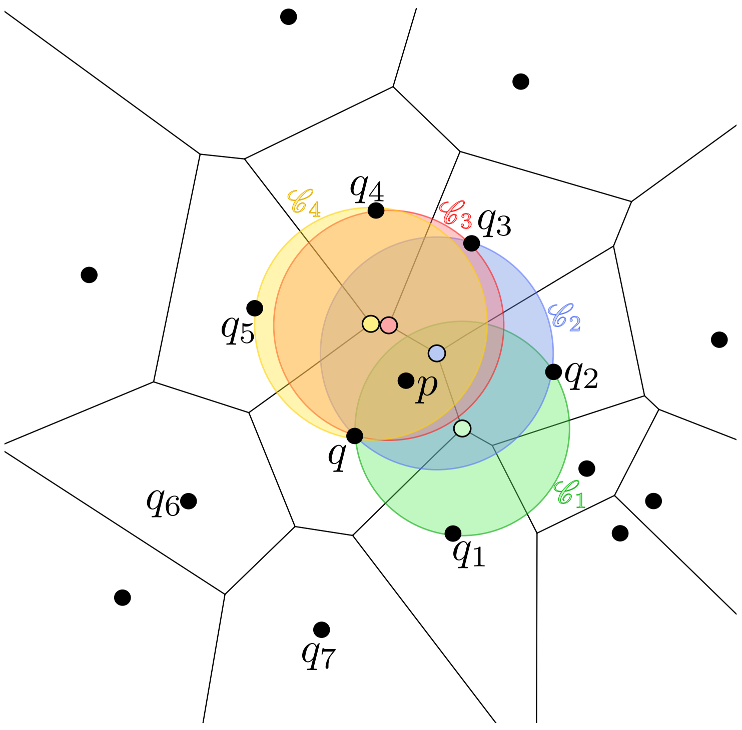


Figure 2.18: The inserted point p will be contained in some circles centered at Voronoi vertices. Each colored circle is centered at the Voronoi vertex having similar color.

The inserted point p will have the property $d(p, q_{i_j}) \leq d(q, q_{i_j})$ for every j, otherwise it would not have been in $\mathscr{C}_j$. This means that $H(p, q_{i_j}) \cap \mathrm{V}\big(q_{i_j}\big) \neq \emptyset$ for every j, as illustrated in Figure 2.19.

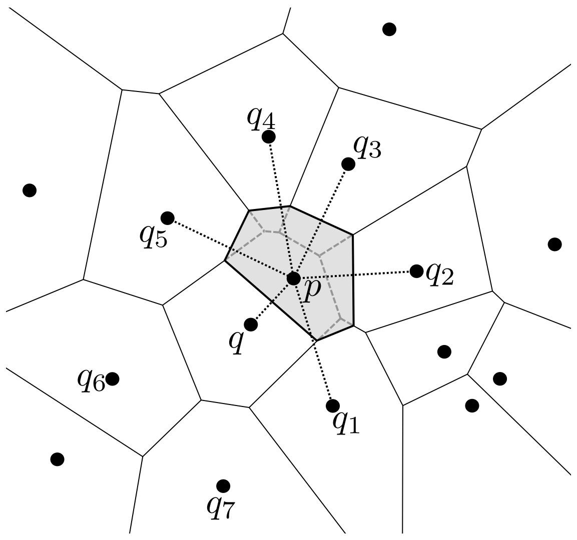


Figure 2.19: The gray area is the newly constructed Voronoi polygon to p. It is enclosed by the perpendicular bisectors at the dotted lines between p and every q_{i_j}.

The Voronoi polygon of p can thus be found by (Sibson, 1981, pp. 26 - 27):

$$\mathrm{V}(p) = \bigcup_{j=1,\ldots,M} H(p, q_{i_j}) \cap \mathrm{V}\big(q_{i_j}\big). \tag{2.8}$$

The Delaunay trianulation of $P \cup \{p\}$ is found by connecting p to q_{i_j} for $j = 1, \ldots, M$, as they now share a Voronoi edge.

It is impossible that p is located in some other circle $\mathscr{C}$ centered at a Voronoi vertex v not shared by $\mathrm{V}(p)$ and passing through the Voronoi neighbors that shares v. Since p is located in $\mathrm{V}(q)$, it is closest to q than any other point from P. If $p \in \mathscr{C}$, then p would be closer to some of the Voronoi neighbors $\mathscr{C}$ passes through, which is a contradiction.

Hence, an insertion of a point only requires a local retriangulation which is bounded by the triangles that has q_{i_j} as vertices.

2.3.2.2 Deletion of a point from the Voronoi diagram

A deletion can be seen as a reverse operation to insertion. In (Midtbø, 1994) it is discussed that a deletion requires a retriangulation of the cell to the deleted point. Based on the previous paragraph regarding the insertion of a point into a Voronoi Diagram, we would like to provide a short argument on why deletion of a point requires a local retriangulation of the dual Delaunay triangulation.

Suppose p is the point to be deleted from a point set P. Based on Figure 2.19, we can see that $\mathrm{V}(p)$ can be regarded as an union of smaller parts to the Voronoi polygons of the Voronoi neighbors to p. Let V_p be the set of Voronoi neighbor to p. A removal of a point p would make the Voronoi polygons to "grow in", as $\mathrm{V}(p)$ contains regions of points closer to all points in V_p than any other point in $P \setminus \{p\}$. Hence, only the Voronoi polygons to the points in V_p are changed. The Voronoi edges to the changed Voronoi polygons becomes different, but only shared between the points in V_p since they are closest to each other than any other point in P. Since the changed Voronoi edges are shared by the points in V_p, only a retriangulation of the triangles formed by the points in V_p in $\mathcal{D}(P)$ is necessary to obtain $\mathcal{D}(P \setminus \{p\})$.

2.4 Error assessment

In order to evaluate the algorithms, we have chosen a set of error assessments that describes how much the simplified surfaces based deviates from the input surfaces. Assume that P_T is a raster TIN of a given raster DEM P. The sets P_T and P contains the same amount of points. Each point p_i from P is assigned an elevation z_i. From the TIN to raster conversation, the point p_i gets associated with an approximated elevation $z_{T_{p_i}}$ from P_T. How much each approximated elevation $z_{T_{p_i}}$ deviates from the original elevation z_i, can be quantified differently based on which error assessment one decides to use.

We have decided to consider two error assessments in our studies when the given data are not associated with a set of weights. The first assessment, is the *root mean* *square error* (RMSE) that is commonly found in GIS literature (Reuter et al., 2009, pp. 94 - 95). The RMSE is often encountered as it describes how much the model and the input data deviates in general, has the same units as the input data, and is not too sensitive to noise in the data. The RMSE $e_{\text{RMSE}}(P, P_T)$ between P and P_T is

$$e_{\text{RMSE}}(P, P_T) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(z_i - z_{T_{p_i}}\right)^2}, \tag{2.9}$$

where N is the number of points in the sets P and P_T and z_i is the sampled elevation assigned to a point in P and $z_{T_{p_i}}$ the corresponding approximated elevation assigned to a point in P_T. The RMSE can be intuitively thought as a measurement on how the approximated elevation in P_T deviates from the elevations in P in general. We consider also the *max deviation* $e_{\text{MAX}}(P, P_T)$ between P and P_T that is given by

$$e_{\text{MAX}}(P, P_T) = \max_{i=1,\ldots,N} \left|z_i - z_{T_{p_i}}\right|.$$

We have chosen to only consider one error assessment, however, when the input data are associated with a set of weights. Each residual $(z_i - z_{T_{p_i}})^2$ is multiplied by the corresponding weight w_i from Equation (2.6) that represents the quality to the measurement z_i. We obtain a weighted RMSE (wRMSE) that is inspired by the weighted least squares (Strutz, 2011, pp. 26 - 27):

$$e_{\text{wRMSE}}(P, P_T) = \sqrt{\frac{\sum_{i=1}^{N} w_i \cdot \left(z_i - z_{T_{p_i}}\right)^2}{\sum_{j=1}^{N} w_j}}. \tag{2.10}$$

The weights are divided by their sum to let the wRMSE be close to the RMSE if the weights are approximately equal. As for the max deviation, we have decided to omit it when evaluating how the triangulated surfaces deviates from the input surfaces. We think that if there is some elevations associated with high amount of noise, the weighting should be able to dampen to contribution to it in Equation (2.10).

3

Surface simplification

The bathymetric maps obtained from an InSAS contains large amount of data. This introduces computational overhead in applications where for instance efficient data storage is important. In this chapter, we present two classes of simplification algorithms that later is applied on bathymetric maps obtained from an InSAS. The algorithms are defined to take into account both the spatial relationship between the points described by a Delaunay triangulation and the weights obtained from the associated coherence to each data point. Furthermore, a feature selection method is presented that is used together with one of the described algorithms.

3.1 Methods based on Delaunay triangulations

The points that are spatially closer to each other and associated with similar elevations might be redundant to keep. These points can represent almost the same information about the surface they were sampled from. This idea of points being close to each other and are more likely to be related in terms of elevation follows Tobler's first law of geography (Tobler, 1970). A Delaunay triangulation $\mathcal{D}(P)$ connects points that are spatially closest to each other in a point set P. From $\mathcal{D}(P)$, we can extract and compare the associated elevations to the nearby points to every point in P.

There exists different classes of surface simplification algorithms, see (Wilson, 2018, pp. 26 - 27), (Heckbert and Garland, 1997, p. 6 - 16). Two of the classes that we consider are point *decimation* and point *refinement*. Decimation algorithms iteratively removes points from their input point set (Heckbert and Garland, 1997, p. 14). Refinement algorithms begins with a subset of the input point set and iteratively adds points to the subset (De Floriani and Magillo, 2002, p. 382), which is the opposite of how decimation algorithms performs the simplifications. Both of the algorithms usually base their point selection on the computed significances (Li et al., 2004, p. 77). The iterative algorithms are *greedy* since they only consider what is best to do one iteration at a time. We describe decimation and refinement in Section 3.1.1 and Section 3.1.2, respectively. As a design criterion, we chose to only use the points from the input point set. Introducing a new point based on the input data could be challenging to model its weighting of and could require more computations. Point selection algorithms are therefore more appropriate for our application.

3.1.1 Decimation

In *decimation*, or *thinning*, points are iteratively removed from the input point set. The removal of points is based on a significance measure to each point. The significance measurement is supposed to describe how much the point contributes to represent important features of the point set, which in our case is elevation to each point. A decimation algorithm selects the point that has the least significance for every iteration.

Several methods exists for decimation based on Delaunay triangulations such as (Lee, 1991, pp. 272 - 274), (Demaret et al., 2005), (Li et al., 2014). What separates the algorithms generally, is how the significance to each point is computed. We have chosen to follow two of the presented methods for computing the significances from (Demaret et al., 2005). They also presented the general formulation of a decimation algorithm, the "*Thinning*" algorithm, which we based most of our implementations on. We chose to follow their article as we thought they provided a comprehensive description on how decimation can be done in general. Furthermore, we found their proposed significances easy to adapt for our application where we are interested to incorporate the weights from Equation (2.6) in our simplification algorithms.

The Delaunay triangulation $\mathcal{D}(P)$ of an input point set P is computed first. The point set P is assumes to be a raster DEM. The algorithm computes then the significance $\xi(p; \mathcal{D}(P))$ to each point $p \in P$ based on $\mathcal{D}(P)$, except for the corner points to P. The corner points are always included to ensure that the simplified point set has the same convex hull as P. The function $\xi(p; \mathcal{D}(P))$ aims to capture how important p is to describe the sampled surface in P. The computed significance $\xi(p; \mathcal{D}(P))$ should be low if p contains redundant data compared to the other points in P, and high if p contains important data compared to the other points in P. The significance should also take into account that each data point is associated with a weight w that described the quality of its assigned elevation. The point p' that has the least significance $\xi(p'; \mathcal{D}(P))$ of all points from P, is removed from the simplified point set. Points are iteratively removed until some criteria is met. Our implementation allows the decimation algorithm to terminate if either a given number of deletions are performed or until the triangulated point set reaches a given error threshold for a specified error assessment. Algorithm 1 summarizes the described procedure for a weighted decimation algorithm based on Delaunay triangulations.

Algorithm 1 Decimation algorithm with a given number of k deletions.

INPUT: The point set $P \subset \mathbb{R}^3$ to simplify.

A set of weights W that describes the quality of the sampled elevation to each point in P.

A given number of k deletions.

OUTPUT: A point set that contains less points than P.

```
1: procedure DECIMATION($P$, $W$, $k$)
2:     Let $P^{(0)} = P$.
3:     Compute and store $\xi\big(p; \mathcal{D}\big(P^{(0)}\big), W\big)$ for all $p \in P$.
4:     for $i = 0, \ldots, k-1$ do
5:         Find $p^{(i)}$ that has the least significance, i.e $p^{(i)} = \min_{p \in P^{(i)}} \xi\big(p; \mathcal{D}\big(P^{(i)}\big), W\big)$.
6:         Let $P^{(i+1)} = P^{(i)} \setminus \big\{p^{(i)}\big\}$.
7:         Update the stored significances with $\xi\big(p; \mathcal{D}\big(P^{(i+1)}\big), W\big)$
   for all points $p \in P^{(i+1)}$ that were affected by the removal in $\mathcal{D}\big(P^{(i)}\big)$.
8:     end for
9:     return The simplified data set $P^{(k)}$.
10: end procedure
```

The Algorithm 1 shows how the decimation is performed for given number of k deletions. The general idea is similar when an error threshold for a given error assessment is given. The for loop in Algorithm 1, line 4, is replaced by a while loop. For every iteration i in the while loop, the error between the triangulated simplified point set $P^{(i+1)}$ and the given input point set P is computed. The error is computed by a user-specified error assessment, which in our case can be one of the assessments as described in Section 2.4. The while loop terminates after m iterations where model error of $P^{(m+1)}$ is greater than a given error threshold and returns $P^{(m)}$.

The two proposed computations of significance from (Demaret et al., 2005) are called "*Adaptive thinning 1*" (AT1) and "*Adaptive thinning 3*" (AT3). The former method was the slowest among their three proposed significances, but gave the least max deviation between $\mathcal{D}\big(P^{(k)}\big)$ and P when applied to a terrain. The latter was shown to be the fastest among their proposed significances. It returned simplified point sets $P^{(k)}$ where the max deviation between $\mathcal{D}\big(P^{(k)}\big)$ and P was between the max deviation of the two other proposed significances when applied to a terrain. The results referred to here can be found in (Demaret et al., 2005, p. 330).

3.1.1.1 Adaptive Thinning 1: Significance using retriangulated cells

This computation of significance AT1 was found to be the slowest, yet gave the least max deviation in the triangulated simplified point sets among the proposed significances in (Demaret et al., 2005). This significance seems promising to use if the model error should be low and computational time is not crucial.

As described in (Demaret et al., 2005, p. 325), AT1 considers the cell $\mathcal{C}(p)$ to every point p it is supposed to find the significance of. The idea is to see how a possible removal of a point p affects the triangulation and the interpolated elevations to the points close to p. Let $p \in P$ be a point that AT1 is supposed to find its significance of. Assume that Algorithm 1 is at iteration j where the significances $\xi\big(p; \mathcal{D}\big(P^{(j+1)}\big)\big)$ are computed for a selection of points p from $P^{(j+1)}$. Consider the set V_p that contains all of the Delaunay neighbors to p in $\mathcal{D}\big(P^{(j+1)}\big)$. If p is removed from $\mathcal{D}\big(P^{(j+1)}\big)$, only the triangles in $\mathcal{C}(p)$ needs to be changed such that the triangulation of $P^{(j+1)} \setminus \{p\}$ is a Delaunay triangulation (Section 2.3.1). Hence, it is enough to find how the triangles are formed in $\mathcal{D}(V_p)$, see Figure 3.1. The calculation of the M interpolated elevations to every point $c_i \in \mathcal{C}(p) \cap P^{(j+1)}$ for $i = 1, \ldots, M$ is therefore based on $\mathcal{D}(V_p)$ only.

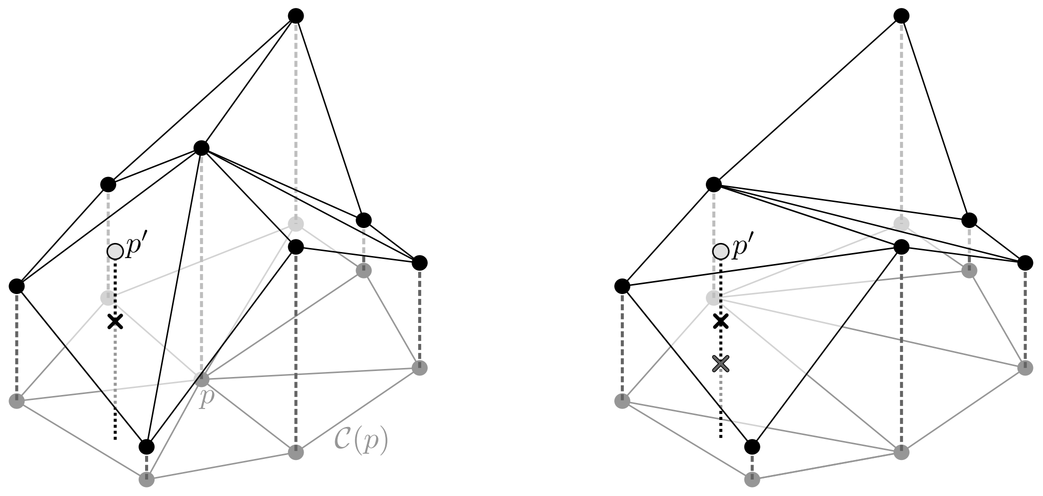


Figure 3.1: How a point removal affects a Delaunay triangulation. *Left:* The point p' is interpolated to one of the triangles in $\mathcal{C}(p)$. Its interpolated elevation becomes the elevation at the cross. *Right:* The retriangulation of $\mathcal{C}(p)$ after p is removed. The old interpolated value of p' has to be updated by the new interpolated value at the gray cross.

Since $c_i \in \mathcal{C}(p) \cap P^{(j+1)}$, we have that $c_i \in P^{(j+1)}$. The associated sampled elevation z_i from the true surface at point $c_i = (x_i, y_i, z_i)$ is therefore known and associated with a weight w_i from W. The AT1 computes the *interpolation error* $e_i = |z_{T_i}(x_i, y_i) - z_i|$ to c_i, where $z_{T_i}(x_i, y_i)$ is the interpolated height to c_i in $\mathcal{D}(V_p)$, which is found by using Equation (2.7), for $i = 1, \ldots, M$. We have adapted AT1 such that it weights each interpolation error e_i with the associated weight w_i to c_i. Our weighted AT1 (wAT1) find the significance $\xi_{\text{wAT1}}\big(p; \mathcal{D}\big(P^{(j+1)}\big), W\big)$ to p by

$$\xi_{\text{wAT1}}\big(p; \mathcal{D}\big(P^{(j+1)}\big), W\big) = \max_{i=1,\ldots,M} e_i \cdot w_i.$$

If all the weights are equal, we obtain AT1 from (Demaret et al., 2005). Otherwise, we refer to wAT1 when the weights are different for each point in P.

3.1.1.2 Adaptive thinning 3: Significance using directional triangles

The proposed significance that we have chosen to consider from (Demaret et al., 2005), is the AT3 that uses *directional triangles* (Demaret et al., 2005, p. 326) that we define in the next paragraph. Finding directional triangles does not require any retriangulation of a Delaunay triangulation. This made AT3 faster than the proposed significances in (Demaret et al., 2005). The algorithm to find a directional triangle is highly inspired by a discussion with Prof. Michael S. Floater, who co-authored in (Demaret et al., 2005).

Assume that Algorithm 1 is at an iteration j where the significances to a selection of points from $P^{(j+1)}$ has to be computed. Let $p \in P^{(j+1)}$ be the point we want to compute its significance of. Let $V_p = \{p_i\}_{i=1}^N$ contain all of the N Delaunay neighbors to p in $\mathcal{D}\big(P^{(j+1)}\big)$. A directional triangle T_{p_i} is found with respect to a vertex $p_i \in V_p$. If one draws a line through p_i and p, the line will intersect with either an edge at the boundary $\partial\mathcal{C}(p)$ to the cell $\mathcal{C}(p)$ or another vertex from V_p. If the line intersects with an edge at $\partial\mathcal{C}(p)$, there will be two distinct points p_{i_m} and $p_{i_{m-1}}$ from $V_p \setminus \{p_i\}$ that are the endpoints to the edge at $\partial\mathcal{C}(p)$, see Figure 3.2.

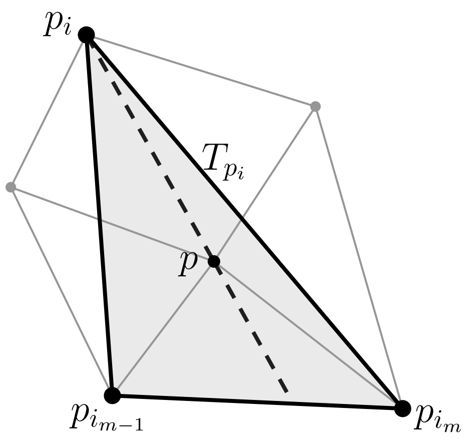


Figure 3.2: The dashed line from p_i through p intersects with an edge at the boundary $\partial\mathcal{C}(p)$ of $\mathcal{C}(p)$. There are two points p_{i_m} and $p_{i_{m-1}}$ connected by that edge. The directional triangle T_{p_i}, enclosed by the black solid lines in the illustration, is formed by using p_i, $p_{i_{m-1}}$, and p_{i_m} as vertices. The gray lines represents the Delaunay triangulation of p and its neighbors V_p are illustrated as the small gray dots.

If the line intersects at a point $p_{i_m} \in V_p$, then the directional triangle T_{p_i} is formed by connecting p_i, p_{i_m}, and $p_{i_{m-1}}$. The point $p_{i_{m-1}}$ is next to p_{i_m} in clockwise direction along $\partial\mathcal{C}(p)$.

Let $i_m \in \{k : k \neq i,\ k = 1, \ldots, N\}$ be the $N-1$ indices such that the sequence $p_i, p_{i_1}, p_{i_2}, \ldots, p_{i_{N-1}}$ is a traversal through all of the points along $\partial\mathcal{C}(p)$ in counter-clockwise direction, starting from p_i. The point p_{i_1} lies next to p_{i_2} along $\partial\mathcal{C}(p)$ in counterclockwise direction, the point p_{i_2} lies next to p_{i_3} along $\partial\mathcal{C}(p)$ in counterclockwise direction, and so on. We let the indices be such that $p_{i_0} = p_{i_{N-1}}$. To find a directional triangle T_{p_i}, a possible way is to consider the angles $\theta_{i_m} \in [0, 2\pi)$ between the edge e_i connecting p and p_i and the edge e_{i_m} connecting p and $p_{i_{i_m}}$ for all i_m, $m = 1, \ldots, N-1$. We are only interested in finding k where θ_{i_k} is greater than or equal to π for the first time when traversing along $\partial\mathcal{C}(p)$ in counterclockwise direction. When the k is found, a line through p and p_i will cross an edge connecting p_{i_k} and $p_{i_{k-1}}$ at $\partial\mathcal{C}(p)$, see Figure 3.3.

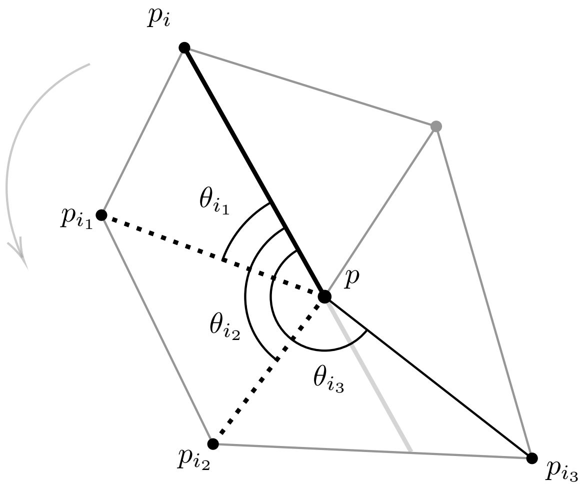


Figure 3.3: Using angles between the edges to find a directional triangle. In this illustration, $\theta_{i_3} > \pi$. Thus the vertices p_i, p_{i_3}, and p_{i_2} will form a directional triangle to p with respect to p_i.

A directional triangle T_{p_i} is then formed by connecting the vertices p_i, $p_{i_{m-1}}$ and p_{i_m}.

The angle θ_{i_m} between the edges $\bar{e}_i$ and $\bar{e}_{i_{i_m}}$ can be found by considering the *cross product* between them. The edges can be represented as vectors in $\mathbb{R}^3$ by setting their last component equal to zero. This is equal to setting their value along the z-axis equal to zero. Let $\boldsymbol{e}_i$ and $\boldsymbol{e}_{i_m}$ be the vectors in $\mathbb{R}^3$ representing the edges $\bar{e}_i$ and $\bar{e}_{i_m}$, respectively [1]. If we let the vertices $p = (x, y)$, $p_i = (x_i, y_i)$, and $p_m = (x_{i_m}, y_{i_m})$, the vectors $\boldsymbol{e}_i$ and $\boldsymbol{e}_{i_m}$ can be found by

$$\boldsymbol{e}_i = \begin{bmatrix} x_i - x \\ y_i - y \\ 0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{e}_{i_m} = \begin{bmatrix} x_{i_m} - x \\ y_{i_m} - y \\ 0 \end{bmatrix}. \tag{3.1}$$

The cross product $\boldsymbol{e}_i \times \boldsymbol{e}_{i_m}$ is related to the angle θ_{i_m} between $\boldsymbol{e}_i$ and $\boldsymbol{e}_{i_m}$ (Matthews, 1998, p. 9) by

$$\boldsymbol{e}_i \times \boldsymbol{e}_{i_m} = \|\boldsymbol{e}_i\|\|\boldsymbol{e}_{i_m}\| \sin\theta_{i_m} \boldsymbol{n}\,, \tag{3.2}$$

[1] We want to make a distinction between the edge $\bar{e}_i$ and the vector $\boldsymbol{e}_i$ in $\mathbb{R}^3$ to avoid ambiguity. An edge $\bar{e}_i$ can be represented quite differently than a vector. For instance, one could let $\bar{e}_i$ be represented by its endpoints p and p_i by letting $\bar{e}_i = (p, p_i)$ as done in e.g (Hjelle and Dæhlen, 2006, p. 24). However, a vector in $\mathbb{R}^3$ is always represented by three components from $\mathbb{R}$.

where $\|\cdot\|$ is the Euclidean norm[2] of its argument and $\boldsymbol{n} \in \mathbb{R}^3$ the normal vector to $\boldsymbol{e}_i$ and $\boldsymbol{e}_{i_m}$. Since the components of $\boldsymbol{e}_i$ and $\boldsymbol{e}_{i_m}$ are known from Equation (3.1), it is possible to compute $\boldsymbol{e}_i \times \boldsymbol{e}_{i_m}$ directly. We choose our coordinate system to be a right-handed Cartesian coordinate system and refer to (Kreyszig, 2011, pp. 368 - 370) for a description of the coordinate system and how it affects the definition of the cross product. The cross product between $\boldsymbol{e}_i$ and $\boldsymbol{e}_{i_m}$ is

$$\boldsymbol{e}_i \times \boldsymbol{e}_{i_m} = ((x_i - x)(y_{i_m} - y) - (x_{i_m} - x)(y_i - y))\hat{\boldsymbol{k}}, \tag{3.3}$$

with $\hat{\boldsymbol{k}}$ being the unit basis vector along the axis considered as the z-axis in this thesis, see Figure 2.9.

Equating Equation (3.2) and Equation (3.3), and letting $\hat{\boldsymbol{k}} = \boldsymbol{n}$, gives that

$$(x_i - x)(y_{i_m} - y) - (x_{i_m} - x)(y_i - y) = \|\boldsymbol{e}_i\|\|\boldsymbol{e}_{i_m}\| \sin\theta_{i_m}. \tag{3.4}$$

In Equation (3.4), $\sin\theta_{i_m}$ is the only quantity that determines the sign of the left side of the equality. The sign of $\sin\theta_{i_m}$ changes from positive to negative when $\theta_{i_m} \geq \pi$. From how $\boldsymbol{e}_i$ and $\boldsymbol{e}_{i_m}$ are defined in Equation (3.1), $\sin\theta_{i_m}$ begins by being positive. When traversing in counter clockwise direction, $\sin\theta_{i_m}$ is positive until some k where $\theta_k \geq \pi$. Thus, it is sufficient to find a k where

$$(x_i - x)(y_{i_k} - y) - (x_{i_k} - x)(y_i - y) \leq 0$$

for the first time when traversing through the sequence $p_{i_1}, p_{i_2}, \ldots, p_{i_{N-1}}$ along $\partial\mathcal{C}(p)$.

After having found T_{p_i}, the elevation $z_{T_{p_i}}(x, y)$ at point $p = (x, y)$ in T_{p_i} is found by using the barycentric coordinates as shown in Equation (2.7). Since $p \in P^{(j+1)}$ where $P^{(j+1)} \subset P$, the sampled elevation z along with its associated weight w is known. The interpolation error e_i of p at the directional triangle T_{p_i} is then found by $e_i = \left|z_{T_{p_i}}(x, y) - z(x, y)\right|$ for $i = 1, \ldots, N$. Each e_i is multiplied by $w_{T_{p_i},1} + w_{T_{p_i},2} + w_{T_{p_i},3}$, where $w_{T_{p_i},1}$, $w_{T_{p_i},2}$, and $w_{T_{p_i},3}$ are the weights from W associated to the vertices of T_{p_i} for $i = 1, \ldots, N$. The significance $\xi_{\text{wAT3}}(p; \mathcal{D}(P), W)$ to the point p from the weighted AT3 (wAT3) is obtained by

$$\xi_{\text{wAT3}}(p; \mathcal{D}(P), W) = \max_{i=1,\ldots,N} e_i \cdot \left(w_{T_{p_i},1} + w_{T_{p_i},2} + w_{T_{p_i},3}\right),$$

where we obtain AT3 from (Demaret et al., 2005) if all of the associated weights in W are equal.

