SONDERPUBLIKATION Nr. 003

Interactive processing of MBES bathymetry and backscatter data using Jupyter Notebook and Python

Sophie Andree

DHyG-Sonderpublikation
Nr. 003
Juni 2021

Herausgeber:
Deutsche Hydrographische Gesellschaft e. V. (DHyG)

DOI: 10.23784/DHyG-SP_003

DHyG-SONDERPUBLIKATION

Nr. 003

Interactive processing of MBES bathymetry and backscatter data using Jupyter Notebook and Python

Sophie Andree

Abstract

Nowadays multibeam echo sounders are the most common and efficient method to perform hydrographic surveys. Processing hydrographic data (bathymetry and backscatter) requires a set of procedures to obtain reliable information. Several commercial and scientific software solutions are already available for this purpose, but accessibility is limited. This will be addressed by the use of open source libraries. The combination of Jupyter Notebook and Python has great potential for interactive data processing. A workflow for bathymetry and backscatter processing was developed based on Kongsberg EM122 data and evaluated for its applicability in this thesis. With focus on the XYZ and seabed image datagrams, the underlying data was first analyzed. Next, a Python module for generating attributed point clouds was developed and, based on this, an interactive notebook for filtering and correcting the data was built with the open source tools PDAL, Entwine and Potree. Validation was performed with reference data processed on board with commercial software, and independent data from a Kongsberg EM 710 multibeam. Generally, the applicability of the approach could be confirmed with the chosen implementation. Marginal differences have been found in pre-processing. Overall, Jupyter Notebook has proven to be effective, but processing steps must be clearly defined for specific applications and cannot be scaled up arbitrarily.

Zusammenfassung

Heutzutage sind Fächerecholote die gängigste und effizienteste Methode zur Durchführung hydrographischer Vermessungen. Die Verarbeitung hydrographischer Daten (Bathymetrie und Rückstreuung) erfordert eine Reihe von Prozessen, um zuverlässige Informationen zu erhalten. Mehrere kommerzielle und wissenschaftliche Softwarelösungen sind für diesen Zweck bereits verfügbar, aber die