3.1.2 Refinement

In contrast to decimation that iteratively removes points from the input point set, a refinement method begins with a subset of the input point set and iteratively inserts points into the subset. A refinement algorithm can be faster than a decimation algorithm if the simplified point set should contain far less points than the input

[2] If a vector $\boldsymbol{a} \in \mathbb{R}^n$ has components a_i for $i = 1, \ldots, n$, then the Euclidean norm is defined by $\|\boldsymbol{a}\| = \sqrt{\sum_{i=1}^{n} a_i^2}$ (Kreyszig, 2011, p. 313).

point set (Pfeifer and Mandlburger, 2018, p. 369). This is because refinement initializes with a coarse model of the input point set, which might be closer to the desired output than the whole input point set.

As for decimation, an iterative refinement algorithm follows a general algorithm and is defined accordingly to how the significancy to each point is computed. A refinement algorithm inserts points that has the largest significancy compared to the other considered points one iteration at a time. The significancy should describe how important the point is to include in a simplified point set in order to represent important features of the sampled surface.

The algorithm initializes by forming a set $P^{(0)}$ that contains the points that forms $\operatorname{conv}(P)$ from the input point set P, and possibly combines it with some other given points from P. The refinement algorithm extends $P^{(0)}$ by inserting one point for every iteration. The algorithm terminates when a number of k insertions into $P^{(0)}$ has been performed or if the triangulated extended point set satisfies an error threshold for a specified error assessment. Algorithm 2 summarizes the general definition of an iterative refinement algorithm for a given number of k point insertions.

Algorithm 2 Refinement algorithm for a given number of k point insertions.

INPUT: The point set $P \subset \mathbb{R}^3$ to simplify.
A set of weights W that describes the quality to each point in P.
A given number of k insertions.

OUTPUT: A point set containing less points than P.

```
1:  procedure REFINEMENT($P$, $W$, $k$)
2:      Let $P^{(0)} \subset P$ be the subset of points where $\operatorname{conv}\big(P^{(0)}\big) = \operatorname{conv}(P)$.
3:      Compute and store $\xi\big(p; \mathcal{D}\big(P^{(0)}\big), W\big)$ for all $p \in P \setminus P^{(0)}$.
4:      for $i = 0, \dots, k-1$ do
5:          Find $p^{(i)}$ that has the most significance, i.e $p^{(i)} = \max_{p \in P \setminus P^{(i)}} \xi\big(p; \mathcal{D}\big(P^{(i)}\big), W\big)$.
6:          Let $P^{(i+1)} = P^{(i)} \cup \big\{p^{(i)}\big\}$.
7:          Update the stored significances with $\xi\big(p; \mathcal{D}\big(P^{(i+1)}\big), W\big)$
    for all points $p \in P \setminus P^{(i+1)}$ that were affected by the insertion in $\mathcal{D}\big(P^{(i)}\big)$.
8:      end for
9:      return The reduced data set $P^{(k)}$.
10: end procedure
```

The Algorithm 2 shows how the refinement is performed for a given number of k

insertions. The general idea is similar when an error threshold for a given error assessment is given. The for loop in Algorithm 2, line 4, is replaced by a while loop. For every iteration i in the while loop, the error between the triangulated simplified point set $P^{(i+1)}$ and the given input point set P is computed. The error is computed by a user-specified error assessment, which in our case can be one of the assessments as described in Section 2.4. The while loop terminates after m iterations where model error of $P^{(m+1)}$ is greater than a given error threshold and returns $P^{(m)}$.

We consider only one definition of significance for refinement, even though several algorithms exists such as those described in (Heckbert and Garland, 1997, pp. 10 - 14) and (Soommart and Paitoonwattanakij, 1999). The definition follows closely the "*local error measure*" from (Garland and Heckbert, 1995), as they found this significance to be the fastest and returned the most accurate triangulated point sets among the other significances they tested for.

Let P be the input point set Algorithm 2 is supposed to simplify and contains N points. Assume that Algorithm 2 is at iteration j and must compute the significance $\xi_{\text{wRef}}\big(p; \mathcal{D}\big(P^{(i+1)}\big), W\big)$ to M points p_i from $P \setminus P^{(j+1)}$, where $0 \leq M \leq N$. Each point p_i is located in some triangle in $\mathcal{D}\big(P^{(j+1)}\big)$. Assume that p_i is located in the triangle T_{p_i} and its interpolated height within T_{p_i} is $z_{T_{p_i}}$. Since $p_i \in P \setminus P^{(j+1)}$, its associated elevation z_i sampled from some surface and its weight $w_i \in W$ computed from Equation (2.6) is also known for all $i = 1, \ldots, M$. The weighted significance $\xi_{\text{wRef}}\big(p; \mathcal{D}\big(P^{(i+1)}\big), W\big)$ is found by weighting the interpolation error $e_i = \big|z_{T_{p_i}} - z_i\big|$ for each p_i;

$$\xi_{\text{wRef}}\big(p; \mathcal{D}\big(P^{(i+1)}\big), W\big) = \max_{i=1,\ldots,M} e_i \cdot w_i.$$

The significance as proposed in (Garland and Heckbert, 1995) is obtained when the weights in W are equal.

3.2 Point extraction based on quadtree splitting

We consider how a point feature extraction can be used together with refinement. The refinement algorithm begins with a coarse model of the input point set. The triangles in the initial triangulation consists of few triangles that covers the convex hull to the input point set. The triangles can cover large areas and have many points located within them. There are therefore many points to compute the significances of and updated in the first iterations in the simplification process.

The input data are assumed to be in raster format. This format is convenient to use for analysis of the represented surface as we can for instance compute any feature within a neighborhood centered at each pixel. It is thus possible to obtain a feature image where each pixel describes how each point from the input raster data contributes to represent the selected feature. Thereafter, a resampling of the feature image can be performed to extract the points that describes most of the feature.

A surface can be characterized by abrupt changes in its topography (Zhou and Chen, 2011, p. 39). We consider the variance of the elevations within a local neighborhood to each pixel of a given raster DEM data. The variance of the DEM can describe the variation of the elevations to each point in the given DEM and is simple in its definition. We therefore think this could be useful for our application.

Suppose P is a raster DEM formed by a grid $\mathcal{I}$ of size $N \times M$. The value $z[i,j]$ where $(i,j) \in \mathcal{I}$ is the assigned elevation to a point p located at (x_i, y_j) in the plane. Suppose a rectangular neighborhood of size $(2n+1) \times (2m+1)$ is given for some chosen non-zero positive integers n and m. The neighborhood is centered at $(i,j) \in \mathcal{I}$ when it computes the estimated variation for every point associated with the indices $(i,j) \in \mathcal{I}$. Before the estimated variation $\sigma[i,j]$ of $p_{ij} = (x_i, y_j, z[i,j])$ is computed, a second order polynomial fit is applied to the points within the neighborhood. The second order polynomial fit is applied to reduce the presence of noise in the elevations assigned to the points within the neighborhood. Let $S = (2n+1)\cdot(2m+1)$ and $p_k = (x_{i_k}, y_{i_k}, z[i_k, j_k])$ the points that are located within the neighborhood centered at (i,j) for $k = 1, \ldots, S$. The second order bivariate polynomial fit is performed by finding the polynomial coefficients stored in a vector $\boldsymbol{\beta} = [\beta_1,\, \beta_2, \ldots, \beta_6]^T$ by the method of least squares (Hastie et al., 2009, pp. 44 - 45). This approach of fitting a second order bivariate polynomial to a surface is described in (Li et al., 2004, pp. 124 - 126), but note that Equation (6.27) in (Li et al., 2004, p. 126) should not include the rightmost $n \times 6$ matrix X. Our vector $\boldsymbol{\beta}$ can be found by

$$\boldsymbol{\beta} = \left(M^T M\right)^{-1} M^T \boldsymbol{z},$$

where

$$M = \begin{pmatrix} 1 & x_{i_1} & y_{j_1} & x_{i_1}^2 & x_{i_1}y_{j_1} & y_{j_1}^2 \\ 1 & x_{i_2} & y_{j_2} & x_{i_2}^2 & x_{i_2}y_{j_2} & y_{j_2}^2 \\ \vdots & & & & & \vdots \\ 1 & x_{i_S} & y_{j_S} & x_{i_S}^2 & x_{i_S}y_{j_S} & y_{j_S}^2 \end{pmatrix} \text{ and } \boldsymbol{z} = \begin{pmatrix} z[i_1, j_1] \\ z[i_2, j_2] \\ \vdots \\ z[i_S, j_S] \end{pmatrix}.$$

The raster data P is padded with its nearest neighbor when the polynomial fit is applied to the pixels along the borders of P. We chose to pad with the nearest neighbors as they do not introduce discontinuities in the polynomial fits along the borders.

The fitted values $z'[i_k, j_k]$ of $z[i_k, j_k]$ for all k can be found by

$$z'[i_k, j_k] = \beta_1 + \beta_2 x_{i_k} + \beta_3 y_{j_k} + \beta_4 x_{i_k}^2 + \beta_5 x_{i_k} y_{j_k} + \beta_6 y_{j_k}^2, \quad k = 1, \ldots, S.$$

The estimated $\sigma'[i,j]$ centered at $(i,j) \in \mathcal{I}$ in a neighborhood of size $(2n+1) \times (2m+1)$ is then computed by

$$\sigma'[i,j] = \frac{\sum_{k=1}^{S} \left(z'[i_k, j_k] - \mu_{ij}\right)^2}{S}, k = 1, \ldots, S,$$

where

$$\mu_{ij} = \frac{\sum_{k=1}^{S} z'[i_k, j_k]}{S}$$

is the average of z' located in the neighborhood centered at (i,j).

The final image σ is thereafter found by scaling $\sigma'[i, j]$ such that it has values between one and zero inclusive for all $(i, j) \in \mathcal{I}$ (Gonzalez and Woods, 2010, pp. 101 - 102),

$$\sigma[i, j] = \frac{\sigma'[i, j] - \sigma'_{\text{MIN}}}{\sigma'_{\text{MAX}} - \sigma'_{\text{MIN}}}$$

where $\sigma'_{\text{MAX}} = \max\limits_{(i,j)\in\mathcal{I}} \sigma'[i, j]$ and $\sigma'_{\text{MIN}} = \min\limits_{(i,j)\in\mathcal{I}} \sigma'[i, j]$.

The variance image σ is thereafter non-uniformly resampled. The resampling is supposed to take the points associated with high variance. At the same time, the resampling must not take too many points that are close to each other and have high variance. This is because refinement cannot reduce the number of points in the initial set of points. We want this feature selection to help refinement converge without making the resulting simplified point set contain more points than necessary. Therefore, we have decided to use an adaptive resampling scheme based on *quadtrees*.

A quadtree is a tree where each node is associated to a rectangle that partitions a rectangular domain recursively (Burrough et al., 2015, p. 63), see Figure 3.4. It is constructed by subdividing a domain into rectangles until a given threshold is met or the rectangles cannot be subdivided further. This data structure is commonly encountered in GIS since it serves as an effective way of storing data at various levels of details (Li et al., 2004, p. 198).

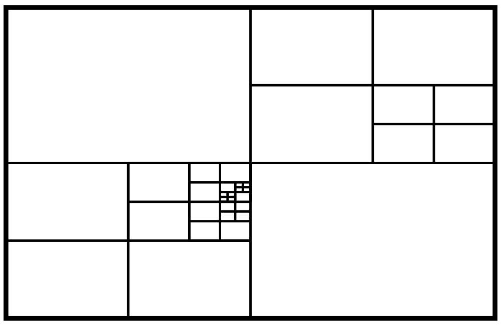


Figure 3.4: Illustration of how a rectangular domain, bounded by the thick lines, is subdivided into rectangles by quadtree splitting.

The algorithm does not store the tree obtained from the quadtree split. It is only the rectangles from the rectangular subdivision of the domain that are considered to extract points from the variance image. We decided to let the base cases of the splitting be determined by two thresholds t_c and t_v. The threshold t_c is the number of allowed number of points within each rectangle. The threshold t_v is the desired maximum variation σ within each rectangle. The algorithm terminates if either t_c is met or the maximum value of σ within a rectangle is not higher than t_v. We refer to the whole process of constructing a variation image and perform the quadtree based resampling of it as *variation sampling*. When needed, we explicitly state the parameters involved in a variation sampling.

4

Results

The simplification algorithms were applied on three types of different data sets that are synthetic data not associated with coherence, synthetic data associated with coherence, and a real data set acquired from an InSAS. In this chapter, we present the results from the algorithms applied on the different data sets.

4.1 Validity of the local updates of significances

Our implementations are not based on any pre-developed code, except for the functions that comes with MATLAB. We have interchangeably used MATLAB 2017a and MATLAB 2019a throughout the development. The computations of the significances were applied on a toy example that we could compute the significances of without the aid of any programs and compare with the results from the implementations. At the end, we ran our implementations though a test that we consider as the most important. In this test, we verified whether our implementation correctly performed the local updates of the significances after either a point insertion or a point deletion.

The test was performed by applying our implemented decimation algorithms and refinement algorithm in two different ways on an input point set $P \subset \mathbb{R}^3$. The tests for both decimation and refinement followed similar structure. Hence, we provide a general formulation of our tests. We let an Algorithm refer to either a decimation algorithm or the refinement algorithm. We let an operation either be a point deletion or point insertion. If the Algorithm refers to a decimation algorithm, an operation is a point deletion. If the Algorithm refers to the refinement algorithm, an operation is a point insertion. The tests were performed as follows:

1 The Algorithm was called within a for loop k times. The Algorithm performed one operation on the given point set for every call within the for loop. For each call, the Algorithm constructed a new Delaunay triangulation and calculated the significance to every point in the given point set. The given point set was either increased or reduced by one point for every iteration. After k iterations, the Algorithm returned a point set P_1.

2 The ALGORITHM was called only once where it had to perform k OPERATIONS, where k was the same number as applied in **1**. This call made the ALGORITHM try to update the significances only to the points that were affected by an OPERATION within a loop, thus exploiting the property of Delaunay triangulations where a local retriangulation is necessary after an OPERATION. The ALGORITHM returned a point set P_2.

The point sets P_1 and P_2 were thereafter compared. If all of the points in P_1 and P_2 were equal, the test was successful. We regarded two points $p_1 = (x_1, y_1, z_1)$ and $p_2 = (x_2, y_2, z_2)$ as equal if $x_1 = x_2$, $y_1 = y_2$, and $z_1 = z_2$. We found from our tests with different point sets P and number of k OPERATIONS that one function in MATLAB sometimes gave erroneous results that we had to circumvent. The function `pointLocation` from Mathworks sometimes found the wrong triangles to the points at the boundary $\partial\text{conv}(P)$ of $\text{conv}(P)$ they were supposed to be located in. We decided to move the points at $\partial\text{conv}(P)$ except for the corner points by 10^{-13} into $\text{conv}(P)$ and use the translated points only for finding which triangles the points were located in. After this circumvention, our tests were successful for our selection of different P and k.

We concluded that the local updates of the significances performed as expected based on our tests. This was important for us to verify, as the implementations using local updates performed significantly better than constructing a new Delaunay triangulation and recomputing the significances to all of the points of the given point set in terms of computational time.

4.2 Measured efficiency of the implementations

Refinement can be faster than decimation if the simplified point set should contain far less points than the original point set (Section 3.1.2). We measured the wall-clock time for each of the implemented algorithms to see if our implementations followed the expected trends in time. We use the term *wall-clock time* to refer to the execution time of a algorithm throughout this thesis. The measured wall-clock times served as an indicator to see if we have implemented the algorithms such that excessive computations were avoided.

We chose an input point set of 40000 points. The point set was simplified to a selection of number of points. For each amount of points, we repeated the simplification five times. The wall-clock time was measured for every repeated simplification and averaged. The measured wall-clock times are shown in Figure 4.1.

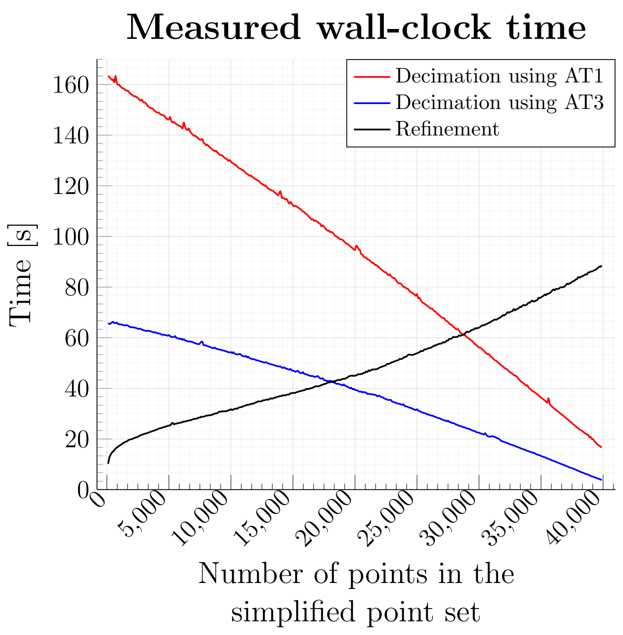


Figure 4.1: The measured wall-clock times for to the simplification algorithms. The measurements were performed on a computer having an Intel i7-8700 processor (3.2 GHz base frequency).

The AT3 was faster than AT1. The results in (Demaret et al., 2005, p. 330) showed that AT3 is supposed to be faster than AT1. The refinement algorithm was faster than the decimation algorithms until a certain number of points in the simplified point sets was reached. The refinement algorithm was faster than AT1 as long as the simplified point set contained approximately less than 75 percent of the points from the input point set. The refinement algorithm was faster than AT3 as long as the simplified point set was less than approximately 40 percent of the points from the input point set.

4.3 Simplification of surfaces without coherence

The three simplification algorithms were applied on sets of synthetic surfaces that were generated by two different functions. The tests where the wall-clock time was measured, was performed on a computer having an Intel i7-8700 processor with 3.2 GHz base frequency and six cores.

4.3.1 The generated surfaces

Two bivariate functions that describes different surfaces were chosen. The generated surfaces have different characteristics compared to each other. This makes it possible to examine how well the algorithms handles different cases of seabeds.

Franke's test function

The first test function was the Franke's test function as presented in (Franke, 1979, p. 13). It is defined as

$$\begin{aligned} z[i,j] = 0.75 \exp\left(-\frac{(9x_i-2)^2}{4} - \frac{(9y_j-2)^2}{4}\right) + \\ 0.75 \exp\left(-\frac{(9x_i+1)^2}{49} - \frac{9y_j+1}{10}\right) + \\ 0.5 \exp\left(-\frac{(9x_i-7)^2}{4} - \frac{(9y_j-3)^2}{4}\right) - \\ 0.2 \exp\left(-(9x_i-4)^2 - (9y_j-7)^2\right), \end{aligned} \tag{4.1}$$

where the coordinate samples x_i and y_j were both uniformly spaced in $[0,1]$ and formed a grid $\mathcal{I}$ where $(i,j) \in \mathcal{I}$. Together with $z[i,j]$, they form a raster DEM $\forall\,(i,j) \in \mathcal{I}$.

The surface is smooth and has two peaks and one pit which are slowly varying (Figure 4.2). This test function was chosen to test how well the algorithms handles a smooth and slowly varying surface.

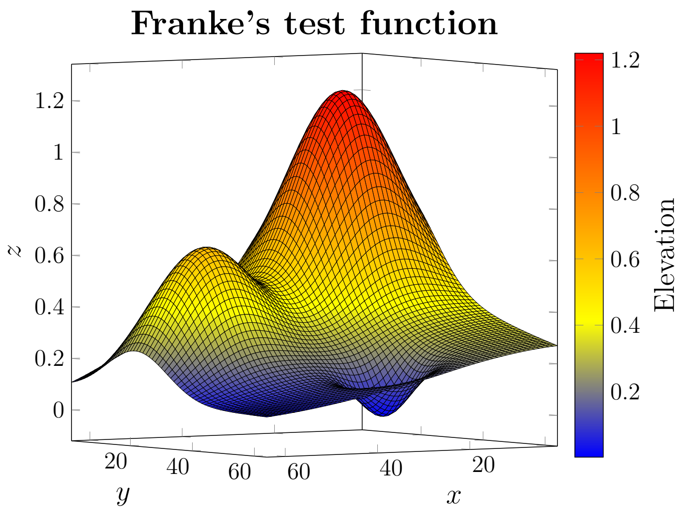


Figure 4.2: Franke's test function as presented in (Franke, 1979, p. 13) and the values for x and y mapped to integer coordinates. The number of points is 64×64 in this plot.

It is also of interest to the evaluate how the algorithms can handle edges and rapid variations. Therefore, we defined our own test function where the surface have regions with abruptly changing topography.

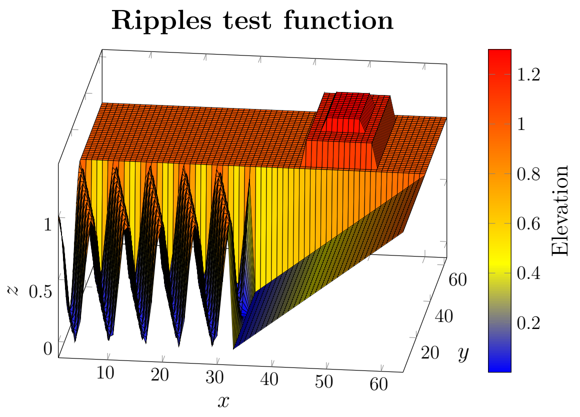
Ripples test function

The Ripples test function is inspired by the data set acquired from an InSAS that is considered in Section 4.5. The function represents an extreme case of a seabed surface. The Ripples test function is defined as

$$z[i,j] = \begin{cases} 0.5\cos\left(2\pi(5x_i + y_j)\right) + 0.5, & 0 \le i < \frac{1}{2}N \text{ and } 0 \le j < \frac{1}{2}M \\ \dfrac{2}{N}\left(i - \dfrac{1}{2}N\right), & \frac{1}{2}N \le i < N \text{ and } 0 \le j < \frac{1}{2}M \\ 1.2, & \frac{11}{20}M \le i < \frac{19}{20}M \text{ and } \frac{13}{20}N \le j < \frac{17}{20}N \\ 1.3, & \frac{13}{20}M \le i < \frac{17}{20}M \text{ and } \frac{7}{10}N \le j < \frac{4}{5}N \\ 1, & \text{otherwise} \end{cases} . \tag{4.2}$$

The samples x_i and y_j formed a grid $\mathcal{I}$ where $(i,j) \in \mathcal{I}$ and is uniformly sampled from the interval $[-1,1]$. As for Franke's test function, $z[i,j]$ form a raster DEM together with x_i and y_j $\forall\,(i,j) \in \mathcal{I}$.

The test function is divided into four regions with different characteristics (Figure 4.3). We have named the regions as R_1, R_2, R_3, and R_4. Each region has its own dominating feature. The region R_1 is flat and has a constant elevation. The region R_2 has two rectangles placed on top of each other. The region R_3 is rapidly varying, described by the discrete sampled cosine in Equation (4.2). The region R_4 is flat, but gradually increases its elevation from zero at the left hand side of the region to one at the right hand side of the region. Region R_1 and R_2 is separated from R_3 and R_4 by a cliff that has a depression angle of 90°.



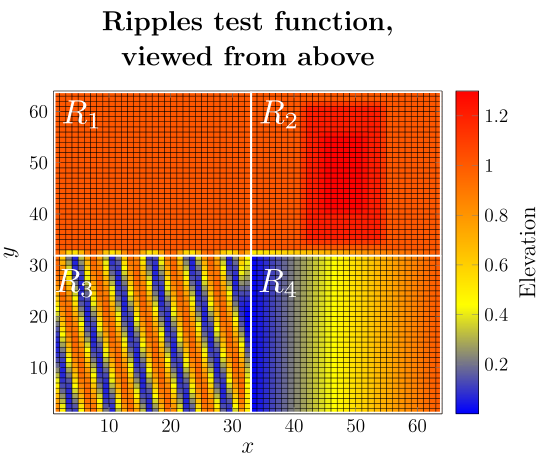


Figure 4.3: The ripples test function. In this plot, the Ripples test function is generated by 64 × 64 points. *Left:* The Ripples test function viewed from an angle. *Right:* The regions R_1, R_2, R_3, and R_4 the Ripples test function consist of.

The regions need different densities of points to be represented by. The region R_3 is the most rapidly varying region compared to the other regions. The edges to the rectangles in R_2 and the cliff between the regions R_1 and R_2 and the regions R_3 and R_4 introduces discontinuity to the surface. It is of interest to examine how the algorithms manages to represent the various regions and how the simplification algorithms distributes their points for each region.

4.3.2 Franke's test function

The algorithms were applied on a point set generated by Franke's test function. All of the studies that are described in this section were performed on a point set that contained 40000 points, except for the studies described in Section 4.3.2.3.

4.3.2.1 Number of points after reaching an error threshold

It is of interest to examine how many points the simplified point sets contains after reaching a given error threshold. We say that a point set satisfies an error threshold if it is the smallest point set from an algorithm that have an error equal to or smaller than the threshold. We have considered the RMSE and max deviation as error assessments (Section 2.4). We chose a set of error thresholds that was used for both assessments. The thresholds were 10^{-k} for $k = 1, \ldots, 7$. All of the algorithms returned point sets with less than ten percent of the input point set for the error thresholds of 10^{-2} and 10^{-1}, see Table 4.1, Table 4.2, and Table 4.3. The point sets from AT1 and refinement contained less than ten percent of points from the input point set and satisfied an error threshold of 10^{-3} for both RMSE and max deviation. The AT3 returned a point set that contained less than ten percent of points from the input point set for RMSE threshold of 10^{-3}, but not for the max deviation. The AT3 returned point sets that had the largest amount of points compared to AT1 and refinement.

Table 4.1: The number of points in the simplified sets obtained from AT1 and the percentage of points from the input point set.

Error threshold	Number of points in simplified set, RMSE	Percentage in size of original point set, RMSE	Number of points in simplified set, max deviation	Percentage in size of original point set, max deviation
$1{\times}10^{-7}$	39465	98.662	39966	99.915
$1{\times}10^{-6}$	36588	91.47	39580	98.95
$1{\times}10^{-5}$	26861	67.152	34400	86.0
$1{\times}10^{-4}$	8271	20.678	16763	41.908
$1{\times}10^{-3}$	842	2.105	2256	5.64
$1{\times}10^{-2}$	83	0.208	233	0.582
$1{\times}10^{-1}$	9	0.022	24	0.06

Table 4.2: The number of points in the simplified sets obtained from AT3 and the percentage of points from the input point set.

Error threshold	Number of points in simplified set, RMSE	Percentage in size of original point set, RMSE	Number of points in simplified set, max deviation	Percentage in size of original point set, max deviation
$1{\times}10^{-7}$	39865	99.662	39998	99.995
$1{\times}10^{-6}$	38359	95.898	39978	99.945
$1{\times}10^{-5}$	33383	83.458	38389	95.973
$1{\times}10^{-4}$	13153	32.882	31436	78.59
$1{\times}10^{-3}$	1374	3.435	8422	21.055
$1{\times}10^{-2}$	150	0.375	497	1.242
$1{\times}10^{-1}$	17	0.042	62	0.155

Table 4.3: The number of points in the simplified sets obtained from refinement and and the percentage of points from the input point set.

Error threshold	Number of points in simplified set, RMSE	Percentage in size of original point set, RMSE	Number of points in simplified set, max deviation	Percentage in size of original point set, max deviation
1×10^{-7}	39480	98.7	39986	99.965
1×10^{-6}	36795	91.988	39666	99.165
1×10^{-5}	28034	70.085	34849	87.122
1×10^{-4}	8778	21.945	17675	44.188
1×10^{-3}	862	2.155	2209	5.522
1×10^{-2}	88	0.22	263	0.658
1×10^{-1}	10	0.025	29	0.072

For an easier comparison between RMSE and max deviation, we plotted the number of points against the selected error thresholds for RMSE and max deviation in Figure 4.4 and Figure 4.5, respectively.

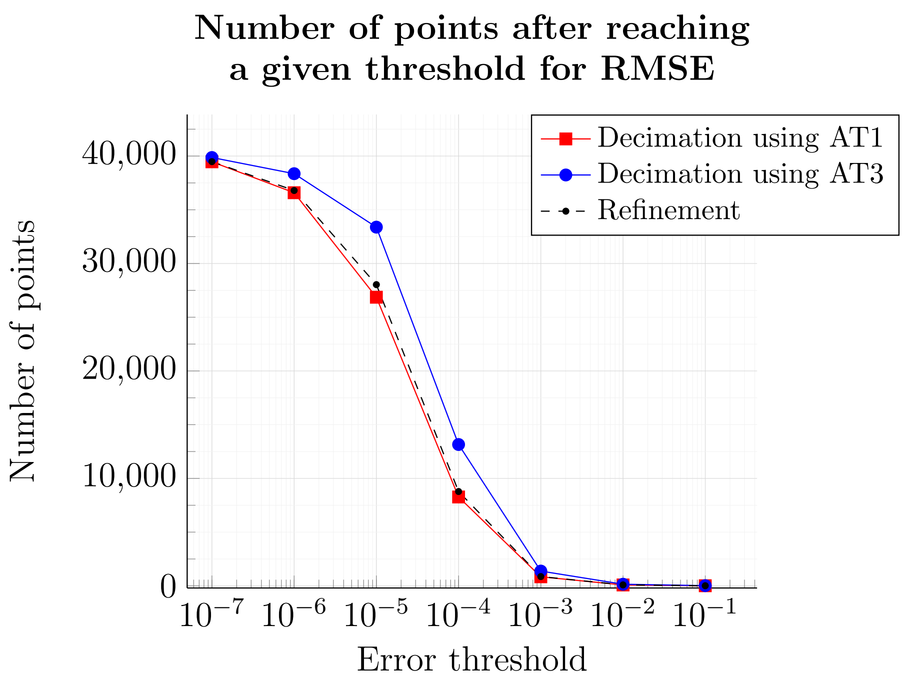


Figure 4.4: The number of points in the simplified point sets that satisfied the RMSE thresholds.

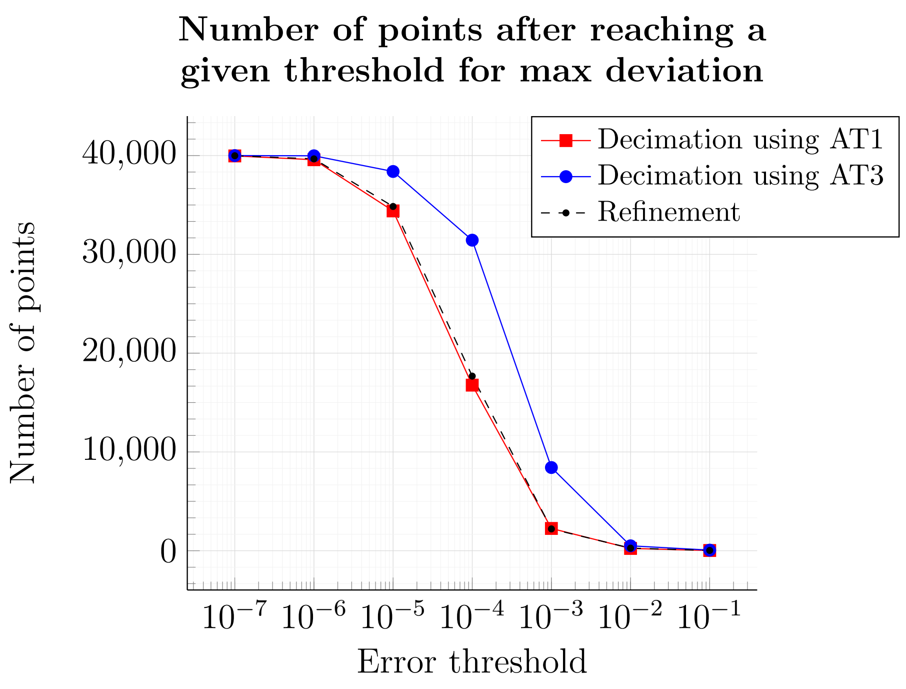


Figure 4.5: The number of points in the simplified point sets that satisfied the max deviation thresholds.

The RMSE and max deviation made the simplified point sets follow similar trend in terms of amount of points (Figure 4.4 and Figure 4.5). We chose therefore to consider only one of the error assessments in the subsequent analyses of the algorithms applied on the data set generated by Franke's test function, namely the RMSE.

It is of interest to examine how the distribution of points differed in the point sets that satisfied the same RMSE thresholds from the algorithms. The distribution of points in a point set determines how the triangles are formed in the simplified surface. This affects how the approximated surfaces becomes compared to the input surface. The RMSE thresholds 10^{-3} and 10^{-2} were the largest thresholds we tested for that made the algorithms return point sets with less than ten percent of point from the input point set. The point sets that satisfied these thresholds differed by a factor of ten in percentage of the amount of points from the input point set (Table 4.1, Table 4.2, and Table 4.3). The triangulations that satisfied RMSE threshold of 10^{-3} are shown in Figure 4.6 and the triangulations that satisfied RMSE threshold of 10^{-2} are shown in Figure 4.7.

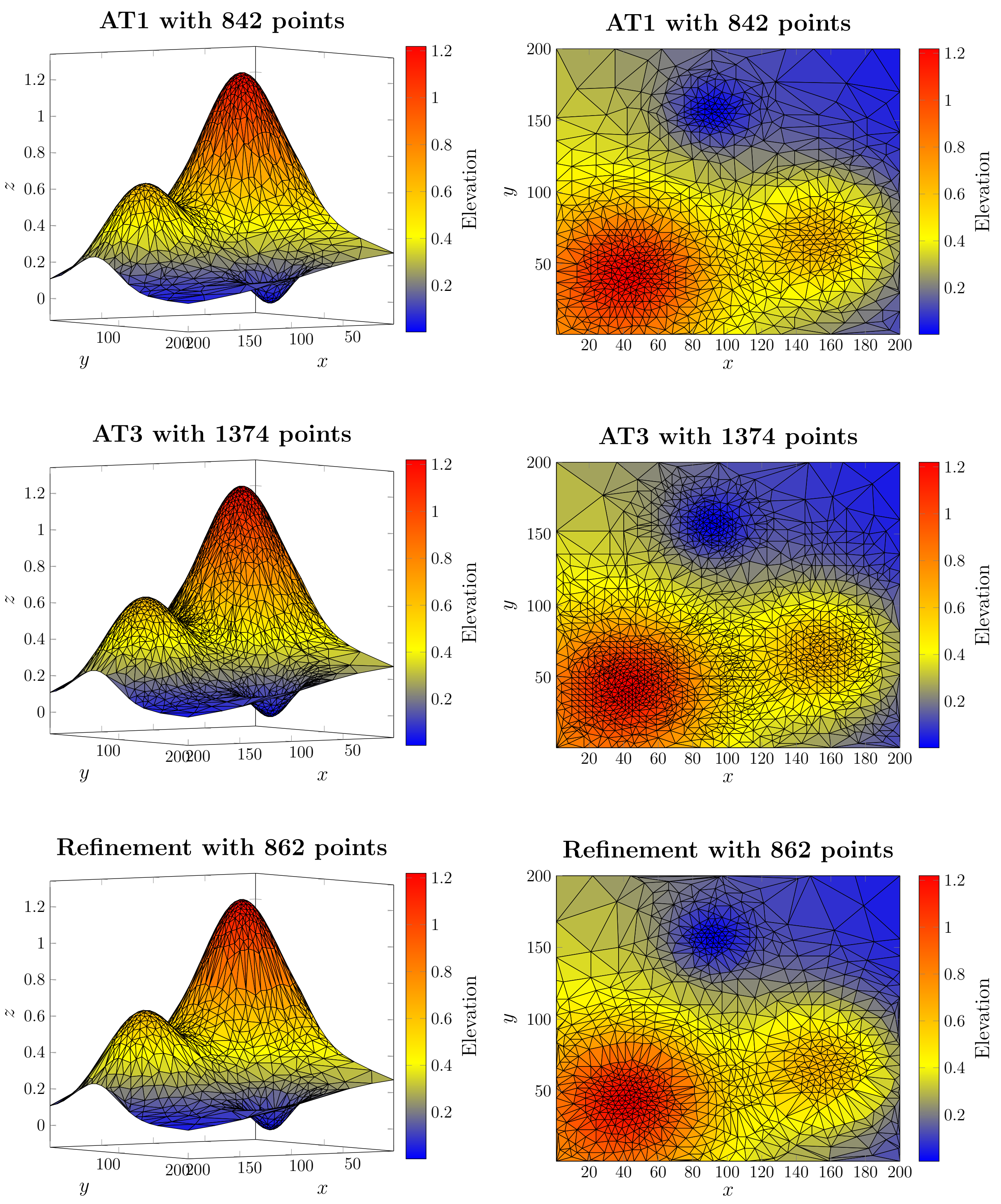


Figure 4.6: The triangulated point sets that satisfied RMSE threshold of 10^{-3}.

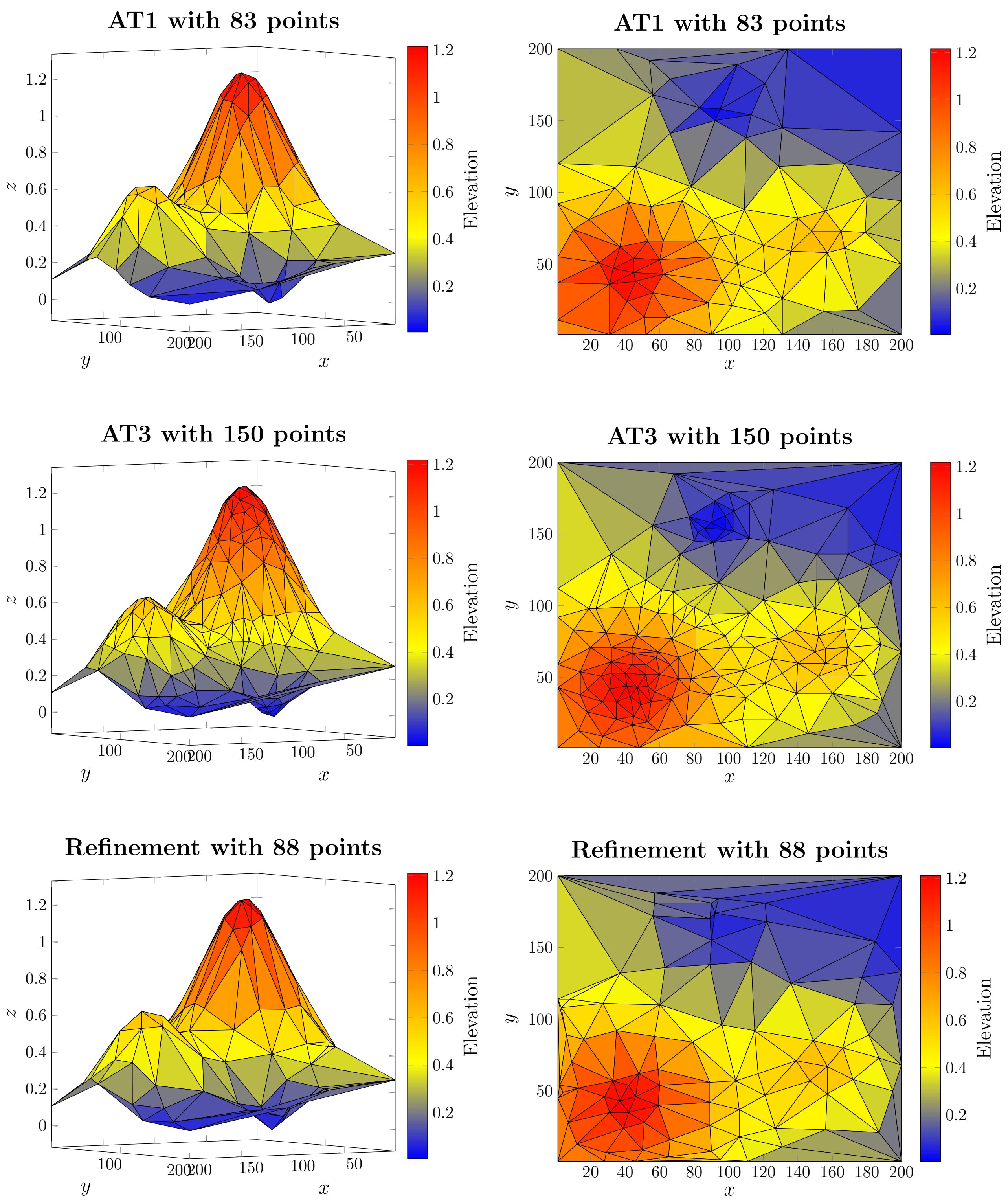


Figure 4.7: The triangulated point sets that satisfied RMSE threshold of 10^{-2}.

There is no clear distinctions between the triangulations. All of the algorithms managed to preserve the peaks and the pit in their triangulated point sets. The only difference we observe is that AT3 used more triangles, but this was expected from Table 4.2. The AT3 returned point sets that contained more points compared to AT1 and the refinement algorithm (Figure 4.4). It is therefore of interest to quantify better the triangulated structures to examine whether the triangulations differed.

4.3.2.2 Distribution of the longest edges and smallest angles to the triangles in a triangulation

We chose to look at the longest edge and smallest triangle to each triangle in a triangulation. Long and thin triangles are generally not preferred in a triangulation (Section 2.3). We computed the angles as described in (Press et al., 2007, pp. 1120 - 1121) to avoid numerical inaccuracies. The edge lengths were computed by using the Euclidean norm between the vertices to the triangles. The longest edge length and smallest angle to each triangle in a triangulation is presented in a scatter plot. We normalized the longest edge lengths by the maximum possible edge length. The maximum edge length is the diagonal to the input raster data. The raster data are assumed to be defined over a rectangle or a square since this is the case for the bathymetries from the InSAS we consider (Section 2.2). The input point set we considered was formed by a 200×200 large grid. Since the input grid had 200 points along each axis, the diagonal was $200 \cdot \sqrt{2}$ long from Pythagoras' theorem. The longest edge lengths to each triangle was therefore normalized by $200 \cdot \sqrt{2}$. We regard a long and thin triangle as a triangle where its smallest angle α is such that $\alpha \leq 5^\circ$ and its longest normalized edge length l is such that $l \geq 0.4$.

The simplified point sets contained equal amount of points such that the triangulated structures could be compared. The number of triangles can vary even though the amount of points are equal. The triangulated point sets will, however, share the same lower and upper bounds for the number of triangles, see Lemma 1.2 in (Hjelle and Dæhlen, 2006, p. 9). We made the algorithms simplify the given point set that contained 40000 points down to 2000 points and 200 points. The triangulations and their corresponding scatter plots of the simplified point sets with 2000 and 200 points are shown in Figure 4.8 and Figure 4.9, respectively. The colors to the markers in the scatterplots are used only to show better the distribution of the markers. The red boxes in the scatter plots shows the area of when a triangle is considered as long and thin.

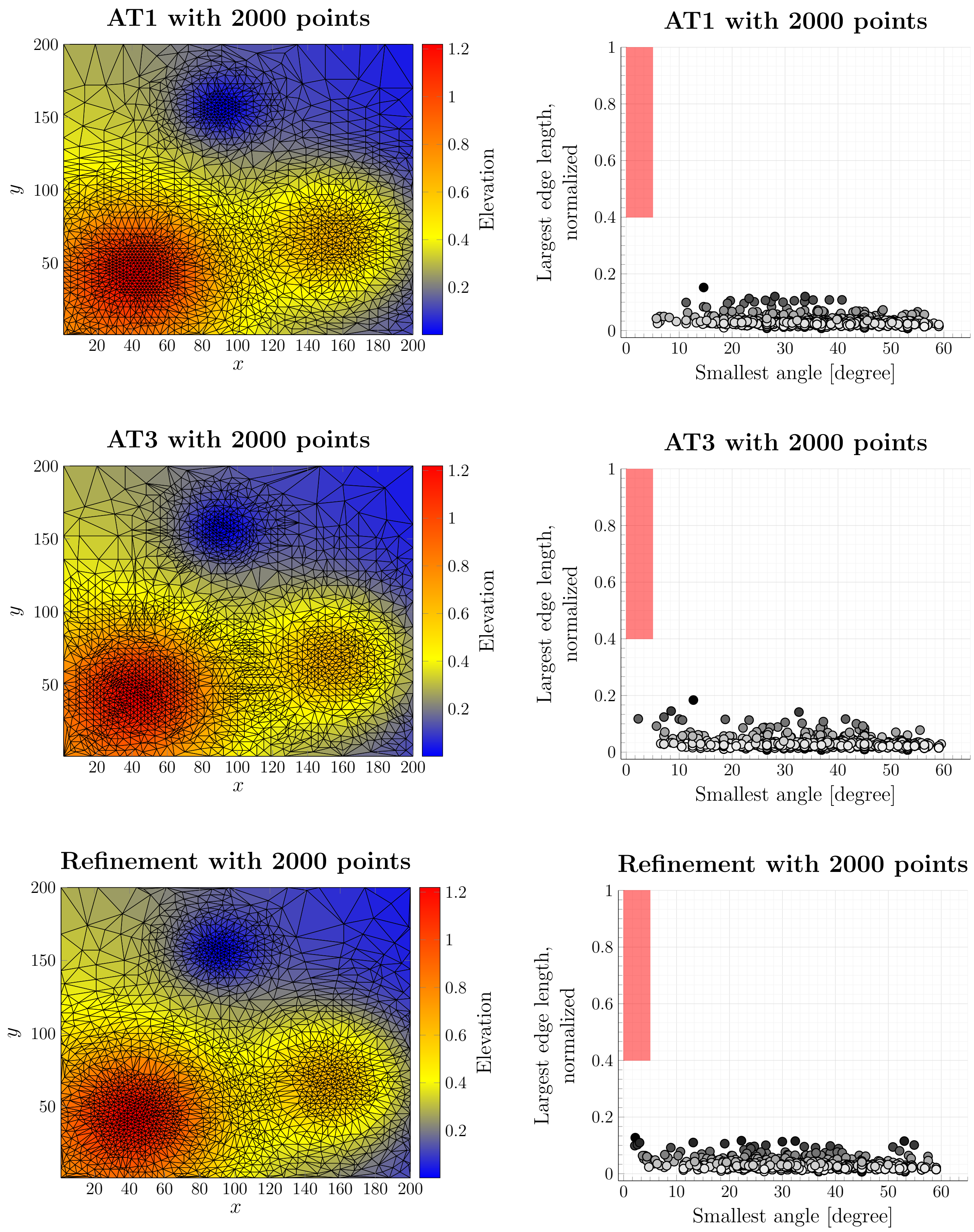


Figure 4.8: Triangulated point sets that contained 2000 points. *Left column:* The triangulated point sets viewed from above. *Right column:* The scatter plots of each triangulation.

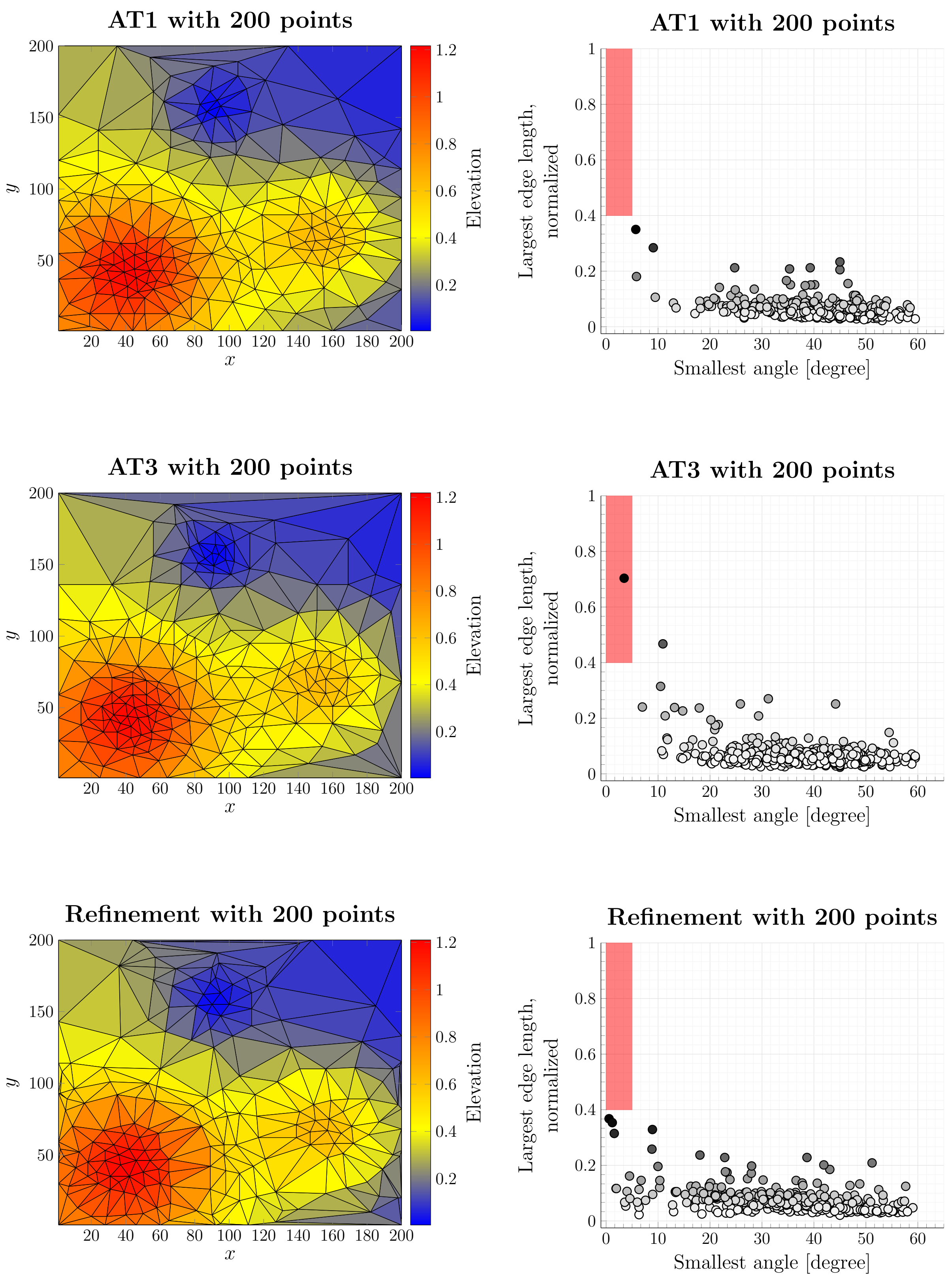


Figure 4.9: Triangulated point sets that contained 200 points. *Left column:* The triangulated point sets viewed from above. *Right column:* The scatter plots of each triangulation.

The AT3 was the only algorithm that returned a point set that had one long and triangle present in its triangulation (Figure 4.8 and Figure 4.9). We also see from the scatter plots, that the refinement algorithm returned point sets that formed triangles where many of them had $\alpha \leq 10°$. The AT1 and refinement returned point sets that formed triangulations where the triangles were spread for $10° \leq \alpha \leq 60°$.

We counted the number n_T of long and thin triangles in the triangulations formed by N_T triangles to be sure how many long and thin triangles were present in the triangulations, see Table 4.4. The table confirms our observation that only AT3 returned a point set that contained only one long and thin triangle when triangulated.

Table 4.4: Table over long and thin triangles, which are triangles with $\alpha \leq 5°$ and $l \geq 0.4$.

Algorithm	$\frac{n_T}{N_T}$, 200 points	Percentage of long and thin triangles, 200 points	$\frac{n_T}{N_T}$, 2000 points	Percentage of long and thin triangles, 2000 points
AT1	$\frac{0}{369}$	0	$\frac{0}{3912}$	0
AT3	$\frac{1}{381}$	0.26	$\frac{0}{3938}$	0
Refinement	$\frac{0}{375}$	0	$\frac{0}{3922}$	0

4.3.2.3 Variation sampling to construct an initial subset of points for refinement

The refinement algorithm returned point sets that contained almost the same number of points as the point sets from AT1 for a selection of RMSE thresholds (Figure 4.4). The refinement algorithm converged faster than AT1 as long as the simplified point contained less than 75 percent of the points from a point set that contained 40000 points. A question of interest is to see if it is possible to make the refinement algorithm converge even faster. If nothing is given to the algorithm, it initializes by using the four corner points of the input point set. If the point set is large, the distance between the corners along at least one axis will also be large. This will make the initial triangulation cover a large area and thereby many points to calculate and update the significances to. If the refinement algorithm is given a number of initial points that contains more than only the corner points, the initial triangulation will consist of more triangles where each triangle covers smaller areas. This could make the refinement algorithm converge faster since there might be fewer points to calculate and update the significances to in the beginning of the refinement process.

As a reference, we measured the wall-clock time to the refinement algorithm without passing an initial set of points to it. We constructed point sets of size $n \times n$ for $n = 250, 500, 750, 1000$. The refinement algorithm returned point sets that satisfied RMSE threshold of 10^{-3}. The refinement algorithm was called three times for each value of n. The wall-clock time was measured for each repeated call and thereafter averaged. Our results are shown in Table 4.5.

Table 4.5: The measured wall-clock time in seconds of how long it took the refinement algorithm to reach a RMSE error threshold of 10^{-3} without an initial subset of points.

$n \times n$	Number of points	Average wall-clock time [s]
250×250	1067	28.38
500×500	2185	142.47
750×750	3180	354.72
1000×1000	4242	677.1

Thereafter, the refinement algorithm was given ten different initial sets of 1000 randomly and uniformly drawn points from the input point set. The input point set contained $n \times n$ points where n was the same values as in Table 4.5. The refinement algorithm inserted points into the initial point set until a RMSE threshold of 10^{-3} was met. Ten different sets of initial points were passed to the refinement algorithm. This was done to examine whether the distribution of the initial points affected how long it took the algorithm to converge and how many points it returned. We computed the standard deviation to examine how each unique set of initial points affected the number of insertions and the wall-clock time for the algorithm to converge. For each n, the simplification was repeated three times. The wall-clock time was measured for every repeated simplification and thereafter averaged. The results from our experiments are presented in Table 4.6.

Table 4.6: The measured wall-clock time in seconds of how much it took the refinement algorithm to reach a RMSE error threshold of 10^{-3}.

$n \times n$	Average number of points	Standard deviation, number of points	Average wall-clock time [s]	Standard deviation, wall-clock time [s]
250×250	1452.6	14.99	4.57	0.33
500×500	2346.1	22.62	36.98	2.66
750×750	3333.8	13.81	122.89	5.75
1000×1000	4347.7	27.41	285.03	31.55

The measured wall-clock times were less for refinement with randomly sampled initial points than for refinement without an initial set of points (Table 4.5 and Table 4.6). However, the simplified point sets obtained from refinement with randomly sampled initial points contained larger amount of points compared to refinement without a set of initial points. The least difference in amount of points between the simplified points sets occurred when the input point set contained 1000×1000 points. The simplified point sets from refinement with initial subsets of points had approximately 2.47 percent more points than the simplified point set from refinement without an initial set of points.

We examined how variation sampling (Section 3.2) performed compared to random sampling of the initial points. It is of interest to examine whether variation sampling can make the refinement algorithm converge faster and return point sets with fewer points than refinement with randomly and uniformly sampled initial points. We chose the point set with 1000×1000 points, as it was our largest point set that we tested for in Table 4.6.

The variation images requires a given neighborhood size to estimate the variation to each pixel. We computed a set of variation images based on a selection of neighborhood sizes. The images are presented in Figure 4.10.

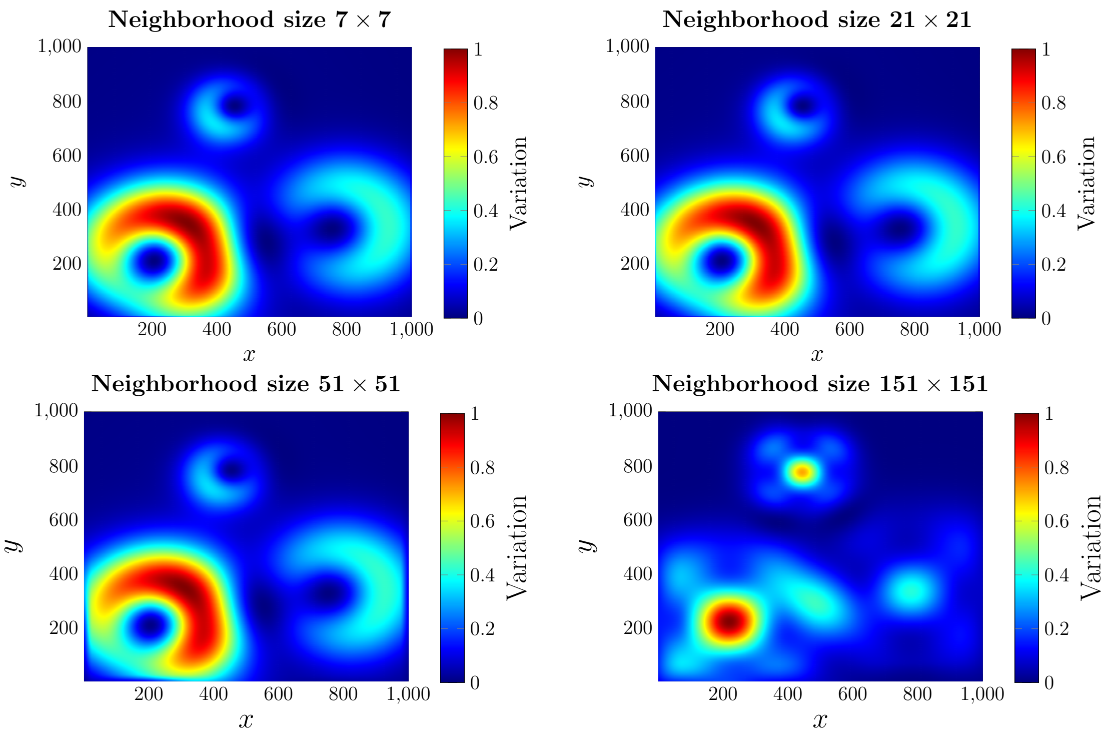


Figure 4.10: The computed variation images.

The variation image that was computed with neighborhood size of 151×151 differs from the other images (Figure 4.10). It is not clear which neighborhood size captured the best the variation to the input point set. The variation image computed with a 151×151 neighborhood seems to capture the peaks and pit of the surface. However, the other images seems to capture the areas that have a greater change in inclination to the peaks and the pit. We chose to compare how variation sampling based on the images computed with a 51×51 and 151×151 large neighborhood.

The extraction of initial points was performed for different values of t_v. Each rectangle in the quadtree based split is divided into four new rectangles as long as the largest variation in the variation image σ within the rectangle is less than t_v. We have set the threshold t_c of number of pixels within each rectangle as $t_c = 1000$ to avoid too many points in the simplified point set.

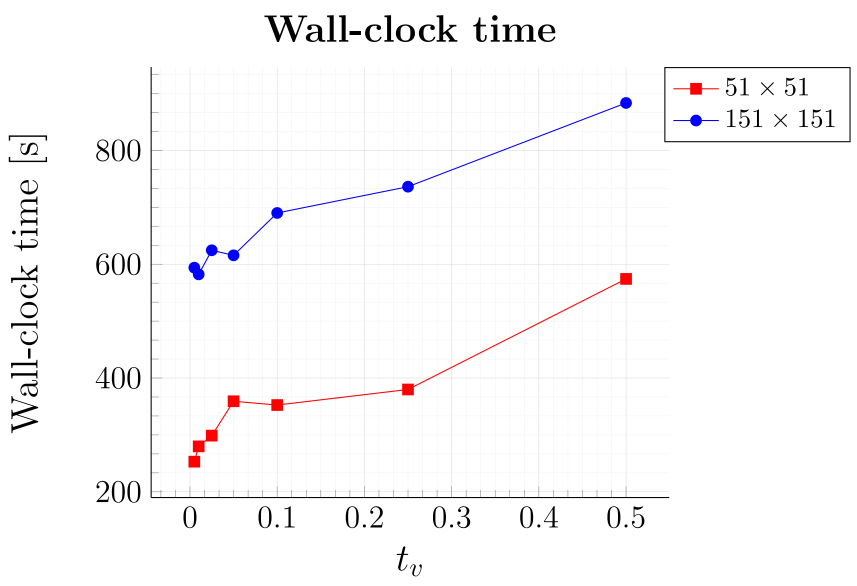


Figure 4.11: The measured wall-clock time of how long the refinement based on variation sampling spent to converge. The variation sampling was based on the variation images computed with neighborhood sizes of 51×51 and 151×151.

The variation sampling based on the variation image computed by a 151×151 large neighborhood made the refinement algorithm converge poorly in terms of wall-clock time compared to refinement without a set of initial points (Figure 4.11). Hence, we decided to perform the variation sampling based on the 51×51 image only. Table 4.7 shows our obtained results for some selected values of t_v.

Table 4.7: How the different t_v affected the point simplification of a set containing 1000×1000 points. The RMSE threshold was 10^{-3}. The wall-clock time was averaged over three runs of the same experiment. We emphasize that our implementation of computing a variation image is parallel and utilized the six cores of our processor, see Appendix A.

t_v	Number of initial points	Number of insertions	Measured wall-clock time [s]
0.005	1196	4421	252.94
0.01	1118	4514	279.88
0.025	1025	4480	298.74
0.05	965	4409	359.13
0.1	881	4395	352.49
0.25	620	4371	379.85
0.5	242	4275	574.17

The refinement converged slower in terms of wall-clock time for $t_v = 0.01, 0.025$ than for refinement with initial set of points, if we take into account that refinement with uniformly and randomly sampled initial points had a standard deviation of 31 seconds in wall-clock time. This is interesting, as for these values of t_v the number of points in the initial sets from variation sampling were greater than the uniformly and randomly sampled points.

It is difficult to find the optimal t_v for our data set based on Table 4.7. The number of points in the simplified point sets from variation sampling were larger than the point sets obtained from the random sampling except for when $t_v = 0.5$. For $t_v = 0.5$ the refinement algorithm was only 15 percent faster than refinement without any initial set of points. However, the refinement with randomly sampled initial points was on average approximately 57.9 percent faster than refinement without an initial set of points. For $t_v = 0.25$, the measured time was lower than for refinement without any set of initial points, but the returned point set contained more points than refinement with randomly sampled initial points. We regard $t_v = 0.25$ as the threshold that made the best set of initial points from variation sampling. For $t_v = 0.25$, the refinement algorithm converged faster in terms of wall-clock time and did not introduce as many points as for the lower values of t_v.

4.3.3 The Ripples test function

The Ripples test function have more abrupt changes in its topography than Franke's test function. It consists of four regions; a flat region R_1 with constant elevation, a region R_2 where two rectangles on top of each other are located, a rapidly varying region R_3, and a region R_4 with an inclination of a flat surface. This test function made it possible to examine how the algorithms handles surfaces with regions that have different features. It is of interest to see how the edges with 90° depression angle are handled by the algorithms, as they are the most extreme cases of cliffs. In our studies, we used the Ripples test function to generate point sets that contained 40000 points. The only exception is the study in Section 4.3.3.3, where we applied the refinement algorithm to point sets that contained different amount of points.

4.3.3.1 The number of points after reaching an error threshold

The algorithms returned point sets that satisfied a set of error thresholds for RMSE and max deviation. The selected thresholds were the same for both of the error assessments. The error thresholds were 10^{-k} for $k = 1, \dots, 7$. The results for AT1, AT3, and the refinement algorithm are presented in Table 4.8, Table 4.9, and Table 4.10, respectively.

Table 4.8: The number of points in the simplified sets obtained from AT1 and the percentage of points from the input point set.

Error threshold	Number of points in simplified set, RMSE	Percentage in size of original point set, RMSE	Number of points in simplified set, max deviation	Percentage in size of original point set, max deviation
1×10^{-7}	10179	25.448	10185	25.462
1×10^{-6}	10036	25.09	10185	25.462
1×10^{-5}	9702	24.255	10098	25.245
1×10^{-4}	7825	19.562	9982	24.955
1×10^{-3}	4064	10.16	5298	13.245
1×10^{-2}	1072	2.68	4036	10.09
1×10^{-1}	113	0.282	559	1.398

Table 4.9: The number of points in the simplified sets obtained from AT3 and the percentage of points from the input point set.

Error threshold	Number of points in simplified set, RMSE	Percentage in size of original point set, RMSE	Number of points in simplified set, max deviation	Percentage in size of original point set, max deviation
1×10^{-7}	11131	27.828	11132	27.83
1×10^{-6}	11126	27.815	11132	27.83
1×10^{-5}	11024	27.56	11131	27.828
1×10^{-4}	10665	26.662	11127	27.818
1×10^{-3}	8436	21.09	10776	26.94
1×10^{-2}	2785	6.962	8292	20.73
1×10^{-1}	496	1.24	2137	5.342

Table 4.10: The number of points in the simplified sets obtained from the refinement algorithm and the percentage of points from the input point set.

Error threshold	Number of points in simplified set, RMSE	Percentage in size of original point set, RMSE	Number of points in simplified set, max deviation	Percentage in size of original point set, max deviation
1×10^{-7}	10269	25.672	10241	25.602
1×10^{-6}	10125	25.312	10241	25.602
1×10^{-5}	9883	24.708	10154	25.385
1×10^{-4}	8166	20.415	10025	25.062
1×10^{-3}	4208	10.52	6093	15.232
1×10^{-2}	1152	2.88	3898	9.745
1×10^{-1}	169	0.422	776	1.94

The algorithms returned point sets that had at most approximately 27.8 percent of the points from the input point set (Table 4.8, Table 4.9, and Table 4.10). The AT3 returned point sets that contained more points than AT1 and refinement to satisfy the thresholds for RMSE and max deviation (Table 4.9).

To get a better overview of the results from Table 4.8, Table 4.9, and Table 4.10, the results from the tables are plotted together. The plots Figure 4.12 and Figure 4.13 shows the amount of points in the point sets that satisfied the error thresholds for RMSE and max deviation, respectively.

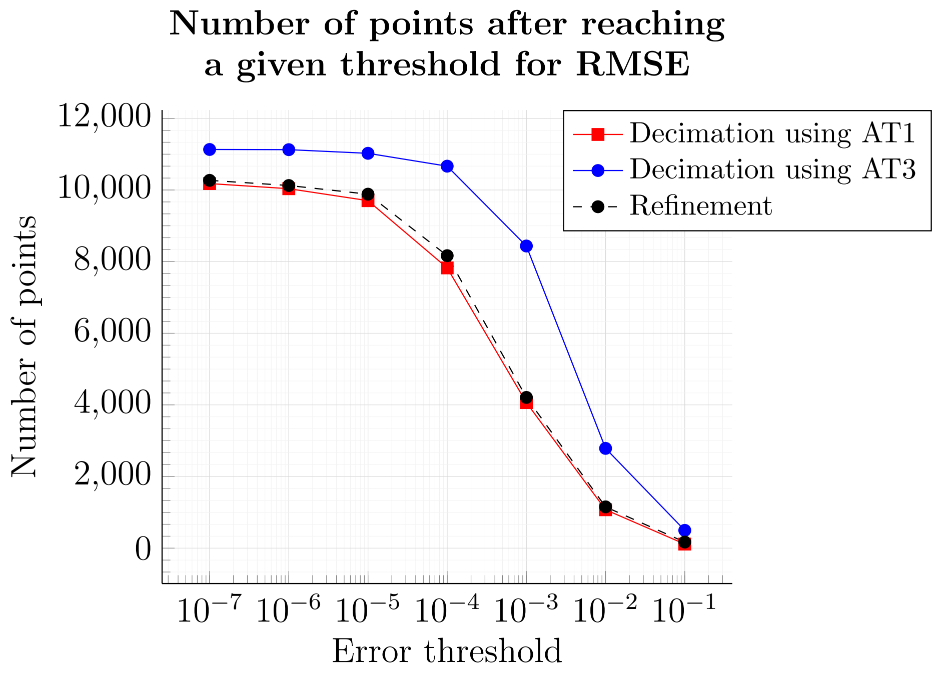


Figure 4.12: The amount of points in the point sets that satisfied a selection of RMSE thresholds.

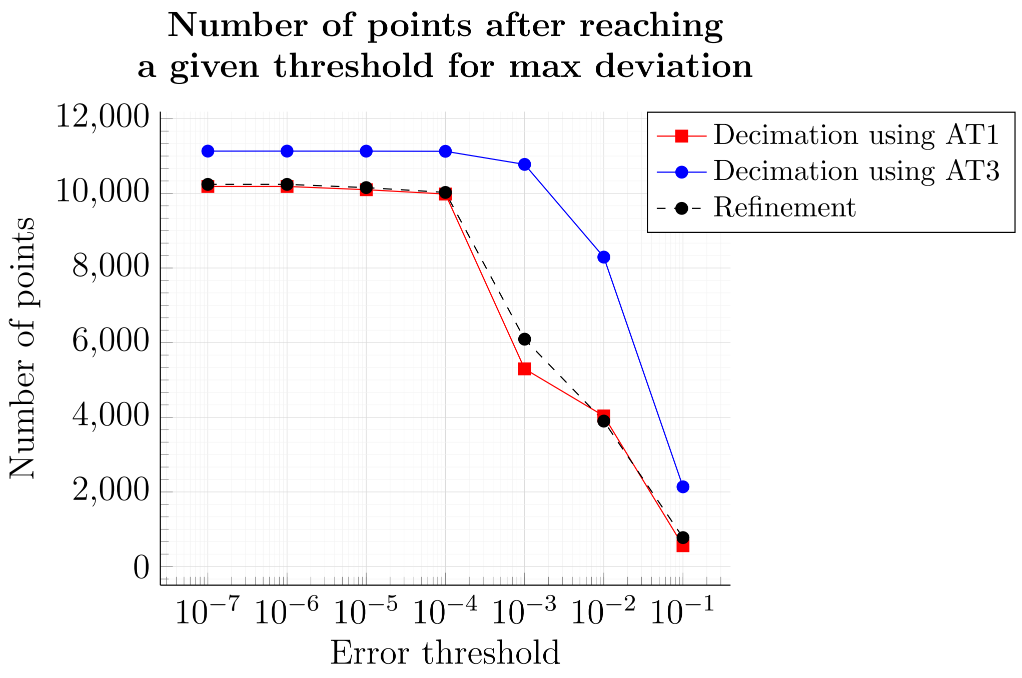


Figure 4.13: The amount of points in the point sets that satisfied a selection of max deviation thresholds.

The amount of points in the point sets that satisfied the RMSE thresholds and max deviation thresholds for each algorithm differed (Figure 4.12 and Figure 4.13). The AT3 returned larger point sets than AT1 and refinement. The largest difference in amount of points between the point sets from AT3 and the point sets from AT1 and refinement was at the error threshold 10^{-3} for RMSE and max deviation. We considered the RMSE in our experiments. We chose the RMSE such that our results can be compared with the results from Section 4.3.2.

It is of interest to consider how the triangulated points sets from AT1, AT3, and refinement looks like for RMSE threshold of 10^{-3} and 10^{-2}. The point set from AT3 contained approximately the double number of points than the point sets from AT1 and refinement (Figure 4.12). The point sets that satisfied RMSE threshold of 10^{-2} are of interest since they contained less than ten percent of points from the input point set. The triangulated point sets that satisfied the RMSE thresholds of 10^{-3} and 10^{-2} are shown in Figure 4.14 and Figure 4.15, respectively.

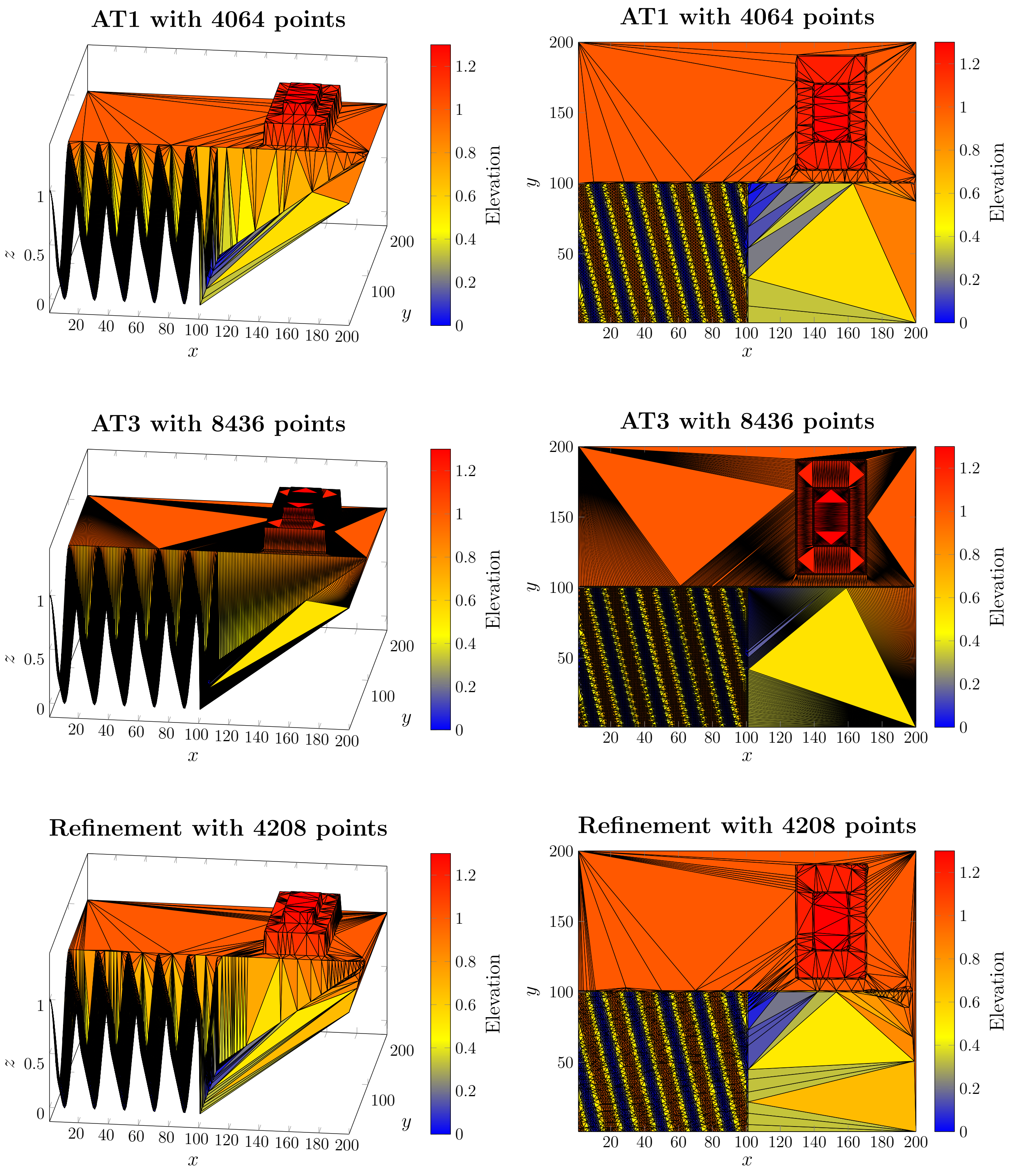


Figure 4.14: The point sets that satisfied the RMSE threshold 10^{-3}.

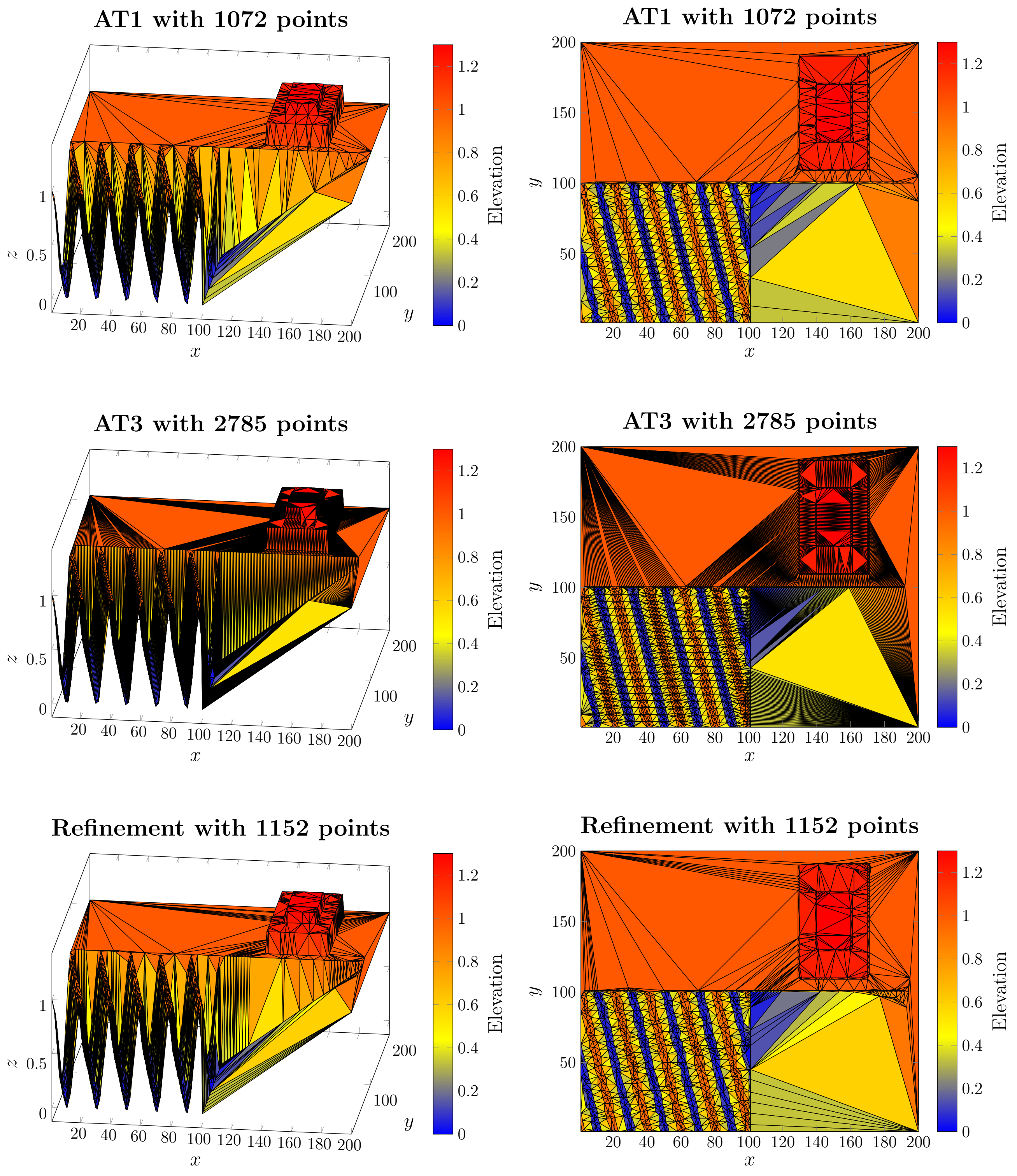


Figure 4.15: The point sets that satisfied the RMSE threshold 10^{-2}.

All of the algorithms preserved the rectangles in region R_2. The edges to the boxes have a high density of points even though it is sufficient to represent each edge with four points, see Figure 4.16 for an example of a dense representation of an edge. This applies also to the cliff between the regions R_1 and R_2 and the regions R_3 and R_4. The rapidly varying region R_3 is well represented in the triangulated point sets. The flat surfaces in the regions R_1 and R_4 contains fewer points than R_2 and R_3. There is a small arc-shaped artifact of points in the point sets from refinement in region R_4, see Figure 4.16. This is an artifact because there is nothing on the flat surface that needs to be represented by these points.

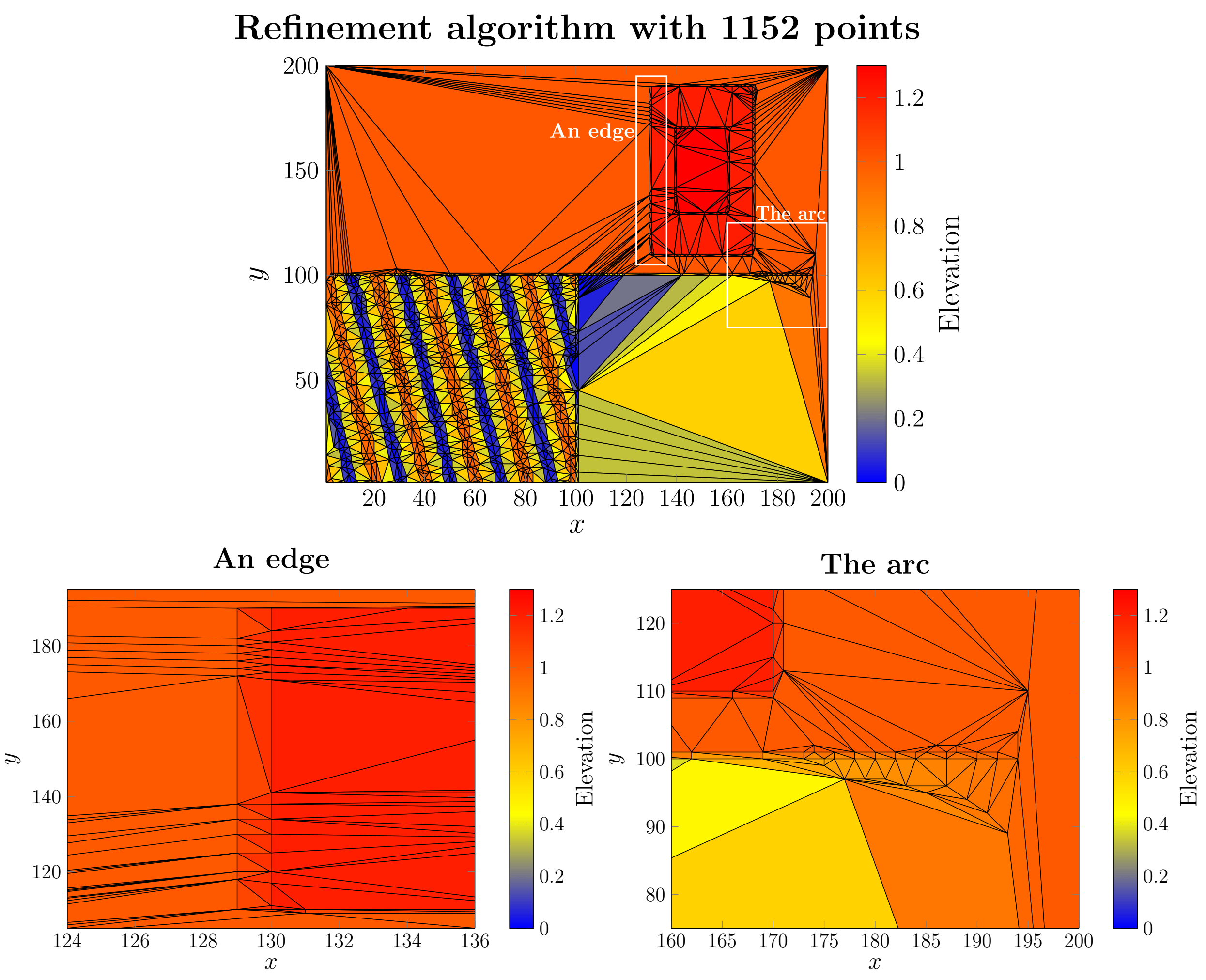


Figure 4.16: *Top*: An edge and the arc in located in the white boxes. *Lower left:* An example of a dense representation of an edge. *Lower right:* The arc that was present in the point sets from refinement.

The triangles in the triangulated point sets from AT3 are longer and thinner, where most of them are so thin that they appears as black areas in the plots. It is not clear from the plots alone whether this is a consequence on how AT3 is defined or because AT3 returned point sets that had approximately the double amount of points than the point sets obtained from AT1 and refinement.

4.3.3.2 Distribution of the longest edges and smallest angles to the triangles in a triangulation

Similarly as done in Section 4.3.2.2, we chose to compute the smallest angles α and the normalized longest edges l to each triangle in a selection of triangulated point sets from the simplification algorithms. The edge lengths were normalized by the largest possible edge length to the given raster data. We regard a long and thin triangle as a triangle with $\alpha \leq 5^\circ$ and $l \geq 0.4$. We chose to let the algorithms simplify a point set of 40000 points down to point sets of 4000 points and 400 points. The scatter plots of l and α to the triangulated point sets with 4000 points and 400 points are shown in Figure 4.17 and Figure 4.18, respectively. The markers in the scatter plots are colored only to show better the distribution of the markers. The red boxes in the scatter plots shows the area of when a triangle is considered as long and thin.

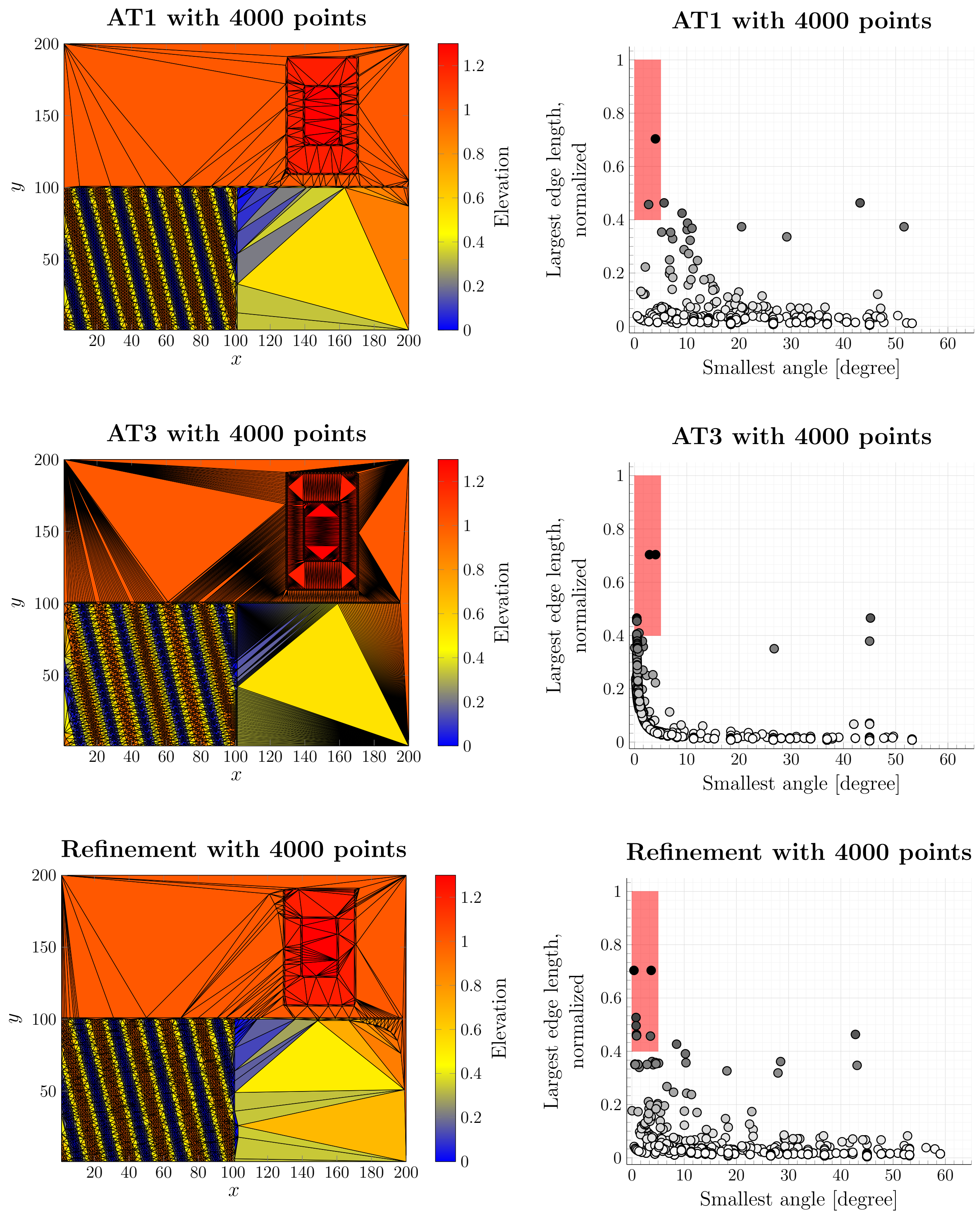


Figure 4.17: The simplified point sets with 4000 points. *Left column:* The triangulated point sets viewed from above. *Right column:* The scatter plots of the triangulations.

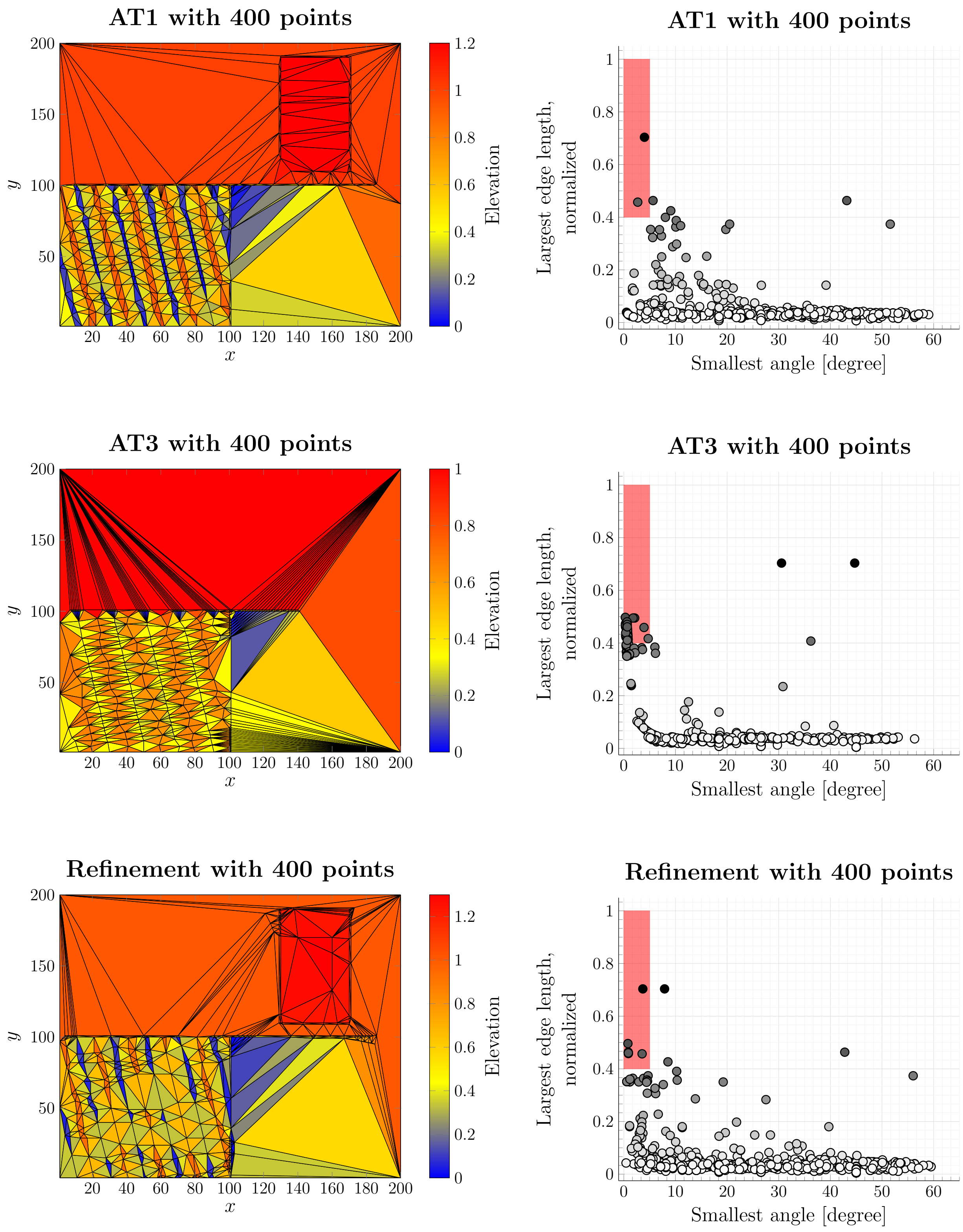


Figure 4.18: The simplified point sets with 400 points. *Left column:* The triangulated point sets viewed from above. *Right column:* The scatter plots to each triangulation.

The AT3 seems to return point sets that forms more long and thin triangles when triangulated than the triangulated point sets from AT1 and refinement (Figure 4.17 and Figure 4.18). We counted the number n_T of long and thin triangles and found percentage of long and thin triangles in the triangulations of N_T triangles in each point set, see Table 4.11.

Table 4.11: Table over long and thin triangles.

Algorithm	$\frac{n_T}{N_T}$, 400 points	Percentage of long and thin triangles, 400 points	$\frac{n_T}{N_T}$, 4000 points	Percentage of long and thin triangles, 4000 points
AT1	$\frac{2}{773}$	0.26	$\frac{2}{7896}$	0.03
AT3	$\frac{37}{781}$	4.74	$\frac{28}{7948}$	0.35
Refinement	$\frac{8}{777}$	1.03	$\frac{10}{7898}$	0.13

The AT3 returned points sets that contained the most long and thin triangles when triangulated (Table 4.11). The percentage of long and thin triangles are smaller in the triangulated point set from AT3 with 4000 points than the triangulated point set from AT3 with 400 points. However, the scatter plot to the triangulated point set from AT3 with 4000 points shows that there is a denser collection of triangles with $\alpha \leq 5°$ compared to the triangulated point sets that contains 4000 points from AT1 and refinement (Figure 4.17).

4.3.3.3 Variation sampling to construct an initial subset of points for refinement

The AT1 and refinement returned point sets that contained almost the same number of points (Figure 4.12). Similar results were obtained for Franke's test function (Figure 4.4). Variation sampling constructed initial point sets that made refinement inferior to refinement with sets of uniformly and randomly sampled initial points when applied to the point sets generated by Franke's test function (Section 4.3.2.3). It was not clear from the experiments on Franke's test function whether it was the definition of variation sampling that constructed initial point sets inferior to randomly sampled initial points. It could happen that it was the smooth surface of the Franke's test function that made the variation sampling excessive to use or we did not test for the correct parameters. We performed the same tests on the point sets generated by the Ripples test function to see how the variation sampling made the refinement perform. We used the same parameters as we tested for with Franke's test function to see if it was the slowly varying surface of Franke's test function that made variation sampling inferior to the uniformly and randomly sampled initial points.

The refinement algorithm returned simplified point sets that satisfied RMSE error threshold of 10^{-3}. The input point sets contained different amount of points. The input point sets contained $n \times n$ points for $n = 250$, 500, 750, 1000. For each n, the simplification was repeated three times. The wall-clock time was measured for each repeated simplification and thereafter averaged. The results are shown in Table 4.12.

Table 4.12: The measured wall-clock time in seconds of how long it took the refinement algorithm to reach a RMSE error threshold of 10^{-3} without an initial subset of points.

$n \times n$	Number of points	Average wall-clock time [s]
250×250	6176	63.12
500×500	20080	388.6
750×750	37052	1,635.07
1000×1000	54223	3,473.02

The refinement algorithm was given an initial set of 1000 uniformly and randomly sampled points and simplified the input point set such that it satisfied a RMSE threshold 10^{-3}. We measured the wall-clock time and counted the number of points in the simplified point sets for ten unique sets of initial points. We used ten different initial point sets to examine whether the distribution of the input points had an effect of the convergence to the refinement algorithm in terms of wall-clock time or the number of points in the returned point sets. Our results are presented in Table 4.13.

Table 4.13: The measured wall-clock time in seconds of how much it took the refinement algorithm to reach a RMSE error threshold of 10^{-3}.

$n \times n$	Average number of points	Standard deviation, number of points	Average wall-clock time [s]	Standard deviation, wall-clock time [s]
250×250	7118.2	40.87	16.97	0.49
500×500	21243.4	71.06	234.43	2.29
750×750	38375.5	107.16	927.8	4.73
1000×1000	55800.2	143.94	2,369.75	6.28

There is an improvement in terms of the average wall-clock time when the sets of randomly sampled initial points are passed to the refinement algorithm (Table 4.13). The refinement algorithm was approximately 31 percent faster on average than refinement without a set of initial points for the point set that with 1000×1000 points. The simplified point set from refinement with initial subset of points contained on average approximately 2.9 percent more points than the set obtained from refinement without an initial point. The measured wall-clock times had a relatively small standard deviations for all n. The amount of points in the simplified point sets varied more, but not more than 0.014 percent of points from the input point set.

Based on this, we examined whether variation sampling managed to improve the wall-clock time to refinement and at the same time made the refinement return a point set with less points. We chose the point set with 1000×1000 points as it was this the largest point set we tested for.

We chose to calculate the variation images based on four different neighborhood sizes. This was done to see how much effect each neighborhood size had to the estimated variation image. The calculated variation images that we obtained are shown in Figure 4.19.

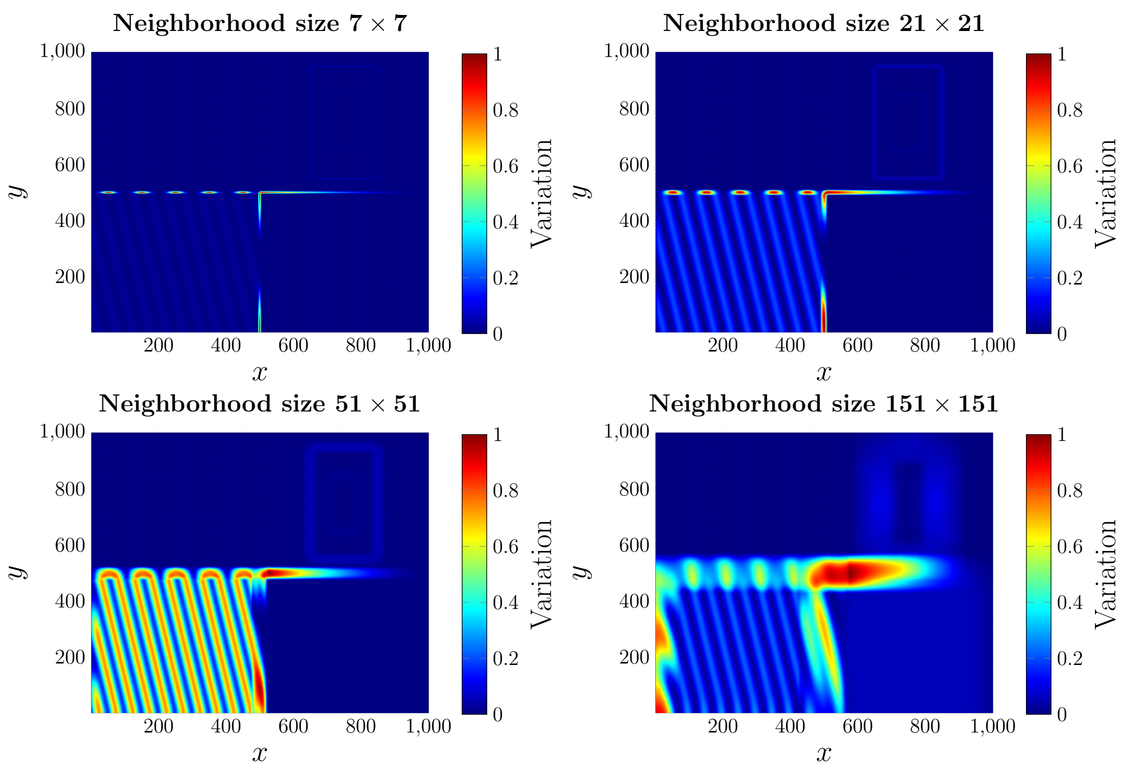


Figure 4.19: The computed variation images.

All of the variation images differs (Figure 4.19). We chose to base the variation sampling on the variation image calculated from a neighborhood size of 51×51. We think this variation image captured best the rapidly varying region R_3 and the edges of the rectangles in R_2 and the cliff separating the regions R_1 and R_2 and the regions R_3 and R_4. The variation sampling was performed by letting the maximum allowed number t_c of pixels within a rectangle to be $t_c = 1000$. We tested how the threshold t_v of maximum variation within each rectangle affected the number of points in the simplified point sets from refinement. We let the refinement algorithm construct point sets with RMSE just below 10^{-3}. The results are presented in Table 4.14.

Table 4.14: The RMSE threshold was 10^{-3}. The computation for each variation image was parallel and utilized the six cores of our processor, see Appendix A.

t_v	Number of initial points	Number of points	Measured wall-clock time [s]
0.005	848	55092	2,318.6
0.01	635	54686	2,325.82
0.025	602	54743	2,338.78
0.05	509	54695	2,397.83
0.1	491	54760	2,517.62
0.25	473	54928	2,518.08
0.5	434	54799	2,813.71

The refinement algorithm performed better with variation sampling for a selection of t_v than for an uniformly and randomly sampled set of initial points. The refinement algorithm returned point sets with less points than refinement with randomly sampled initial points and was faster in terms of wall-clock time for $t_v = 0.01$ and $t_v = 0.025$. For $t_v = 0.01$, refinement with variation sampling returned point sets that contained approximately two percent of points less than from the refinement with random sampling. We concluded that $t_v = 0.01$ constructed the best set of initial points to the refinement algorithm. For $t_v = 0.01$ the refinement algorithm with variation sampling converged faster than refinement without a set of initial points in wall-clock time. Refinement with random sampled initial points contained on average approximately 2.9 percent more points than refinement without initial points whereas refinement with variation sampled initial points contained 0.85 percent more points than refinement without initial points. The number of initial points from variation sampling was less than for the set of randomly chosen initial points.

In Section 4.3.2.3, the variation sampling was inferior to uniformly and randomly sampling of the initial points. We decided to omit considering the sampling of initial points in the forthcoming experiments. Since variation sampling is highly dependent on which parameters are being used, it was difficult to generalize the results we obtained from the the generated point sets.

4.4 Simplification of surfaces with coherence

Our simplification algorithms managed to simplify the point sets such that their triangulations resembled the input surface without considering any weighting. The estimated elevations in the bathymetric data from the InSAS we consider is associated with an estimated coherence. It is of interest to examine how the selection of points gets affected when coherence is considered in the algorithms. All of the studies in this section were performed on a point set that contained 40000 points.

4.4.1 Generated coherence

The generated coherence map C is based on Franke's function from Equation (4.1). The coherence map is constructed by a method described in (Burrough et al., 2015, pp. 244 - 245) and introduced in (Horn, 1981, p. 31). The method computes a reflectance map over a given surface. The reflectance map describes the shadows that will be present if a light source from a given location and depression angle shines on the surface. The surface that we computed the reflectance map of was generated by Franke's test function. We let the computed reflectance map be the coherence C associated to the data sets generated by Franke's test function and the Ripples test function. The values in the reflectance map is normalized such that each value was between zero and one inclusive, which also is the case for the coherence maps obtained from the InSAS we consider. The generated coherence map is shown in Figure 4.20.

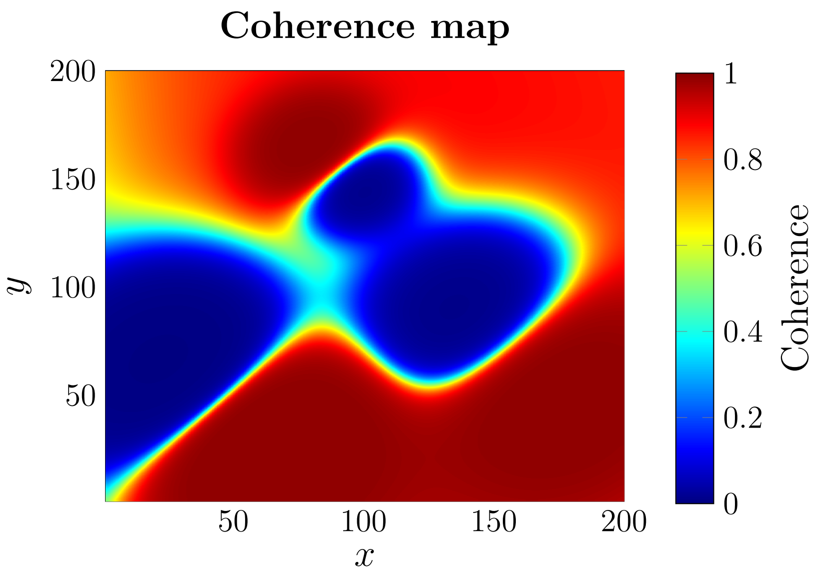


Figure 4.20: The reflectance map that we considered as the coherence to the generated point sets.

The coherence map is based on one surface such that it is possible to compare how much the topography matters in how the points are selected by the algorithms. The different topologies associated with the same weights makes it possible to examine how the interpolation errors and the weights are balanced by the simplification algorithms. The weights that were passed to the algorithms were computed based on Equation (2.6). All of the values that were above 0.99 in the coherence map were set to 0.99 to avoid dividing by zero in the computations of the weights. We emphasize that our implementations always includes the corner points of the input point set regardless of their weights (Section 3.1.1 and Section 3.1.2).

4.4.2 Franke's test function with coherence

The coherence map was based on Franke's test function and thereby followed the topography of the function. A question of interest, is how the algorithms selects the points from the input point set with respect to the coherence map.

The point sets that satisfied wRMSE of 10^{-3} and 10^{-2} differed by a factor ten in percentage of points from the input point set (Table 4.15). It is therefore of interest to examine how the distributions of these point sets were.

Table 4.15: The percentage of points from the input point set in the simplified point sets satisfying wRMSE threshold of 10^{-3} and 10^{-2} from the algorithms.

wRMSE threshold	Percentage of points, wAT1	Percentage of points, wAT3	Percentage of points, weighted refinement
1×10^{-3}	4.86	7.15	6.69
1×10^{-2}	0.2	0.35	0.44

We plotted how the points were distributed at the coherence map for point sets that satisfied wRMSE threshold of 10^{-3} and 10^{-2} in Figure 4.21 and Figure 4.22, respectively. The markers at the coherence maps are the included points in the simplified point sets. The markers are colored by the associated coherence to the points they represent.

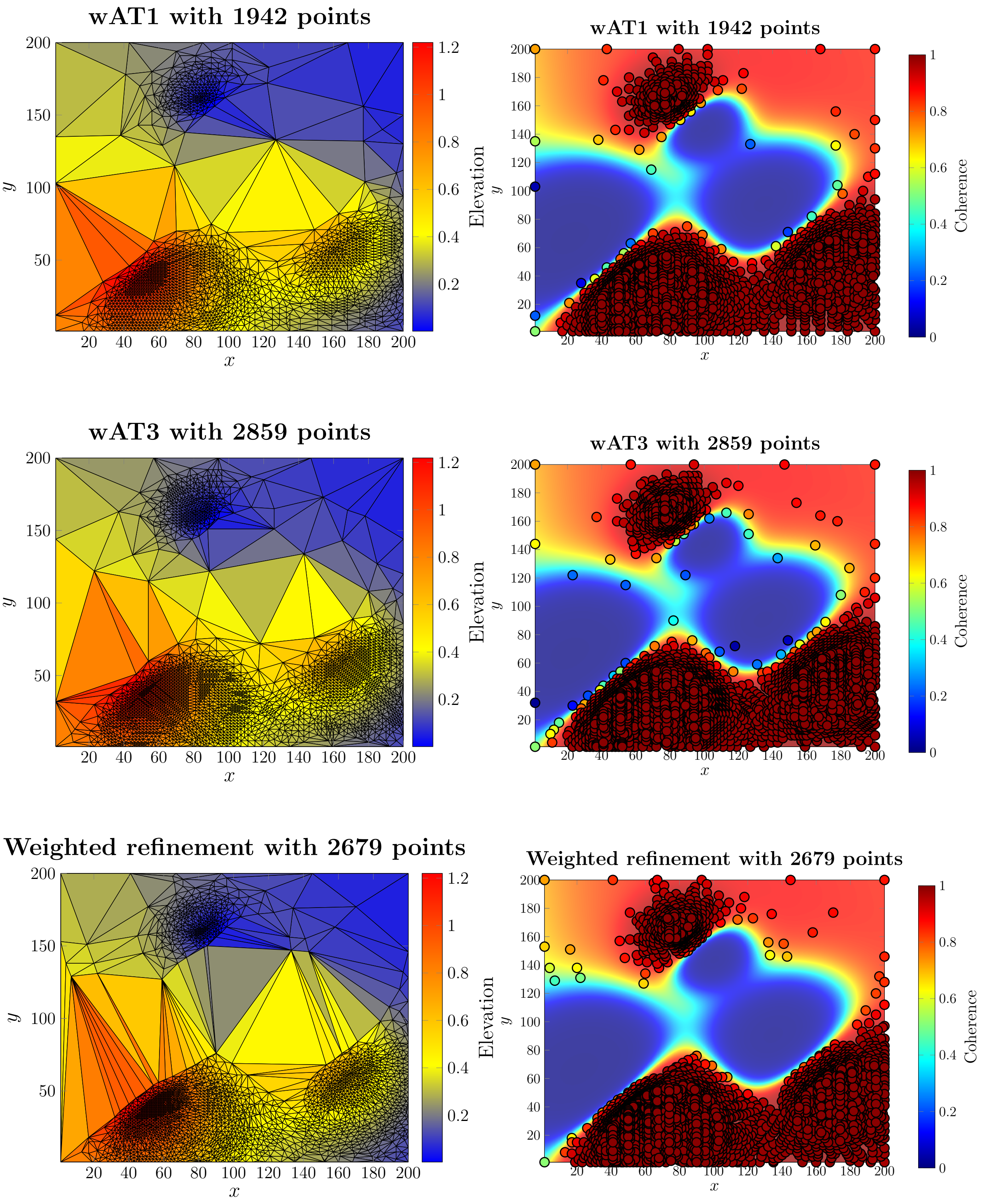


Figure 4.21: The simplified point sets that has a wRMSE error just below 10^{-3}. *Left column:* The obtained triangulations of the simplified point sets. *Right column:* The coherence to each point at the coherence map.

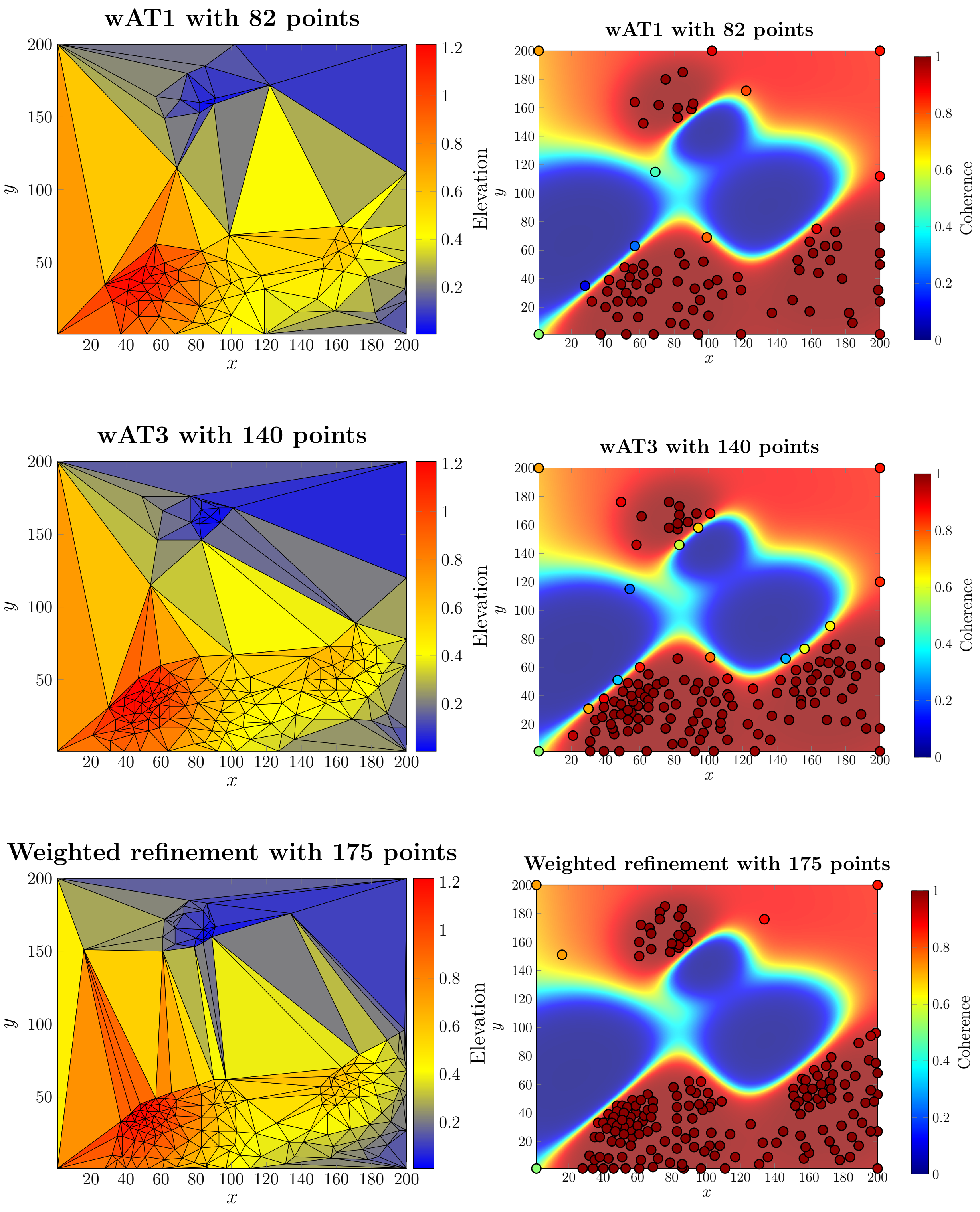


Figure 4.22: The simplified point sets that has a wRMSE error just below 10^{-2}. *Left column:* The obtained triangulations of the simplified point sets. *Right column:* The coherence to each point at the coherence map.

The wAT1 returned point sets that contained the least amount of points for both of the wRMSE thresholds. The weighted refinement returned more points than wAT3 for wRMSE threshold of 10^{-2}. The amount of points that weighted refinement returned for wRMSE of 10^{-3} was closer to wAT3 than wAT1.

The refinement seemed to choose more points that were associated with C closer to one than AT1 and AT3 (Figure 4.21 and Figure 4.22). We computed the percentage of points associated with equal to or more than 0.9 in coherence for a selection of number of desired points in the simplified points. We counted the number of points where $C \geq 0.9$ in the simplified point sets that contained between 100 and 10000 points inclusive, see Table 4.16.

Table 4.16: The percentage of points in the simplified point sets associated with $C \geq 0.9$.

Number of points in simplified point set	Percentage of points where $C \geq 0.9$, wAT1	Percentage of points where $C \geq 0.9$, wAT3	Percentage of points where $C \geq 0.9$, weighted refinement
100	35.08	35.08	97.12
500	35.43	35.43	97.82
1500	36.35	36.35	96.54
2500	37.32	37.32	96.84
5000	39.99	39.99	97.14
7500	43.06	43.06	96.86
10000	46.63	46.64	96.17

The weighted refinement favored points associated with $C \geq 0.9$ (Table 4.16). The weighted refinement returned points sets where between 96 to 97 percent of the points had $C \geq 0.9$ whereas wAT1 and wAT3 returned point sets where between 35 to 46 percent of the points had $C \geq 0.9$.

4.4.3 Ripples test function with coherence

The coherence map was associated to the Ripples test function to examine how the algorithms managed to balance between interpolation error and weighting of the points. A set of 40000 points generated by the Ripples test function was considered in this study. Similarly as we did in Section 4.4.2, we considered a selection of wRMSE thresholds. The simplified point sets that satisfied wRMSE thresholds of 10^{-3} and 10^{-2} did not differ that much in terms of size as they did for Franke's test function (Table 4.17).

Table 4.17: The percentage of points from the input point set in the simplified point sets satisfying wRMSE threshold of 10^{-3} and 10^{-2} from the algorithms.

wRMSE threshold	Percentage of points, wAT1	Percentage of points, wAT3	Percentage of points, weighted refinement
1×10^{-3}	11.22	14.56	12.57
1×10^{-2}	3.39	9.93	4.63

We plotted the triangulated point sets that satisfied the wRMSE thresholds of 10^{-3} and 10^{-2} in Figure 4.23 and Figure 4.24, respectively. The markers at the coherence maps are the included points in the simplified point sets. The markers are colored by the associated coherence to the points they represent.

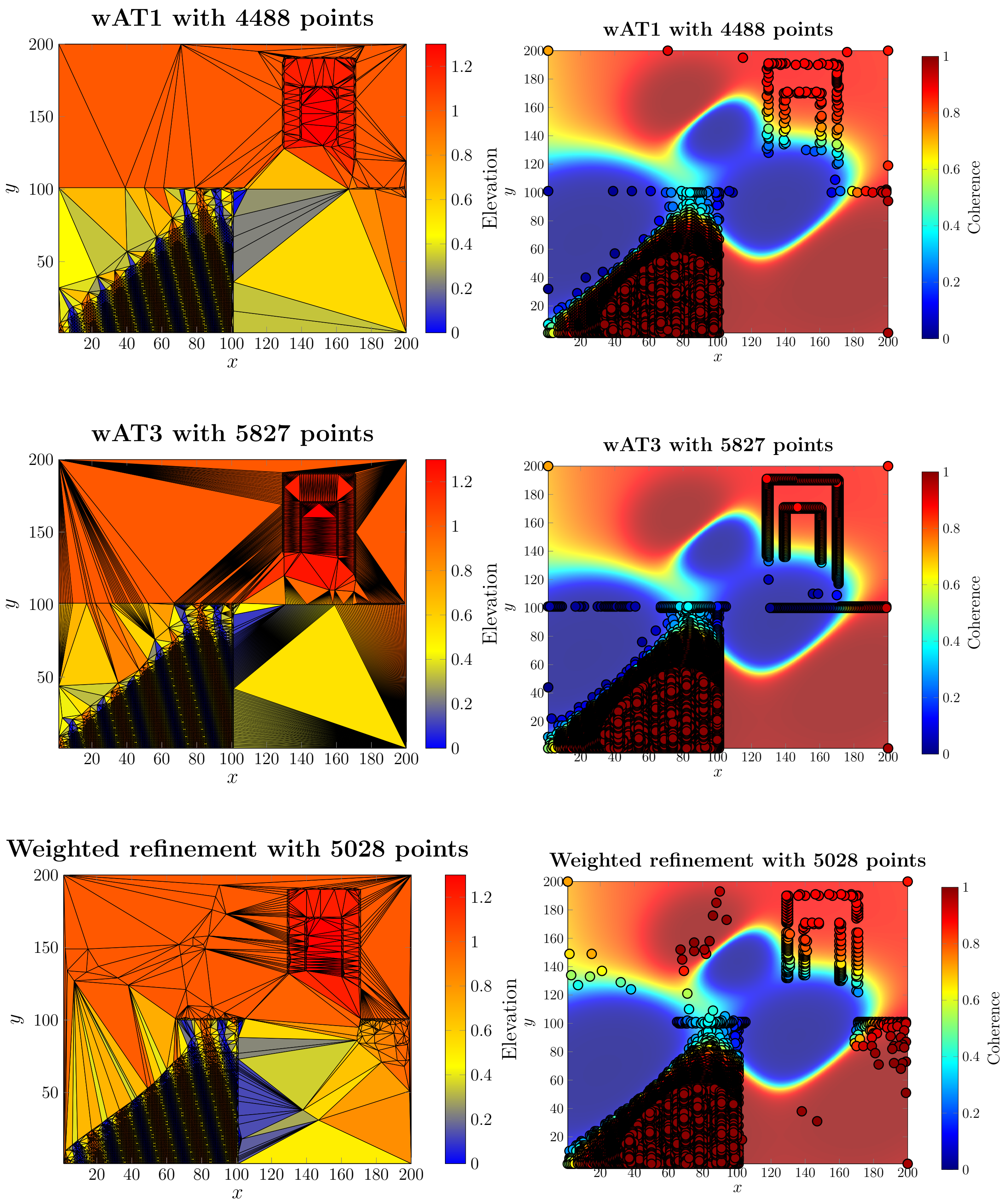


Figure 4.23: The simplified point sets that has a wRMSE error just below 10^{-3}. *Left column:* The obtained triangulations of the simplified point sets. *Right column:* The coherence to each point at the coherence map.

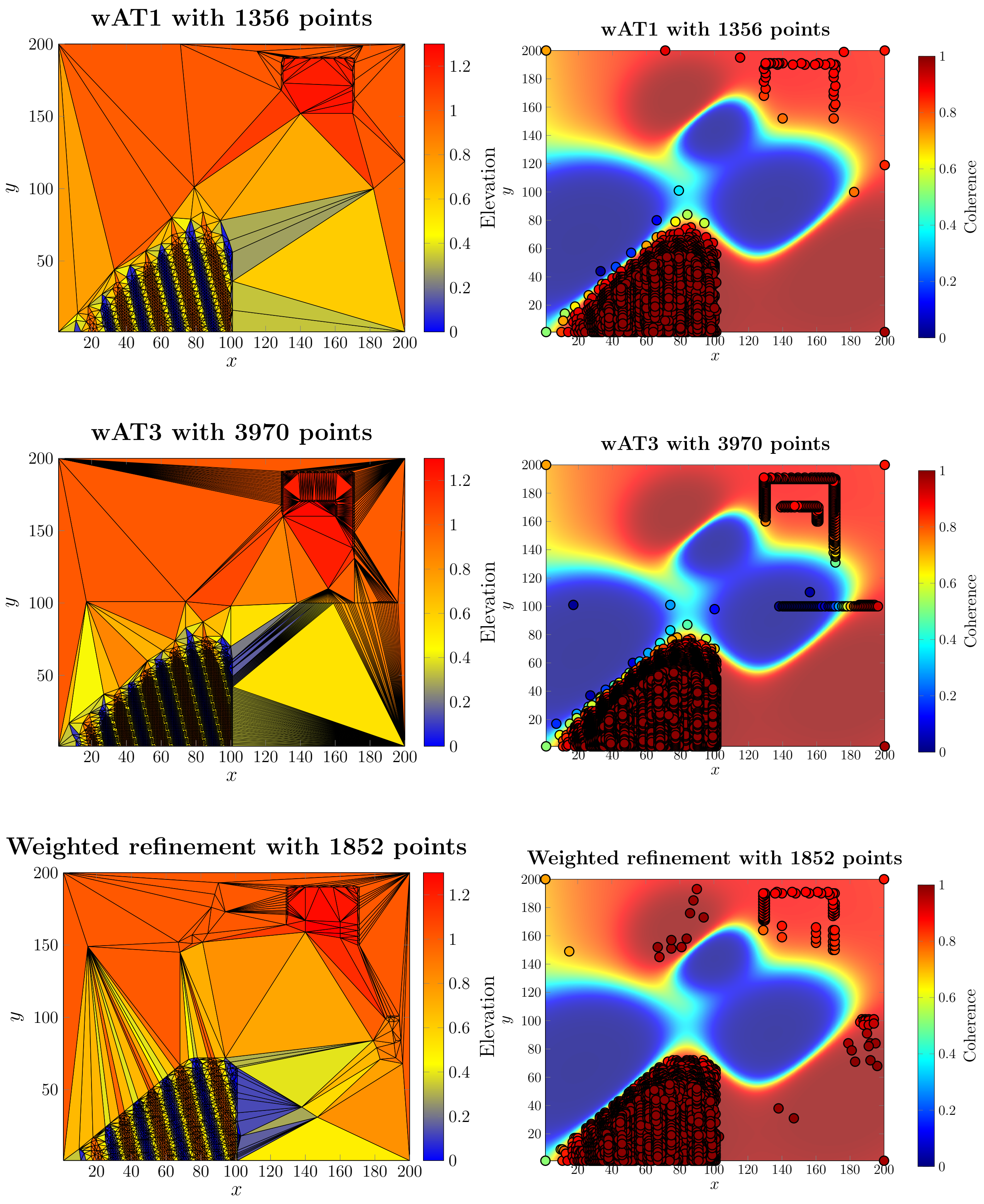


Figure 4.24: The simplified point sets that has a wRMSE error just below 10^{-2}. *Left column:* The obtained triangulations of the simplified point sets. *Right column:* The coherence to each point at the coherence map.

The weighted refinement returned point sets with amount of points closer to the number of points from wAT1 than it did for the Franke's test function (Section 4.4.2). However, it seems that weighted refinement chose points at the coherence map that is closer to one than wAT1. The coherence maps shows that the decimation algorithms keeps points along the edges even though the points are associated with low coherence. The weighted refinement, however, seemed to favor points that had high coherence.

The algorithms selected points that mainly were associated with $C \geq 0.8$, which can be seen as the red markers in the coherence maps from Figure 4.23 and Figure 4.24. Some points along the cliff between the regions R_1 and R_2 and the regions R_3 and R_4 were chosen, even though they had coherence below 0.5 (Figure 4.23 and Figure 4.24).

From Figure 4.23 and Figure 4.24 it is not clear whether refinement selected more points associated with $C \geq 0.9$ than wAT1 and wAT3. The algorithms returned point sets with high point density in R_3 and some areas of the rectangles in R_2 with coherence close to one. The wAT1 and wAT3 chose to distribute some of the points at areas with low coherence where the interpolation error dominated. This is especially apparent for wAT3 in Figure 4.24 where it chose to keep the points along the cliff even though many of the points had coherence below 0.5. The percentage of points in a selection of simplified point sets associated with $C \geq 0.9$ is shown in Table 4.18. The weighted refinement returned point sets with more than 90 percent of points associated with $C \geq 0.9$ for all but two of the selected sizes. However, the two point set sizes that did not contain over 90 percent points associated with $C \geq 0.9$ contained more points with $C \geq 0.9$ than the points from wAT1 and wAT3.

Table 4.18: The percentage of points in the simplified point sets where $C \geq 0.9$.

Number of points in simplified point set	Percentage of points where $C \geq 0.9$, wAT1	Percentage of points where $C \geq 0.9$, wAT3	Percentage of points where $C \geq 0.9$, weighted refinement
100	13.35	13.35	97.12
500	13.49	13.49	99.4
1500	13.84	13.84	99.53
2500	14.21	14.21	98.28
5000	15.22	15.22	92.67
7500	16.36	16.39	66.15
10000	17.73	17.76	49.62

4.5 Acquired data from an InSAS

In this section, we consider a raster DEM acquired by an InSAS, where the DEM was associated with a coherence map. The weighted algorithms were applied to the data set. We examined how much the algorithms reduced the input bathymetry and how well they preserved the characteristics of the seabed.

4.5.1 The data set

We consider a data set referred to as the “Sand Dunes” data set due to the wave pattern in the region at approximately 80 meters depth, which is shown as the yellow region in Figure 4.25. A sonar image of the seabed is shown in Figure 4.27 and provides an idea of how the considered seabed looked like. The data was acquired by the system described in Section 2.1.1. The y-axis represents the along-track to the vehicle the InSAS was mounted on and the x-axis represents the cross-track to the vehicle. The cross-track distance to the center of the scene is 65 meters. The vehicle depth was approximately 47 meters during the data collection.

The data set contained originally 4001×3001 points. The coherence was estimated using a 9×9 large window in Equation (2.4). Hence, we chose to resample the input data with the corresponding factor. There are some sharp peaks in the Sand Dunes data set that we refer to as *needles*. They are likely present due to erroneous estimation of the true phase in a phase-unwrapping algorithm.

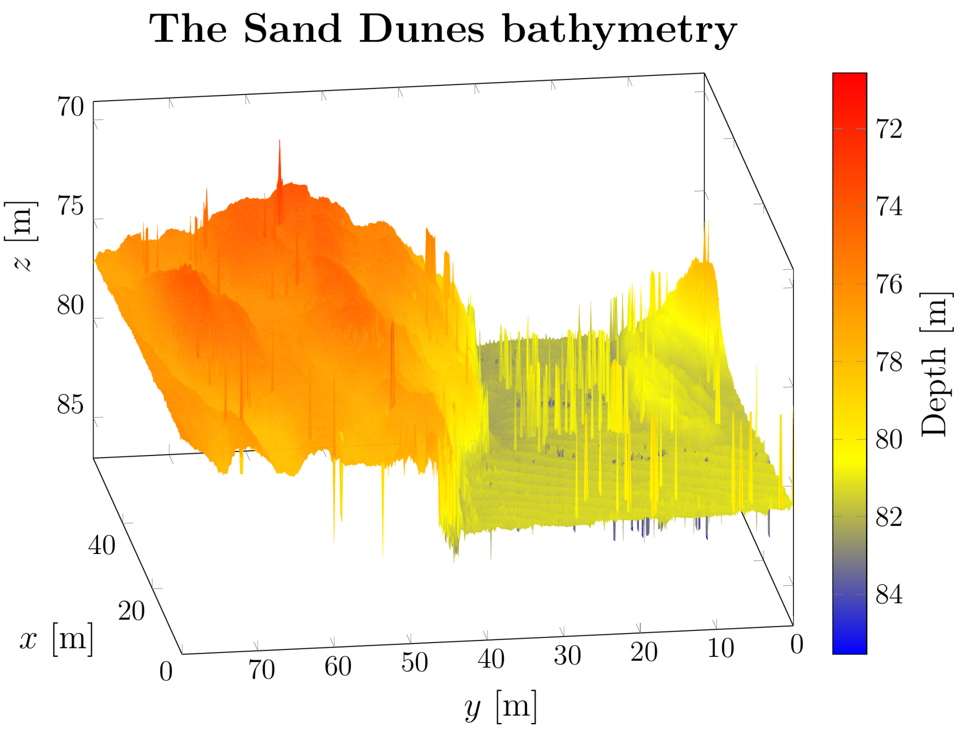


Figure 4.25: The Sand Dunes bathymetry resampled with a factor nine along each axis. The total number of points in the resampled set is 148630. The depth is measured relative to vehicle.

The Sand Dunes data set can be divided into two regions. The data with approximately 76 meters depth and shown as orange and red at the left hand side in Figure 4.25 seems to be dominated by rocks. To the right hand side in Figure 4.25, most of the region contains wave-looking variation that can be interpreted as sand dunes. This region has more finer details because of the sand dunes than the left region that contains relatively large rocks.

To reduce the amount of needles in our data set, we applied a median filter of size 3×3 after the resampling was performed. We chose this filter size to keep some of the needles since it is of interest to examine how the weighted algorithms handles the needles. We padded the bathymetry symmetrically along the borders such that the filtered bathymetry had the same size as the resampled bathymetry. The result is shown in Figure 4.26. We used the median filtered Sand Dunes throughout this study.

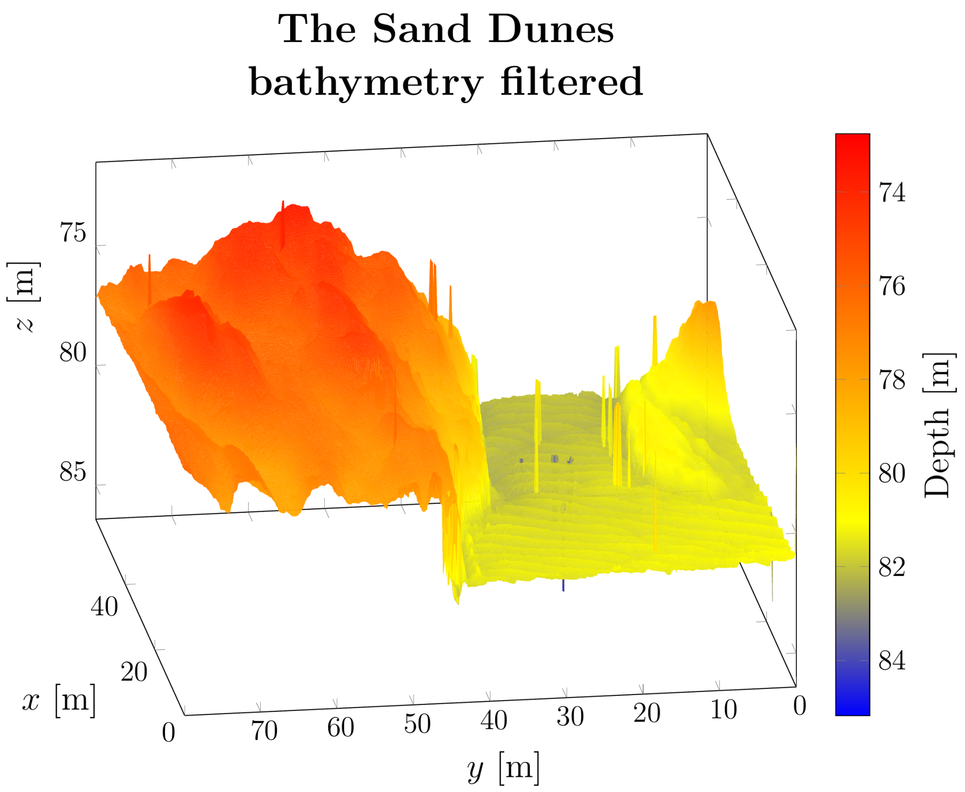


Figure 4.26: The median filtered Sand Dunes.

The associated coherence map was resampled by taking out every ninth sample and remained unfiltered. The coherence map is shown in Figure 4.28.

In practice, points that are associated with a coherence lower than a given threshold can be disregarded, which has been done in several studies by e.g (Sæbø, 2010). The threshold can depend on the bathymetric data and for which application it is supposed to be used for. By disregarding the points associated with low coherence, some of the needles and other data not representative of the seabed could be removed. However, some of the non-representative data of the seabed can also have coherence close to one. This can happen if for instance a phase-unwrapping algorithm has estimated the wrong phase to its given interferogram. We also set the coherence greater than 0.99 equal to 0.99 to avoid dividing diving by zero to compute the weights in Equation (2.6). This clipping lets us also avoid the algorithms to favor too much signals with coherence greater than 0.99; coherence of 0.99 yields SNR of approximately 100 whereas coherence of 0.999 yields a SNR of 1000. We refer to the point sets that contains points associated with a coherence greater than or equal to a given threshold as *thresholded point sets*. When necessary, we also say which threshold the point sets are thresholded with.

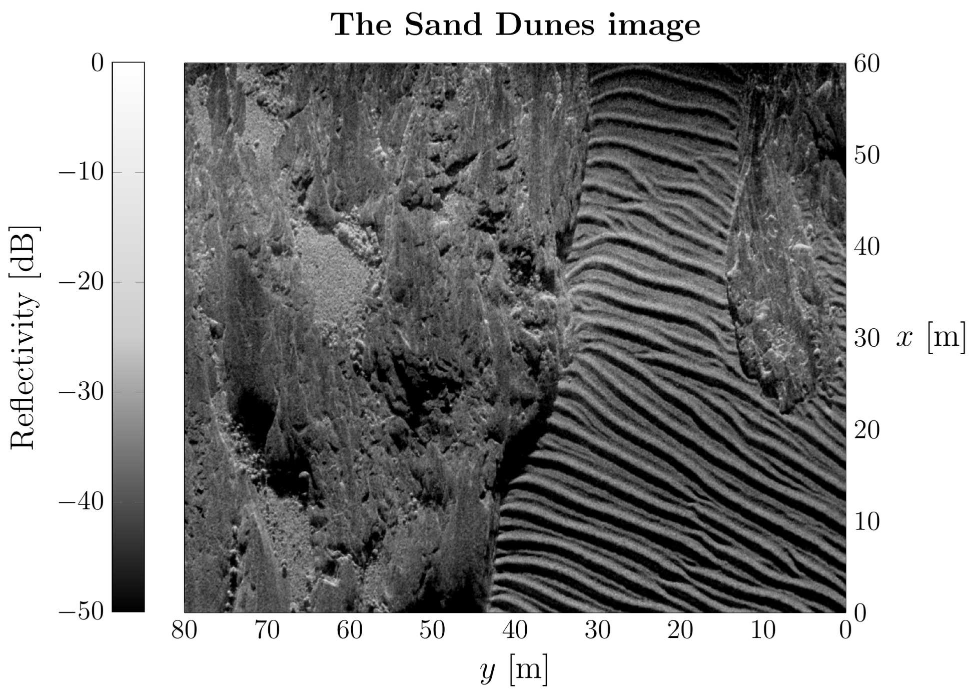


Figure 4.27: Sonar image of the Sand Dunes. The image has been filtered with a 9×9 median filter to reduce the presence of noise.

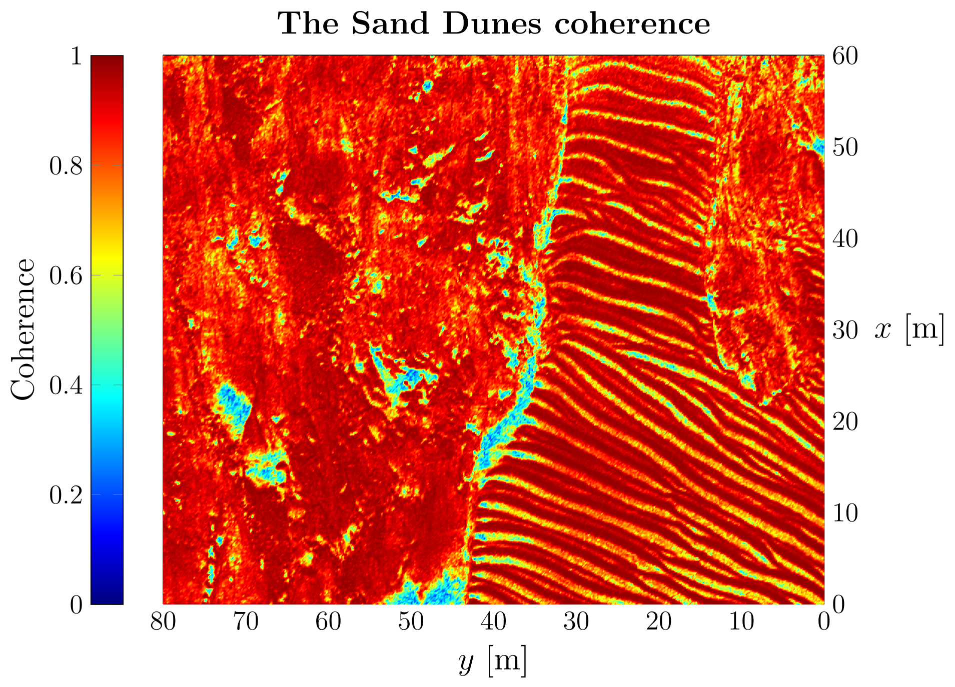


Figure 4.28: The considered coherence map.

4.5.2 Simplification of Sand Dunes with needles

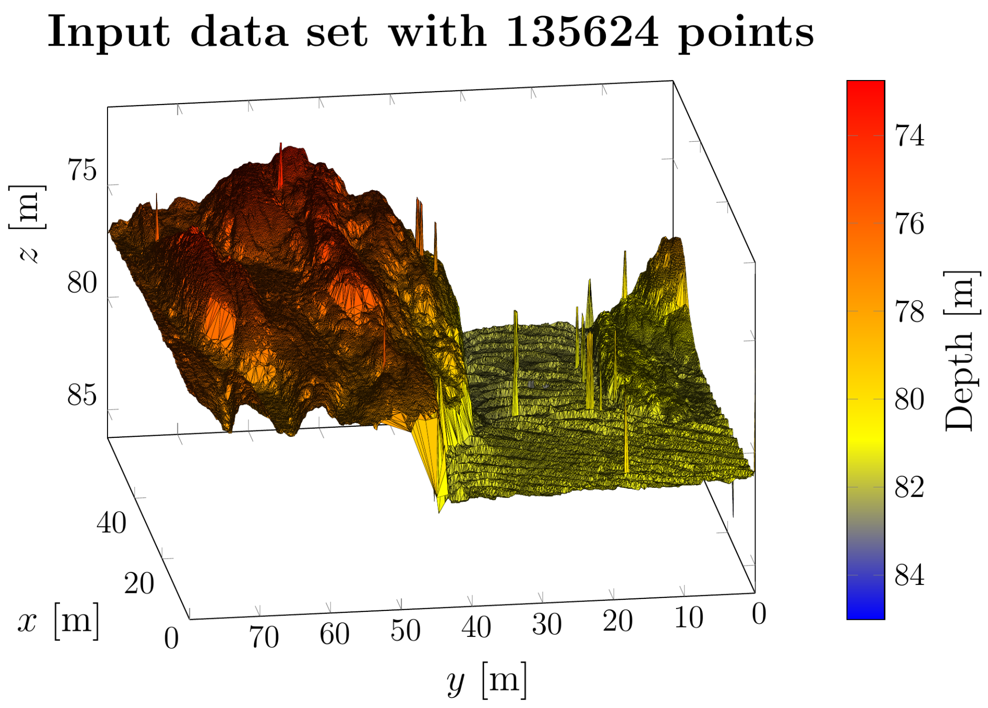
The weighting in the algorithms aims to let the algorithms select points that are more likely to be representative of the seabed. The weighting that we considered was based on the associated coherence map to the Sand Dunes bathymetry. If the algorithms manages to filter out the needles, there would be no need of preprocessing the bathymetry nor manually choosing the coherence threshold that removes all of the needles.

We extracted the points from the Sand Dunes data set that were associated with coherence C equal to or greater than 0.66. We chose to use this threshold at first since coherence of 0.66 means that the estimated SNR from Equation (2.5) becomes approximately equal to two. A SNR of two to a point means that there two times more signal than noise that was used to estimate the elevation assigned to the point.

The thresholded point set contained 135624 points, which is 91.24 percent of the points from the input point set. Needles were still present in the thresholded data set (Figure 4.29). We applied our algorithms on the data set for a given wRMSE threshold to examine if the needles were still present. Based on Table 4.19, we chose to let the refinement algorithm produce point sets with wRMSE just below 0.1. For this wRMSE threshold, the algorithms returned point sets that contained approximately seven, eight, and 16 percent of the points from the Sand Dunes data set. If needles are still present in these sets, the needles will also be present in the larger point sets that satisfied the lower wRMSE thresholds.

Table 4.19: The percentage of points from the input point set in the simplified point sets.

wRMSE threshold	Percentage in size of original point set, wAT1	Percentage in size of original point set, wAT3	Percentage in size of original point set, weighted refinement
$1{\times}10^{-7}$	90.217	91.216	90.704
$1{\times}10^{-6}$	90.147	91.214	90.661
$1{\times}10^{-5}$	89.789	91.177	90.438
$1{\times}10^{-4}$	88.134	90.919	89.251
$1{\times}10^{-3}$	79.937	87.938	82.882
$1{\times}10^{-2}$	41.676	54.333	49.464
$1{\times}10^{-1}$	1.251	4.266	1.69



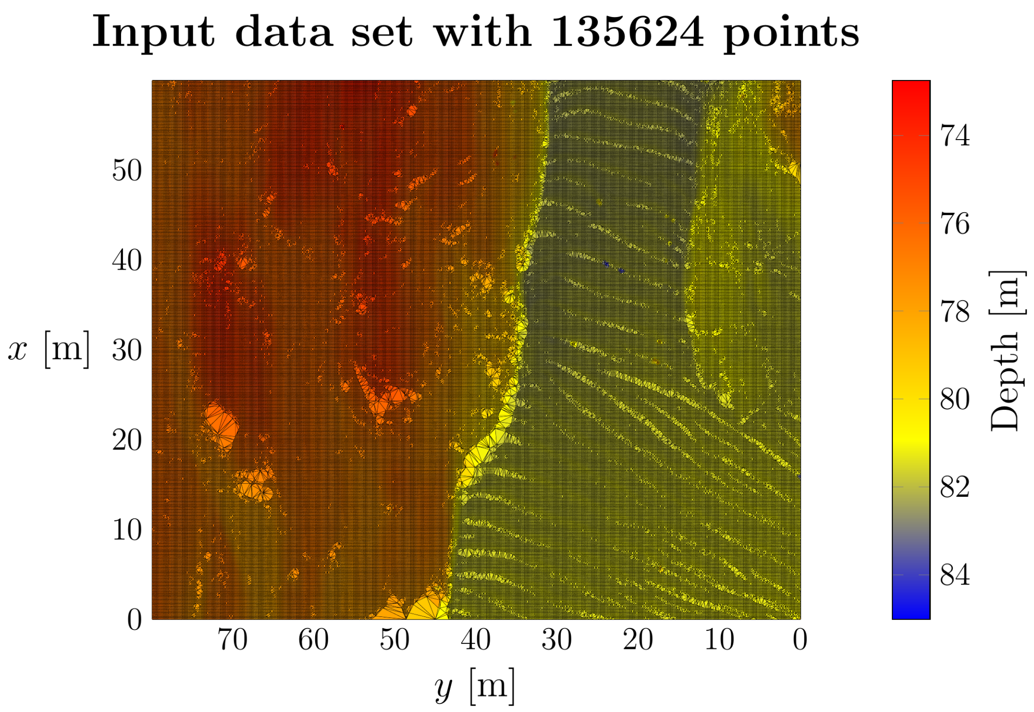


Figure 4.29: The Sand Dunes bathymetry with points associated with 0.66 or more in coherence. The number of points was 135624.

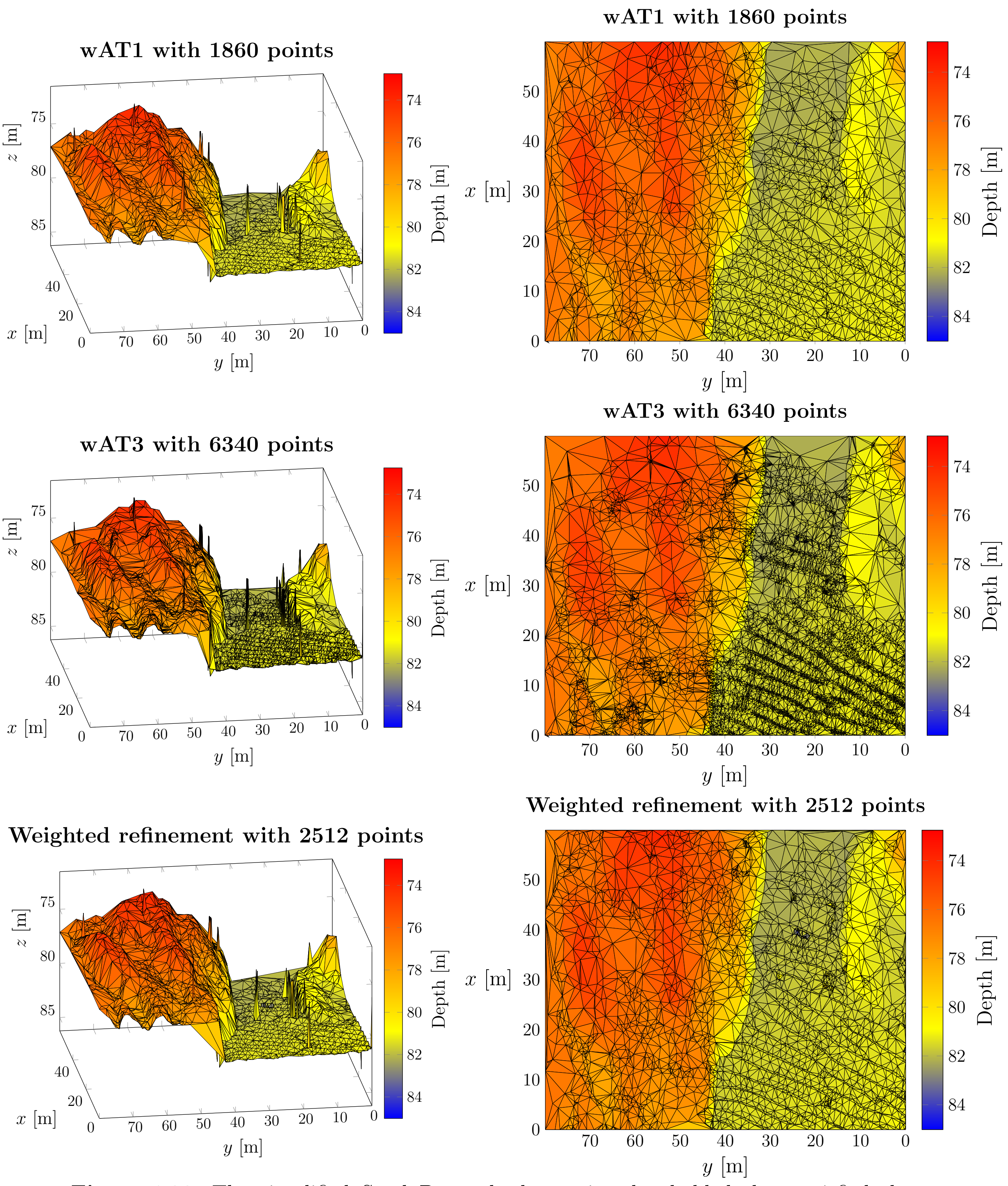


Figure 4.30: The simplified Sand Dunes bathymetries thresholded that satisfied the wRMSE threshold 0.1.

The algorithms preserved the needles in the point sets that satisfied wRMSE threshold of 0.1. However, the point sets were able to also represent the finer details of the Sand Dunes bathymetry such as the sand dunes. The point set from wAT3 contained the most amount of points compared wAT1 and weighted refinement. We experimented with several wRMSE thresholds higher than 0.1, and found that wRMSE threshold of 0.25 made wAT1 and weighted refinement return point sets with no needles and at the same time preserved the characteristics of the seabed, see Figure 4.31.

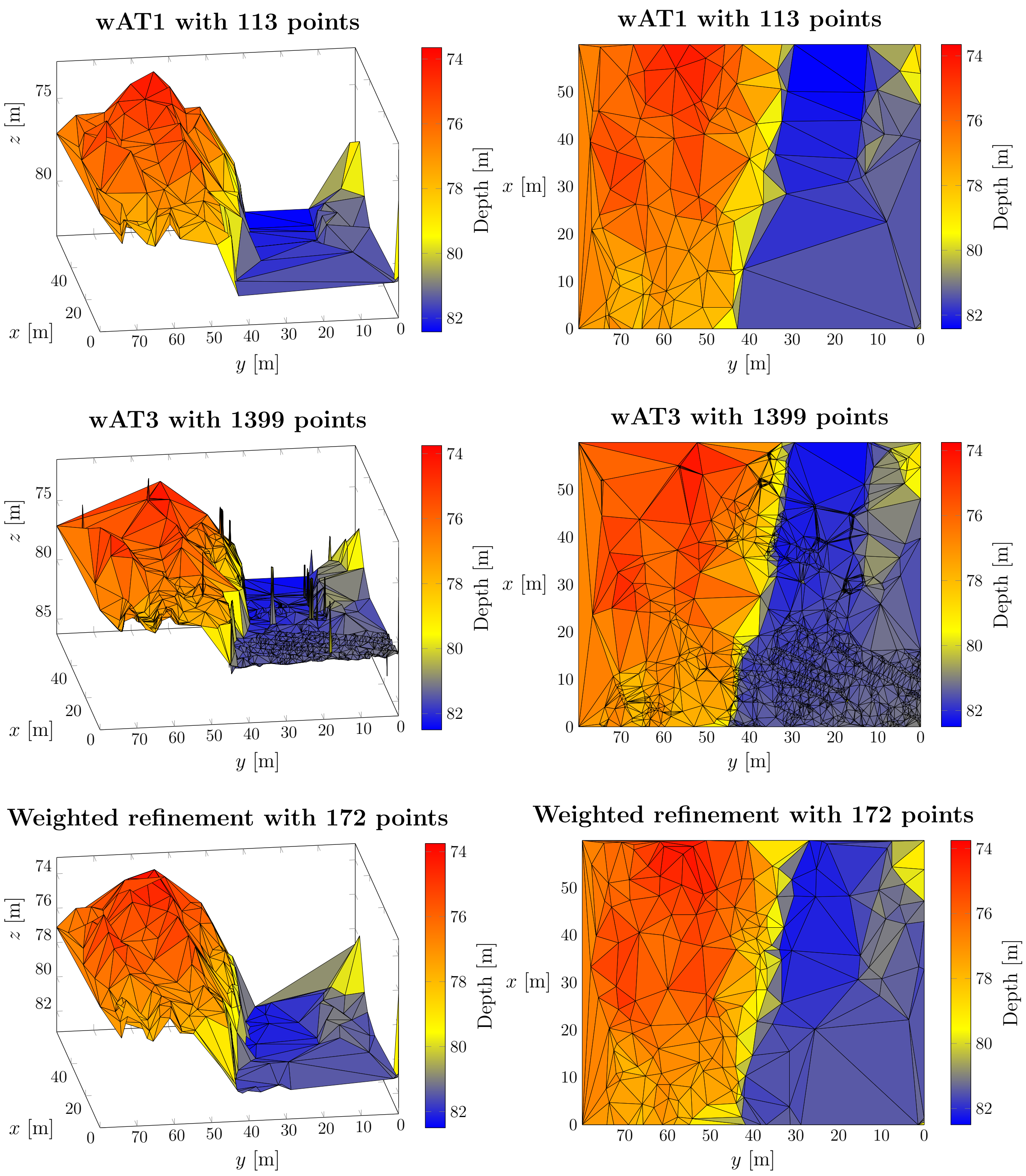


Figure 4.31: The simplified Sand Dunes data sets thresholded with 0.66 in coherence that had wRMSE just below 0.25. The scaling of colors have been limited in the triangulation of the point set obtained from wAT3 for easier visual comparison with the triangulated sets from wAT1 and weighted refinement.

In Figure 4.31 the wAT1 returned a point set of 113 points and the weighted refinement a point set of 172 points. The wAT1 managed to return a point set that contained approximately 0.07 percent of the points from the bathymetry that was not thresholded. The weighted refinement returned a point set of 172 points that was approximately 0.11 percent of points from the bathymetry that was not thresholded. The triangulated point set from weighted refinement had an undesirable triangle between the flat surface and the rock located in the sand dunes region. The triangle was undesirable because it is not representative of the seabed from visual inspection. This might be an effect of refinement that includes the corners point no matter which coherence they are associated with or interpolation error they introduced. The wAT3 did not manage to filter out the needles as much as wAT1 and weighted refinement did.

To demonstrate the importance of the weighting in the algorithms, we applied the algorithms to the thresholded Sand Dunes set assuming that each point had the same non-zero weight. The algorithm will only consider the interpolation errors, as they did in Section 4.3. The resulting triangulated point sets showed that the weighting was important to use in the thresholded Sand Dunes data set (Figure 4.32).

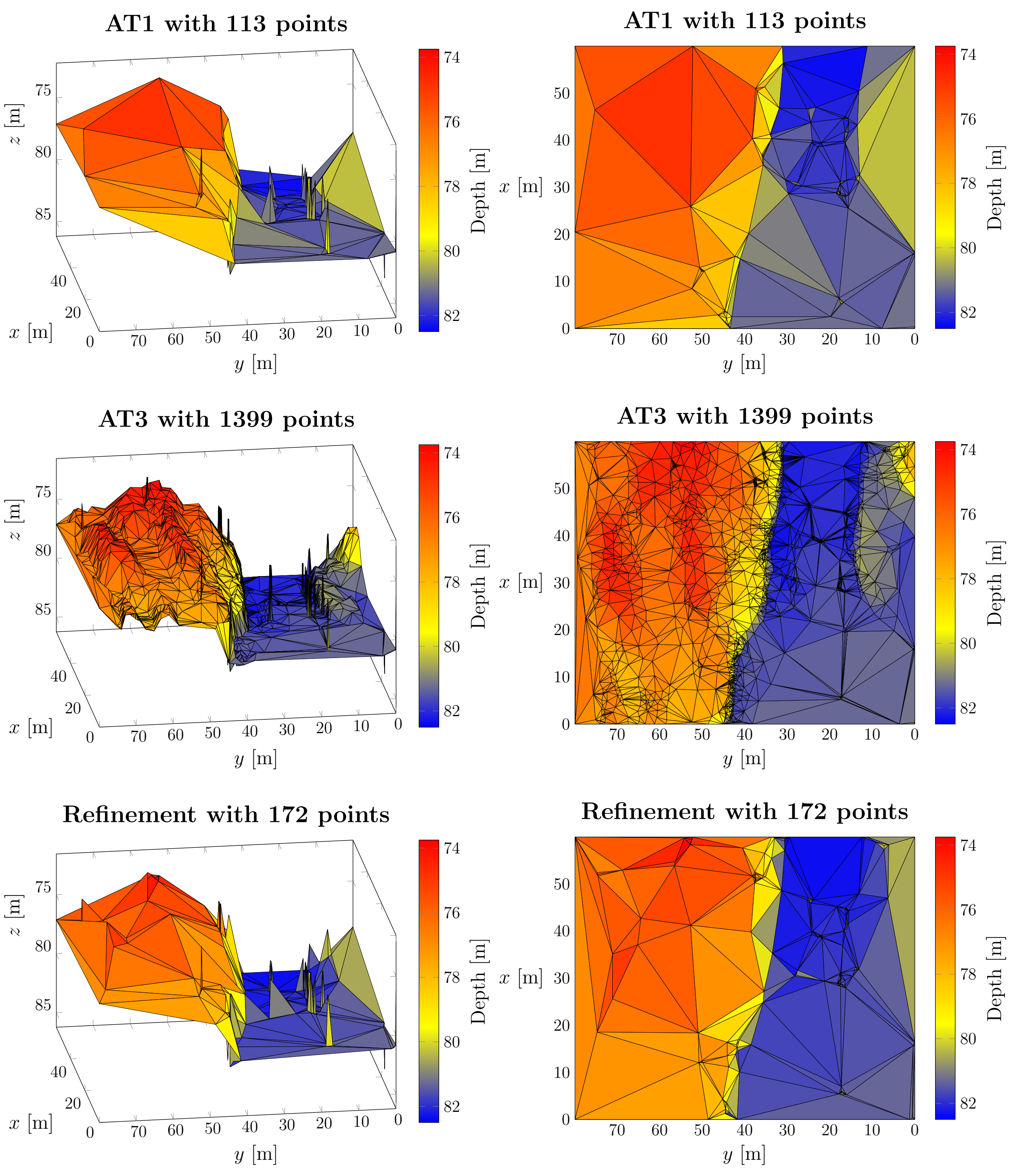


Figure 4.32: The triangulated point sets from the unweighted algorithms. The simplified Sand Dunes data sets thresholded with 0.66 in coherence. The number of points in the simplified points were the same as in Figure 4.31 for each corresponding algorithm.

The unweighted algorithms did not manage to distinguish between representative elevations of the seabed, as they only considered the interpolation error in elevation. The needles were kept and made the simplified surface resemble less the Sand Dunes bathymetry. The triangulated surface from AT3 did resemble the Sand Dunes data the most. However, the needles were still present. The point set from wAT3 contained also more points than the sets from AT1 and refinement.

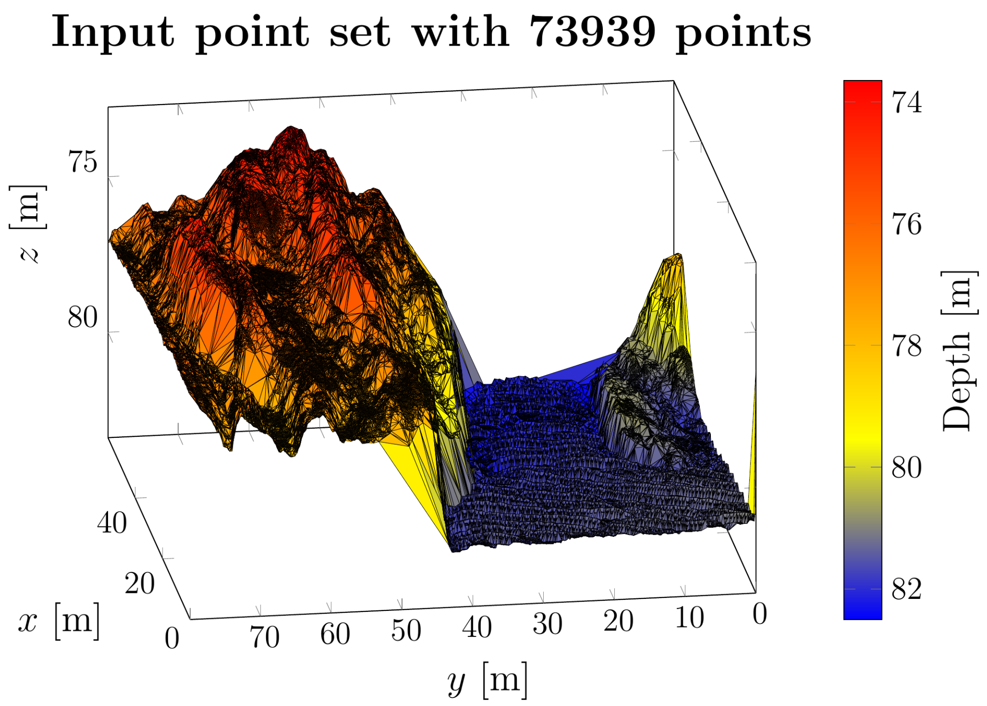
4.5.3 Simplification of Sand Dunes without needles

The wAT1 and weighted refinement had to remove many details of the surface before they produced triangulated surfaces without needles for the Sand Dunes thresholded with 0.66 (Figure 4.31). We found a threshold of 0.91 in coherence that removed most of the needles in our data set (Figure 4.33). The thresholded point set contained 49.75 percent of the points from the input point set. It is of interest to examine how much further the point sets could be reduced in size given a selection of wRMSE thresholds. The wRMSE threshold of 10^{-1} made the simplified point sets contain approximately one percent the amount of points of the input point set, see Table 4.20.

Table 4.20: The percentage of points from the input point set in the simplified point sets.

wRMSE threshold	Percentage in size of original point set, wAT1	Percentage in size of original point set, wAT3	Percentage in size of original point set, weighted refinement
1×10^{-7}	49.318	49.738	49.417
1×10^{-6}	49.269	49.734	49.372
1×10^{-5}	49.052	49.713	49.177
1×10^{-4}	47.988	49.479	48.226
1×10^{-3}	42.277	46.922	43.407
1×10^{-2}	19.121	27.311	22.427
1×10^{-1}	0.912	2.098	1.068

A question of interest, is whether the finer details such as the sand dunes were preserved in the data sets that satisfied wRMSE of 10^{-1}. The point sets from wAT1 and weighted refinement returned point sets that contained approximately one percent of the original Sand Dunes data set and the wAT3 returned a point set that contained approximately three percent of the original data set. The triangulated point sets that satisfied wRMSE of 10^{-1} are shown in Figure 4.34.



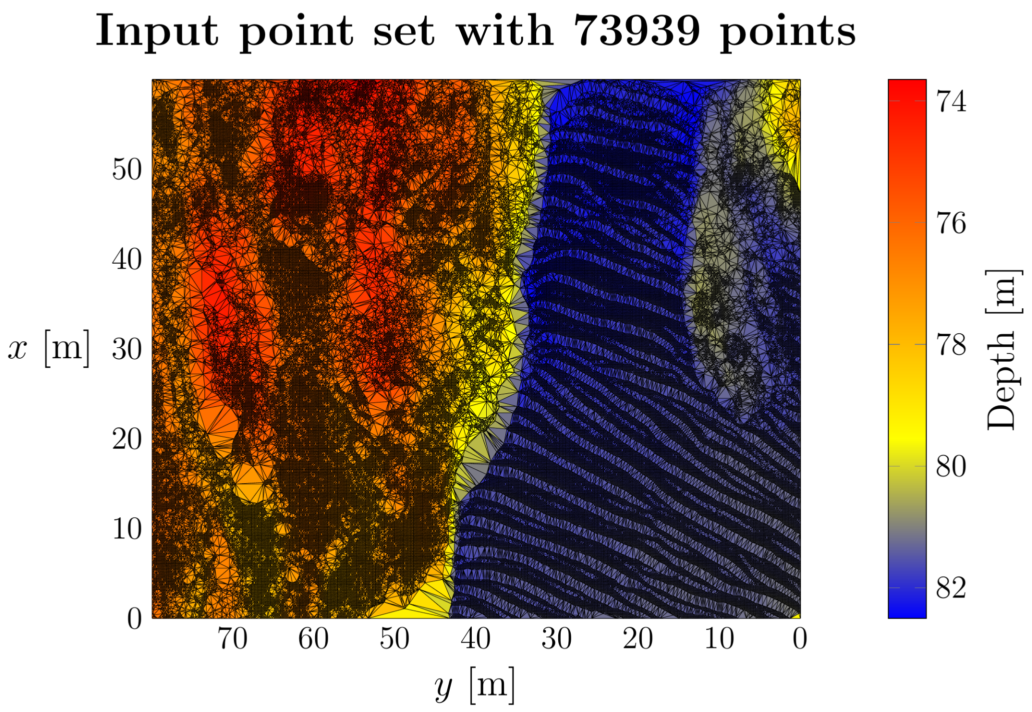


Figure 4.33: The Sand Dunes data set that has points associated with 0.91 or more in coherence.

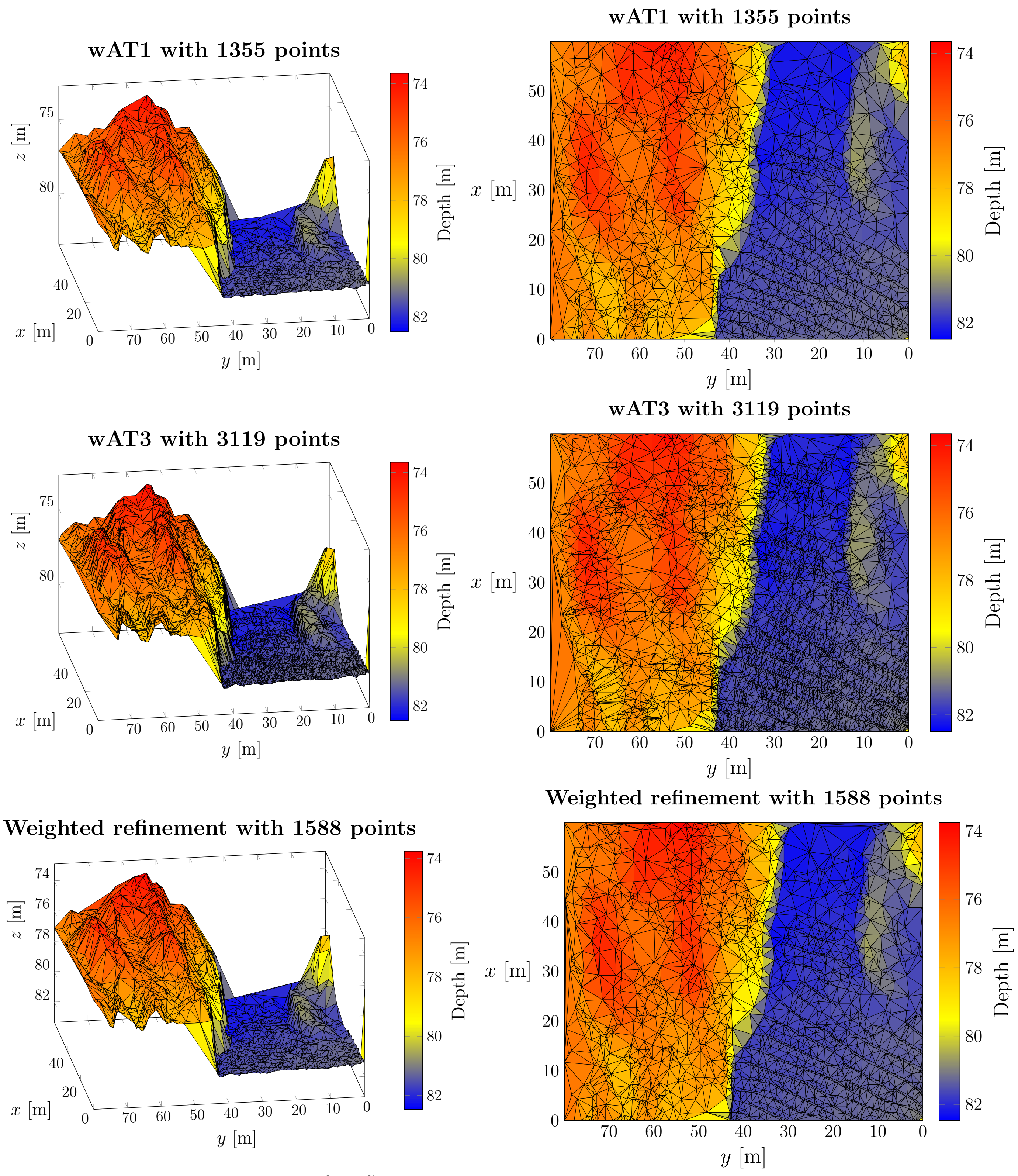
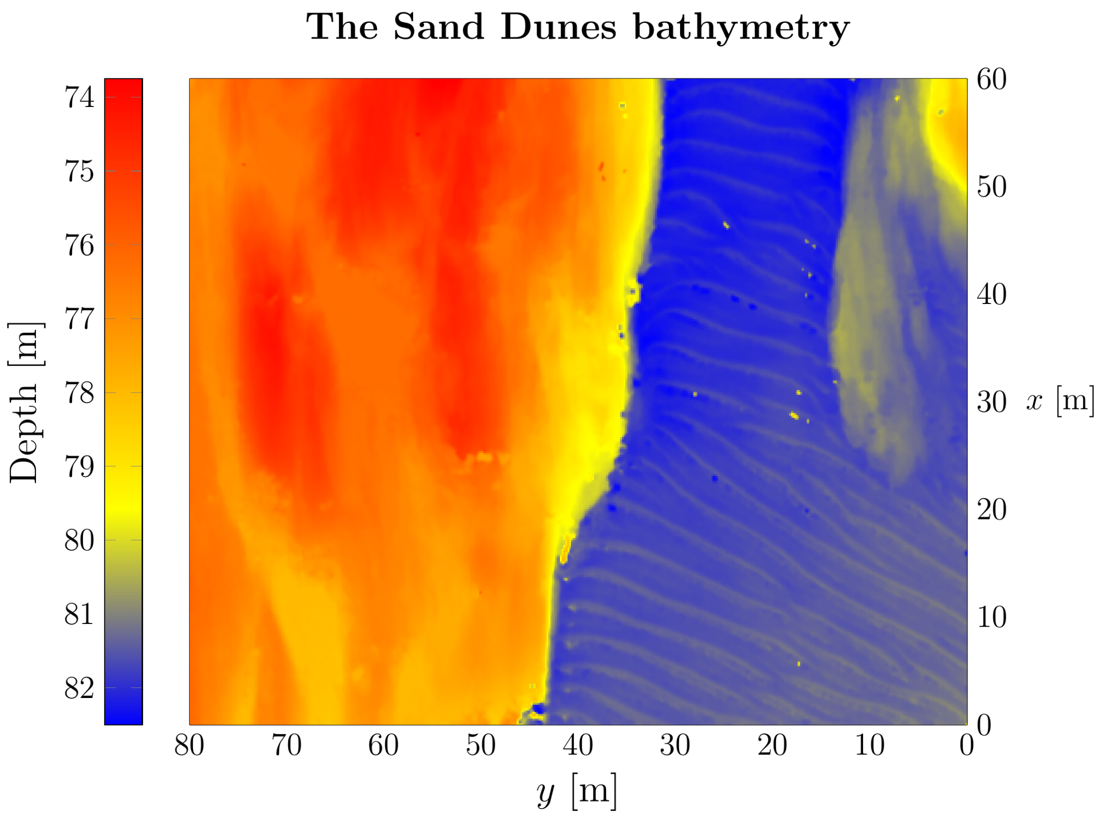


Figure 4.34: The simplified Sand Dunes data sets thresholded with 0.91 in coherence satisfied wRMSE threshold of 0.1.

The wAT1 and weighted refinement were able to represent the sand dunes and at the same time the characteristics of the rocks shown as orange in Figure 4.34. We did not manage to find a wRMSE threshold that also made wAT3 return simplified point sets without needles and at the same let the point sets from wAT1 and weighted refinement contain enough points to represent the input data set. By comparing the thresholded triangulated point set in Figure 4.33 and the point sets from the algorithms in Figure 4.34, we see that most of the two hills are preserved.

It is of interest to examine how well one of the triangulated surfaces from either wAT1 or weighted refinement approximated the thresholded bathymetry. Both of the algorithms returned point sets that contained approximately one percent of the point from the Sand Dunes bathymetry. We considered the surface from the weighted refinement from Figure 4.34. The rasterized TIN from the weighted refinement is shown in Figure 4.35. We computed the absolute difference between the rasterized TIN and the input Sand Dunes bathymetry thresholded with 0.91 (Figure 4.36). The black pixels in the difference images are the points that were excluded after the thresholding. The absolute difference showed that the points deviated mostly by orders of 10^{-1} and 10^{-2} meters. The majority of points that had absolute differences larger than 0.5 meters were located in areas that contained many excluded points.



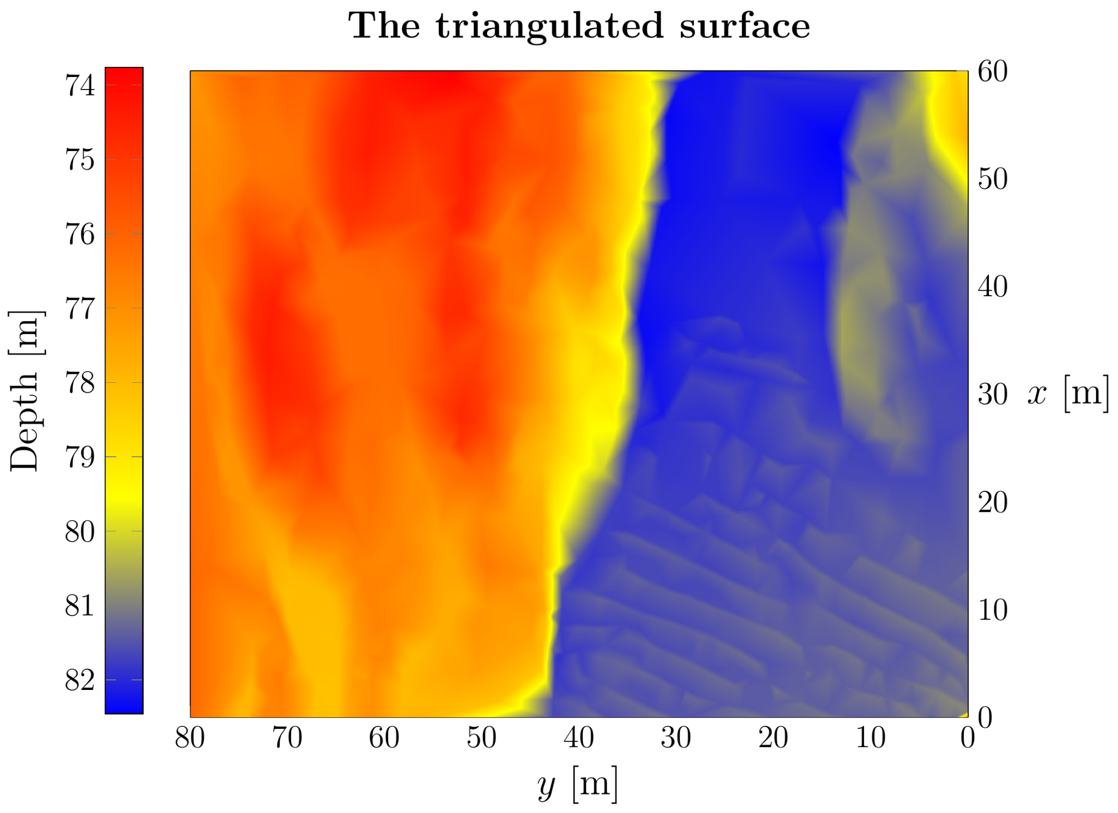
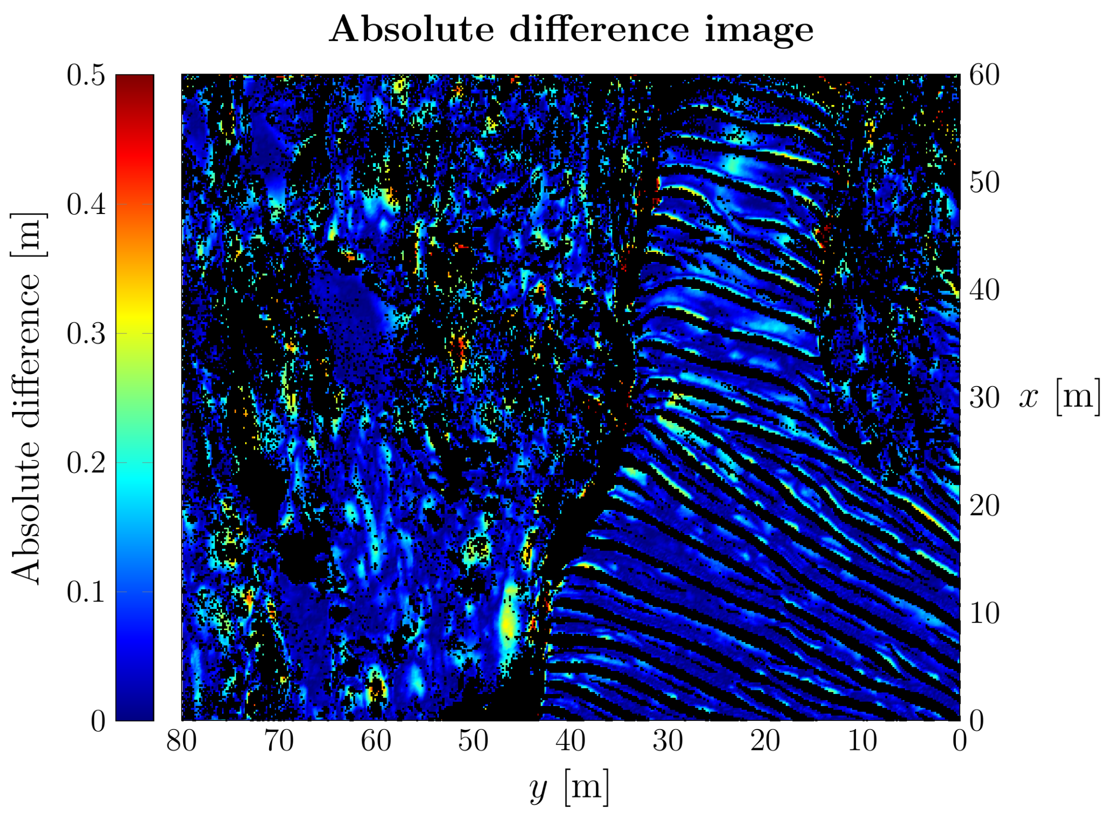


Figure 4.35: *Top:* The median filtered Sand Dunes bathymetry. *Bottom:* The raster of the TIN formed by 1588 points, which is approximately one percent of the points from the input point set.



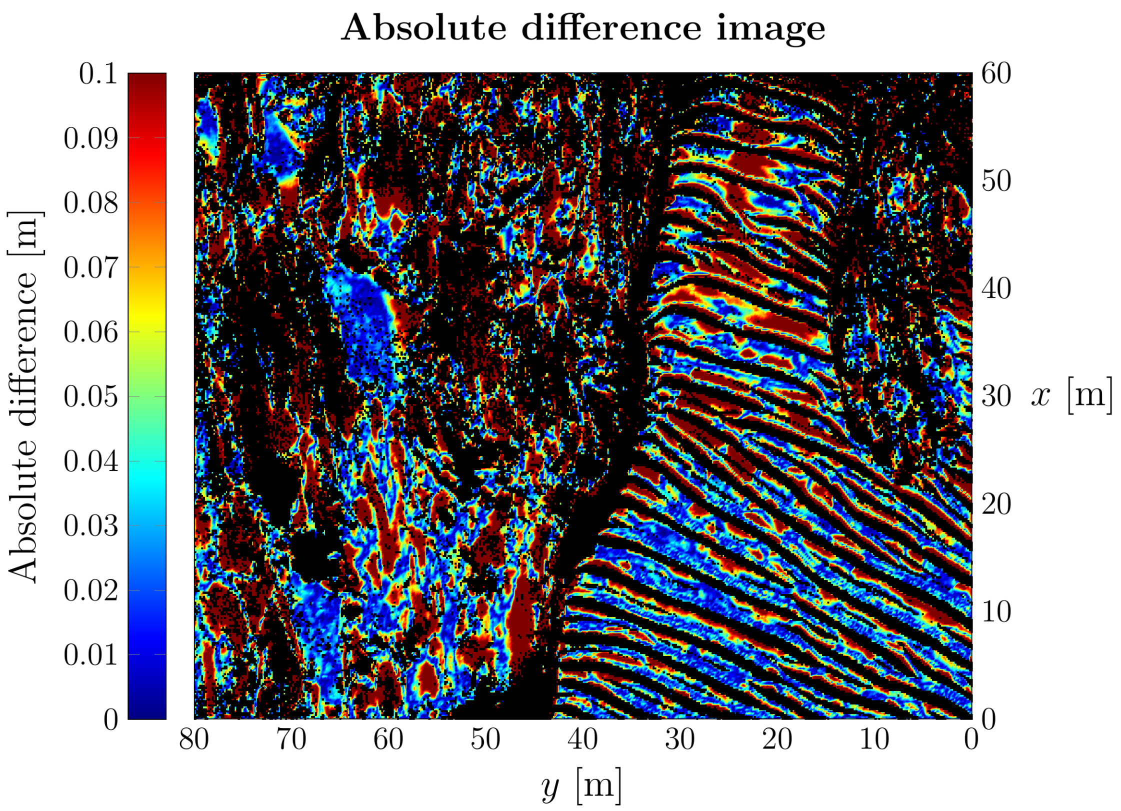


Figure 4.36: Difference images between the interpolated surface from weighted refinement with 1588 points. *Top:* The difference image where we limited the color scale between 0 and 0.5 meters in absolute difference. *Bottom:* The difference image where we limited the color scale between 0 and 0.1 meters in absolute difference.

Many of the points in the sand dunes region and the regions with the rock hills were well represented, with absolute error ot mostly approximately 0.02 (Figure 4.36). Approximately 73 percent of the points in the rasterized TIN deviated by 0.1 meter from the thresholded bathymetry (Table 4.21), even though the rasterized TIN was constructed by approximately one percent of the points from the Sand Dunes bathymetry.

Table 4.21: The percentage of points that deviated less than or equal to m meters in the absolute difference e_{A}.

m	Percentage of points where $e_{\mathrm{A}} \leq m$
0.01	12.78
0.1	73.92
0.5	99.91

5 Discussion

In this chapter, we summarize and discuss our main results from Chapter 4.

5.1 Data without coherence

The algorithms AT1, AT3, and refinement managed to simplify the data sets generated by the two different test functions that we considered.

The refinement algorithm was faster than AT1 as long as the number of points in the simplified point set contained less than approximately 75 percent of an input point set with 40000 points. We think the reason for this is twofold. The AT1 initializes with the whole input point set and removes points iteratively from it. This is inconvenient if the desired point set should be much smaller than its original size. Because of this, AT1 has to perform many more computations than refinement. The refinement algorithm initializes with a coarse model of the input point set and therefore does not need to perform many computations to construct the desired simplified point set. The AT1 performs also a local retriangulation every time its computes the significance for a point. The AT3 does not perform any retriangulation when computing the significance to each point, which makes it significantly faster than AT1.

The AT1 and the refinement algorithm returned point sets with almost the same number of points that satisfied various RMSE thresholds. This is interesting because AT1 is a decimation algorithm that base its simplification on point removals, as opposed to refinement that base its simplification on point insertions. We think both of the algorithms performed almost equally well because they considers explicitly the interpolated elevations at the constructed Delaunay triangulation for each iteration when computing the significances to the points. The AT3 returned more points than AT1 and refinement. We think that the AT3 returned point sets with more points because it considers triangles that is not necessarily a part of the Delaunay triangulation it considers throughout the simplification process. In some sense, AT3 implicitly use a different triangulation structure than Delaunay triangulation by considering the interpolated elevations with respect to the directional triangles to each point it computes the significance of. Since the RMSE is based on Delaunay triangulations of the point sets, the computation of error in AT3 is not based on the triangulated structure the AT3 considers.

In the Ripples test function, the algorithms spent many points on representing the edges to the boxes in R_2 and the cliff that separated the regions R_1 and R_2 and the regions R_3 and R_4. The algorithms were not able to predict or follow the edges in the computation of the significances, which resulted in more points than necessary to represent the edges. This is an effect of the greedy formulation of the algorithms and was also observed in (Garland and Heckbert, 1995). The arc-shaped artifact that is present in the point sets from refinement algorithm is also a consequence of refinement being a greedy algorithm. The arc is present as a consequence on which order the points are inserted along the cliff and when the area that covers R_3 and R_4 is split into two regions.

The triangles in the triangulated point sets from AT3 contained more long and thin triangles than AT1 and refinement for the data sets generated by the Ripples test function. However, the triangulated point sets from AT3 contained long and thin triangles for the data sets generated by the Franke's test function. Most of the triangles had longest normalized edges below 0.4 and smallest angles of all possible degrees. It may be because the Franke's function is slowly varying that made the AT3 avoid long and thin triangles. In addition, it could be of interest to analyze the triangulations by looking at the radius to the circumscribed circles to the triangles. Together with the longest edges and smallest angles to the triangles in a triangulation, one could analyze how the long and thin triangles are formed.

The refinement algorithm with variation sampling converged approximately 31 percent faster than refinement without a set of initial points for the data set containing one million points generated by the Ripples test function. There was no improvement for the refinement algorithm with variation sampling applied to the data set generated by Franke's test function. However, random sampling of the initial points made the refinement algorithm applied on the data set generated by Franke's test function perform better than refinement with variation sampling in terms of wall-clock time. Hence, it seems like the complexity of the surfaces matters in how successful variation sampling is to improve the refinement algorithm. The variation images that we computed for Franke's function are almost the same regardless of neighborhood size whereas the variation images we computed for the Ripples test function are different for every neighborhood size we considered. Indeed, the neighborhoods must be large enough to capture any local variability in the surface. The neighborhood sizes are therefore dependent on how large the local variabilities in a surface are. Since Franke's test function is a slowly varying surface, only the 151×151 large neighborhood captured different variabilities than the other neighborhood sizes we tested for. The Ripples test function is composed by four regions that has different variability and thereby the variation images became different. This could also have been the effect of the second order polynomial fit that made most of the variation images capture similar variations of Franke's function. The polynomial fit might have smoothed the data too much. The motivation behind the second order polynomial fit was to design a method suitable for use data contaminated with noise, which may occur in real data. Because of the dependence of parameters, it was difficult to apply the variation sampling further.

5.2 Data with coherence

The weighted refinement favored points with coherence close to one compared to wAT1 and wAT3 in our studies. In most of the cases we tested for, the weighted refinement algorithm returned point sets where more than 90 percent of the points were associated with 0.9 in coherence. The decimation algorithms applied to the data set generated by the Ripples test function chose points where the interpolation error was high, even though their associated coherence were close to zero. This includes the points at the cliff between the regions R_1 and R_2 and the regions R_3 and R_4 and a small part of the edges to the boxes in R_2. The weighted refinement algorithm avoided including points at the abruptly changing surface, which shows that the algorithm with the weighting we chose seem to favor points with high coherence rather than those that introduces high interpolation errors. We think the refinement chose points with higher coherence because it considers the weighting and interpolation error to one point only for every point the it computes the significance of. This is in contrast to wAT1 and wAT3, which considers the interpolation errors and weights to a set of points for every point they compute to the significance to. The wAT1 and wAT3 seems therefore more sensitive to how the weighting and interpolation error is balanced.

The wAT1 returned point sets with least amount of points compared to weighted refinement and wAT3 in all of our studies. The weighted refinement deviated more from wAT1 in terms of amount of points it returned to satisfy the given wRMSE thresholds than it did for the data sets without coherence. We think the weighted refinement converged slower than wAT1 because it favors points with high coherence and thereby high weights. We think that it is the definition of wRMSE that penalizes more interpolation errors than points with low weights because the weights are divided by their sum in the wRMSE. We could have examined how the weighted algorithms performed with different weighted error metrics.

5.3 Data acquired from an InSAS

The wAT1 and weighted refinement managed to filter out the needles in Sand Dunes bathymetry after disregarding all the points that were associated with coherence less than 0.66. The wAT1 and weighted refinement constructed point sets that satisfied the wRMSE threshold of 0.25. We could not find a wRMSE threshold that made the wAT3 return a simplified point set that resembled the seabed without wAT1 and weighted refinement returned too simplified point sets. The needles are most likely present due to phase-unwrapping errors. Ideally, they should be handled by a phase-unwrapping algorithm. At the same time, the weighting in the simplification algorithms aims to select point that are regarded as more representative of the seabed. In some sense, the algorithms should also be capable to avoid unrepresentative data in their selection process. It seems like the coherence alone is not sufficient to capture the most representative data of the seabed. It was only after thresholding the data set with 0.91 the needles were removed, but at the cost of disregarding more than 50 percent of the input data. It might be so that the algorithms could remove the needles earlier in their simplification process if the weighting was a function of coherence and some other features of the seabed.

We observe that undesirable triangles occurs at the boundary of the input set as a consequence in our implementation to always include the corners points of the input set. However, it is difficult to tell how the boundaries to the simplified surfaces should be handled otherwise.

The weighted refinement returned a point set of 1588 points from the Sand Dunes bathymetery thresholded by 0.91. Approximately 73 percent of the points in the raster TIN based on the point set deviated by 0.1 or less than 0.1 meters from the thresholded bathymetry. We conclude from this that the rasterized TIN was well represented given that the TIN only contained approximately one percent of the points form the input points set.

6

Conclusion

In this thesis we consider three algorithms for raster to TIN conversion. The TINs are based on Delaunay triangulations. The algorithms performs point selection based on the weighted interpolation error to each point in their input point set. The algorithms was applied on three classes of data: synthetic data without coherence, synthetic data with coherence, and a bathymetry from an InSAS.

The algorithms managed to simplify point sets such that they resembled the input surfaces when triangulated, both when the input was associated with coherence and without coherence. The AT1 and refinement returned point sets with almost the same number of points to satisfy a selection of RMSE thresholds. However, the weighted refinement algorithm returned point sets that contained more points than the point sets from wAT1 to satisfy a set of wRMSE thresholds. In all of our test cases, the AT3 and wAT3 returned the most points to satisfy a selection of RMSE and wRMSE thresholds. The results from the variation sampling for constructing sets of initial points to the refinement algorithm was difficult to generalize.The variation sampling is found to be too dependent on its given parameters. Random sampling of initial points performed in overall better than variation sampling. A raster TIN from weighted refinement deviated mostly by 0.1 or less then 0.1 meters in absolute difference. The TIN was formed by approximately one percent of the points from the Sand Dunes bathymetry.

In our opinion, the refinement algorithm, both the weighted and unweighted variant, performed the best of our chosen algorithms. Not only did it converge faster than the decimation algorithms in most of our studies, but it did also return almost the same amount of points than its slower counterparts AT1 and wAT1. Indeed, the AT1 and wAT1 can be faster than refinement, but only for small data reduction. The main application of TINs is to reduce the amount of storage, which is most relevant if the desired output from a simplification algorithm is much smaller than the input point set. The wAT1 and AT1 can therefore be inferior to weighted and unweighted refinement. The weighted refinement managed to return a point set that had 73 percent of its triangulated surface deviated by less than 0.1 meters inclusive, even though the point set contained approximately one percent of the input point set. This showed that the weighted refinement have the potential to reduce point sets with relatively short amount of time and at the same time provide simplified surfaces that represents well the input surface.

6.1 Future work

Based on the results we obtained from our studies, we have several proposals to future research that could be of interest. These includes:

- **Efficient implementations**
  We decided to not focus on effective implementations of our algorithms in this thesis. We note that (Garland and Heckbert, 1995, p. 24) wrote a refinement algorithm in C++ that managed to yield a point set of 50000 points from a point set that contained 1024×1024 points in 46 seconds. Our refinement algorithm written in MATLAB can construct a point set of approximately 35000 points from a point set containing 40000 points in approximately 70 seconds. In (Garland and Heckbert, 1995), they show that the usage of data structures can improve the efficiency to the algorithms remarkably. The decimation algorithms, especially AT1 and wAT1, could also be improved in terms of efficiency by using more efficient data structures. Our implementations could be faster if they were written in C++.

- **Performance with respect to the max deviation**
  For some applications it might be useful to control the max deviations in the simplified surfaces such as in terrain-based navigation for AUVs. It could be of interest to examine how the algorithms performed with respect to the max deviation also.

- **Simplification of large bathymetries**
  A map that spans over a larger region that we considered in this thesis can contain millions of points. If the map is supposed to be used for terrain-based navigation in AUVs, the maps might contain too much data for an AUV to carry. The bathymetric maps can overlap and might also been acquired from different mapping systems. To construct a simplified map over a large area could also be achieved by simplifying the overlapping data and merge them. A study on how the algorithms performs on large bathymetric maps or many bathymetric maps that overlaps could therefore be of interest.

- **Automated analysis of the triangulated surfaces**
  In our studies, we mostly performed visual inspections of our simplified surfaces to see whether they resembled the given surface. This is undesirable if many terrains are to be simplified. The estimation of terrain features such as slope, aspect, and curvature could provide more insight into the simplified surfaces and has the potential to provide a more automated analysis of the simplified surfaces.

- **Refinement for detection of control points in co-registration**
  Co-registration is the process of mapping a pair of images over the same scene, but observed from different locations, such that their pixels represents the same geographical locations in a scene. To perform co-registration of two images, a set of control points has to be found. The refinement algorithm, both the unweighted and weighted variant of it, can possibly be used to find the control

points that are applied in a co-registration method. The refinement algorithm showed to have the potential to find few and essential points to describe a surface.

- **Distortions of the surface**
  In practice, the imaged seabed can be distorted in the imaging process. Our algorithms could be tested on how they perform on surfaces that are slightly distorted. This could be important to know in applications such as merging of the surfaces and finding points that represent important features of the surface.

- **Additional weighting in the decimation algorithms**
  As the weighted decimation algorithms uses a set of values to determine the significance to each point, they might be sensitive to how the interpolation error and weights are combined together. An additional weighting could perhaps provide more control on how the significances are chosen from the weighted interpolation errors. For instance, one could introduce an additional weight in wAT1 that weights how much the values within the cell $\mathcal{C}(p)$ should contribute to the computed significance of a point p. The additional weighting could for example be a function of how far away the points within the cell are from p.

- **Feature extraction method with less parameters**
  Our variation sampling relied heavily on three parameters that defines the size of the neighborhood to the variation images, the maximum number of points within a rectangle in the quadtree split, and the maximum variation allowed within a rectangle in the quadtree split. Our studies showed that they had a major effect on how the set of initial points made the refinement algorithm perform. A less parameter controlled feature extraction could make the refinement algorithm converge better for surfaces in general, as it is possible to let the refinement perform much better if the optimal set of initial points was found. From our studies, we were also uncertain whether our chosen variation estimate did describe the surface well enough. Geomorphologists have throughout the years suggested may features that could be feasible to describe a surface, see e.g (Wilson and Gallant, 2000), (Burrough et al., 2015, Chap. 10), and (Wilson, 2018, Chap. 3), that we think could be considered to use in a feature-based point extraction.

- **Hybrid of weighted refinement and weighted decimation**
  In (Pedrini, 2000, pp. 44 - 48) they present a hybrid of a decimation and a refinement algorithm. It could be interesting to see how the weighted variants of the considered decimation algorithms and the weighted refinement could be combined together.

- **Other definitions of weighting**
  Our weights were inspired by how the coherence and SNR is related, see Equation (2.6). The coherence could be used in different ways, and it could be of interest to see how the different usages of the coherence affects the performance to the weighted simplification algorithms. The coherence alone might not capture all of the unrepresentative data of the seabed. It could be of interest to

examine how the coherence combined with some other quality measurements affected the point selection to the weighted simplification algorithms.

Bibliography

Akenine-Möller, T., Haines, E., Hoffman, N., Pasce, A., Iwanicki, M., and Hillaire, S. (2018). *Real-time rendering*. CRC Press, Fourth edition.

Alliez, P., Cohen-Steiner, D., and Zhu, L. (2019). New in CGAL: Triangulated surface mesh approximation. Announcement published at the official website to CGAL at 29th of January 2019. Available at https://www.cgal.org/2019/01/29/VSA/. Last accessed: 20th of July 2019.

de Berg, M., Van Kreveld, M., Overmars, M., and Schwarzkopf, O. (2008). *Computational geometry*. Springer, Third edition.

Blondel, P. (2009). *The handbook of sidescan sonar*. Springer.

Brown, J. L. (1991). Vertex based data dependent triangulations. *Computer Aided Geometric Design*, 8(3):239–251.

Bruce, M. P. (1992). A processing requirement and resolution capability comparison of side-scan and synthetic-aperture sonars. *IEEE Journal of Oceanic Engineering*, 17(1):106–117.

Burrough, P. A., McDonnell, R., McDonnell, R. A., and Lloyd, C. D. (2015). *Principles of geographical information systems*. Oxford university press, Third edition.

Canepa, G., Bergem, O., and Pace, N. G. (2003). A new algorithm for automatic processing of bathymetric data. *IEEE Journal of Oceanic Engineering*, 28(1):62–77.

Cheng, S.-W., Dey, T. K., and Shewchuk, J. (2012). *Delaunay mesh generation*. Chapman and Hall/CRC.

Demaret, L., Dyn, N., Floater, M. S., and Iske, A. (2005). Adaptive thinning for terrain modelling and image compression. In Dodgson, N. A., Floater, M. S., and Sabin, M. A., editors, *Advances in Multiresolution for Geometric Modelling*, pages 319–338. Springer.

Dokken, T., Skytt, V., and Barrowclough, O. (2015). Locally refined splines representation for geospatial big data. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 40.

Dyn, N., Levin, D., and Rippa, S. (1990). Data dependent triangulations for piecewise linear interpolation. *IMA journal of numerical analysis*, 10(1):137–154.

Farin, G. (1997). *Curves and surfaces for computer-aided geometric design*. Academic Press, Fourth edition.

de Floriani, L. and Magillo, P. (2002). Multiresolution mesh representation: Models and data structures. In Iske, A., Quak, E., and Floater, M. S., editors, *Tutorials on Multiresolution in Geometric Modelling*, pages 363–417. Springer.

Fortune, S. (2017). Voronoi diagrams and delaunay triangulations. In Goodman, J. E., O'Rourke, J., and Toth, C. D., editors, *Handbook of discrete and computational geometry*, chapter 27, pages 705 – 721. CRC Press, Third edition.

Fowler, R. J. and Little, J. J. (1979). Automatic extraction of irregular network digital terrain models. *Computer Graphics (SIGGRAPH '97 Proc)*, 13(2):199–207.

Franceschetti, G. and Lanari, R. (1999a). Synthetic aperture radar interferometry (written by Franceschetti, G. and Fornaro, G.). In *Synthetic aperture radar processing*, chapter 4, pages 167 – 378. CRC press.

Franceschetti, G. and Lanari, R. (1999b). *Synthetic aperture radar processing*. CRC press.

Franke, R. (1979). A critical comparison of some methods for interpolation of scattered data. Technical report, Naval Postgraduate School Monterey CA.

Garland, M. and Heckbert, P. S. (1995). Fast polygonal approximation of terrains and height fields. Technical report, Carnegie Mellon University Pittsburgh, School of Computer Science.

Ghiglia, D. C. and Pritt, M. D. (1998). *Two-dimensional phase unwrapping*. Wiley-Interscience.

Gonzalez, R. C. and Woods, R. E. (2010). *Digital image processing*. Pearson, Third edition. International edition.

Gough, P. T. and Hawkins, D. W. (1997). Unified framework for modern synthetic aperture imaging algorithms. *International journal of imaging systems and technology*, 8:343–358.

Gough, P. T. and Hawkins, D. W. (1998). A short history of synthetic aperture sonar. In *Sensing and Managing the Environment. 1998 IEEE International Geoscience and Remote Sensing. Symposium Proceedings.*, volume 2, pages 618–620. IEEE.

Griffiths, H. D. (1997). A comparison between radar and sonar synthetic aperture interferometry. In *IEE Colloquium on Radar Interferometry (Digest No: 1997/153)*, pages 1–5.

Hansen, R. E. (2011). Introduction to synthetic aperture sonar. In Kolev, N., editor, *Sonar systems*, chapter 1, pages 3 – 28. IntechOpen. Available from https://www.intechopen.com/books/sonar-systems/introduction-to-synthetic-aperture-sonar. Last accessed: 30th of July 2019.

Hansen, R. E., Callow, H. J., Sæbø, T. O., and Synnes, S. A. V. (2011). Challenges in seafloor imaging and mapping with synthetic aperture sonar. *IEEE Transactions on geoscience and Remote Sensing*, 49(10):3677–3687.

Hansen, R. E., Lågstad, P., and Sæbø, T. O. (2019). Search and monitoring of shipwreck and munitions dumpsites using HUGIN AUV with synthetic aperture sonar. Technical report available on https://publications.ffi.no/nb/item/search-and-monitoring-of-shipwreck-and-munitions-dumpsites-using-hugin-auv-with-synthetic-aperture-sonar-technology-study. Last retrieved: 17th of June 2019.

Hansen, R. E., Sæbø, T. O., Synnes, S. A. V., and Lorentzen, O. J. (2018). Challenges in coregistration of repeated passes in synthetic aperture sonar. In *EUSAR 2018; 12th European Conference on Synthetic Aperture Radar*, pages 1–6.

Hanssen, R. F. (2001). Radar interferometry: Data interpretation and error analysis. volume 2 of *Remote Sensing and Digital Image Processing* (Meer, F. van de, Series Editor). Kluwer Academic Publishers.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: Data mining, interference and Prediction.* Springer, Second edition. Corrected 12th printing of the book fron January 2017. The book is available from https://web.stanford.edu/~hastie/ElemStatLearn/. Last retrieved: 20th of July 2019.

Haverkort, H. and Toma, L. (2014). Terrain modeling for the geosciences. In Gonzalez, T., Diaz-Herrera, J., and Tucker, A., editors, *Computing handbook: Computer science and software engineering*, chapter 31, pages 1 – 21. CRC Press, Third edition.

Hayes, M. P. and Gough, P. T. (2009). Synthetic aperture sonar: a review of current status. *IEEE Journal of Oceanic Engineering*, 34(3):207–224.

Heckbert, P. S. and Garland, M. (1997). Survey of polygonal surface simplification algorithms. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science.

Hengl, T. and Evans, I. S. (2009). Mathematical and digital models of the land surface. In Hengl, T. and Reuter, H. I., editors, *Geomorphometry: Concepts, Software, Applications*, volume 33 of *Developments in soil science* (Hartemink, A. E. and McBratney, A. B., Series Editor), chapter 2, pages 31 – 63. Elsevier.

Heywood, I., Cornelius, S., and Carver, S. (2011). *An introduction to geographical information systems.* Pearson Education, Fourth edition.

Hjelle, Ø. and Dæhlen, M. (2006). *Triangulations and applications*. Springer Science & Business Media.

Horn, B. K. P. (1981). Hill shading and the reflectance map. *Proceedings of the IEEE*, 69(1):14–47.

Horn, R. A. and Johnson, C. R. (2013). *Matrix analysis*. Cambridge university press, Second edition.

Jakowatz, C. V., Wahl, D. E., Eichel, P. H., Ghiglia, D. C., and Thompson, P. A. (1999). *Spotlight-Mode Synthetic Aperture Radar*. Kluwer Academics Publishers.

Johnson, D. H. and Dudgeon, D. E. (1993). Array signal processing: concepts and techniques. Prentice hall signal processing series *(Oppenheim, A. V., Series Editor)*. PTR Prentice Hall.

Kay, S. M. (1993). *Fundamentals of statistical signal processing: Estimation theory*, volume 1. Prentice Hall.

van Kreveld, M. (1997). Digital elevation models and TIN algorithms. In van Kreveld, M., Nievergelt, J., Roos, T., and Widmayer, P., editors, *Algorithmic Foundations of Geographic Information Systems*, pages 37–78. Springer.

van Kreveld, M. (2017). Geographic information systems. In Goodman, J. E., O'Rourke, J., and Toth, C. D., editors, *Handbook of discrete and computational geometry*, chapter 59, pages 1555 – 1580. CRC Press, Third edition.

Kreyszig, E. (2011). *Advanced engineering mathematics*. John Wiley & Sons, Inc., Tenth edition.

Kuperman, W. A. and Roux, P. (2014). Underwater acoustics. In Rossing, T. D., editor, *Springer handbook of acoustics*, chapter 5, pages 157 – 212. Springer, Second edition.

Lai, M.-J. and Schumaker, L. L. (2007). *Spline functions on triangulations*. Cambridge University Press.

Lawson, C. L. (1977). Software for C1 surface interpolation. In Rice, J. R., editor, *Mathematical Software III*. Academic press. Available from: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19770025881.pdf. Last retrieved: 29th of July 2019.

Lee, J. (1991). Comparison of existing methods for building triangular irregular network, models of terrain from grid digital elevation models. *International Journal of Geographical Information System*, 5(3):267–285.

Leonard, J. J. and Bahr, A. (2016). Autonomous underwater vehicle navigation. In Dhanak, M. R. and Xiros, N. I., editors, *Springer Handbook of Ocean Engineering*, pages 341–358. Springer.

Li, W., Chen, Y., Wang, Z., Zhao, W., and Chen, L. (2014). An improved decimation of triangle meshes based on curvature. In Miao, D., W., P., Ślęzak, D., Peters, G., Hu, Q., and Wang, R., editors, *International Conference on Rough Sets and Knowledge Technology*, pages 260–271. Springer International Publishing.

Li, Z., Zhu, Q., and Gold, C. (2004). *Digital terrain modeling: principles and methodology.* CRC press.

Lurton, X. (2010). *An introduction to underwater acoustics: principles and applications.* Springer, Second edition.

Maleika, W., Koziarski, M., and Forczmański, P. (2018). A multiresolution grid structure applied to seafloor shape modeling. *ISPRS International Journal of Geo-Information*, 7(3). Available from https://www.mdpi.com/2220-9964/7/3/119/htm. Last retrieved: 20th of July 2019.

Manolakis, D. G. and Ingle, V. K. (2011). *Applied digital signal processing.* Cambridge University Press.

Massonnet, D. and Souyris, J.-C. (2008). *Imaging with synthetic aperture radar.* EPFL press, distributed by CRC Press.

Matthews, P. C. (1998). *Vector calculus.* Springer-Verlag London.

Mayer, L., Jakobsson, M., Allen, G., Dorschel, B., Falconer, R., Ferrini, V., Lamarche, G., Snaith, H., and Weatherall, P. (2018). The Nippon Foundation—GEBCO Seabed 2030 Project: The quest to see the world's oceans completely mapped by 2030. *Geosciences*, 8(2). Available from https://www.mdpi.com/2076-3263/8/2/63. Last retrieved: 20th of July 2019.

Midtbø, T. (1994). Removing points from a delaunay triangulation. In *Proceedings from the sixth International Symposium on Spatial Data Handling*, pages 739–750.

Nochetto, R. H. and Veeser, A. (2011). Primer of adaptive finite element methods. In Naldi, G. and Russo, G., editors, *Multiscale and adaptivity: modeling, numerics and applications*, pages 125–225. Springer. C.I.M.E Summer School, Cetraro, Italy, 2009.

Okabe, A., Boots, B., Sugihara, K., and Chiu, S. N. (2000). *Spatial tessellations: concepts and applications of Voronoi diagrams.* John Wiley & Sons, Second edition.

Pedrini, H. (2000). An improved refinement and decimation method for adaptive terrain surface approximation. Ph.D dissertation at Rensselaer Polytechnic Institute, Troy, New York, United States. Available at https://acervodigital.ufpr.br/handle/1884/107. Last retrieved: 19th of July 2019.

Pfeifer, N. and Mandlburger, G. (2018). Lidar data filtering and digital terrain model generation. In Shan, J. and Toth, C. K., editors, *Topographic laser ranging and scanning: principles and processing*, chapter 11, pages 350–378. CRC press, Second edition.

Preparata, F. P. and Shamos, M. I. (1985). *Computational geometry: an introduction.* Springer Science & Business Media.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical recipes.* Cambridge university press, Third edition.

Quarteroni, A. (2009). *Numerical models for differential problems*, volume 2. Springer. English translation by Quarteroni, S. from the original Italian edition by Quarteroni, A. *Modellistica Numerica per Problemi Differenziali*, fourth edition.

Reuter, H. I., Hengl, T., Gessler, P., and Soille, P. (2009). Preparation of DEMs for geomorphometric analysis. In Hengl, T. and Reuter, H. I., editors, *Geomorphometry: Concepts, Software, Applications*, volume 33 of *Developments in soil science* (Hartemink, A. E. and McBratney, A. B., Series Editor), chapter 4, pages 87 – 120. Elsevier.

Richards, M. A. (2005). *Fundamentals of radar signal processing.* The McGraw-Hill Companies.

Richards, M. A. (2006). A beginner's guide to interferometric SAR concepts and signal processing. *IEEE Aerospace and Electronic Systems Magazine*, 21(6):5–29.

Rippa, S. (1992). Long and thin triangles can be good for linear interpolation. *SIAM Journal on Numerical Analysis*, 29(1):257–270.

Rossing, T. D. (2014). Introduction to acoustics. In Rossing, T. D., editor, *Springer handbook of acoustics*, chapter 1, pages 1 – 7. Springer, Second edition.

Sæbø, T. O. (2010). Seafloor depth estimation by means of interferometric synthetic aperture sonar. Ph.D dissertation at Physics and Technology Department, University of Tromsø, Norway.

Sæbø, T. O. and Hansen, R. E. (2010). Comparison between interferometric SAS and interferometric SAR. In *Proceedings of Synthetic Aperture Sonar and Radar 2010.*

Sæbø, T. O., Synnes, S. A. V., and Hansen, R. E. (2013). Wideband interferometry in synthetic aperture sonar. *IEEE Transactions on Geoscience and Remote Sensing*, 51(8):4450–4459.

Shewchuk, J. R. (2002). What is a good linear element? interpolation, conditioning, and quality measures. In *Proceedings of the 11th International Meshing Roundtable,*, pages 115 – 126. Sandia National Laboratories.

Sibson, R. (1981). A brief description of natural neighbour interpolation. In Barnett, V., editor, *Interpreting multivariate data.* John Wiley & Sons.

Soommart, S. and Paitoonwattanakij, K. (1999). Incremental delaunay triangulation algorithm for digital terrain modelling. *Thammasat Int. J. Sc. Tech*, 4(2):64–75.

Strutz, T. (2011). *Data fitting and uncertainty: A practical introduction to weighted least squares and beyond*. Vieweg+Teubner.

Sun, W., Wang, H., and Zhao, X. (2018). A simplification method for grid-based DEM using topological hierarchies. *Survey Review*, 50(362):454–467.

Tobler, W. R. (1970). A computer movie simulating urban growth in the Detroit region. *Economic geography*, 46(sup1):234–240.

Wilson, J. P. (2018). *Environmental Applications of Digital Terrain Modeling*. John Wiley & Sons.

Wilson, J. P. and Gallant, J. C., editors (2000). *Terrain analysis: principles and applications*. John Wiley & Sons.

Zebker, H. A. and Villasenor, J. (1992). Decorrelation in interferometric radar echoes. *IEEE Transactions on geoscience and remote sensing*, 30(5):950–959.

Zhou, Q. and Chen, Y. (2011). Generalization of DEM for terrain analysis using a compound method. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(1):38–45.

A
Implementation details

We used MATLAB to implement our algorithms. The simplification algorithms developed for this thesis were made without using any pre-developed code except for functions available in MATLAB. Functions that were often used includes `delaunayTriangulation`, `pointLocation`, and `inpolygon`. The implementations were written on various versions of MATLAB. We used MATLAB R2017a and MATLAB 2019a interchangeably throughout the development.

We found some instabilities in the implementations when Mathworks' `pointLocation` function found wrong triangles to some points. The points were located at the boundary of the convex hull of the input point set. This was found by checking whether the computed barycentric coordinates to each point was greater than one plus some small threshold to compensate for numerical errors. After moving the points at the boundary by 10^{-13} inside the convex hull of the input point set and passing the displaced points to `pointLocation`, the found triangles were correct. This circumvention was inspired by (Press et al., 2007, pp. 1135 - 1136), where adding a small displacement factor can be used in a point location algorithm.

We found that retrieving values from objects took was inefficient compared to initializing and keeping variables in terms of wall-clock time. This was the bottleneck for our implementations. The implementations relies heavily on extracting the correct height values from the `delaunayTriangulation` object to use for interpolation when the significances are computed. The only circumvention we saw, was to implement our own retriangulation algorithm and incorporate it into how we structured the data in our implementation. This would have been out of scope for this thesis.

The computation of the variation images uses `parfor` to perform the second order polynomial fit and compute the variation to the fitted elevations. This is only possible if the Parallel Toolbox from Mathworks is available. The quadtree resampling is implemented as a recursive function, which we found was an effective way of implementing the splitting.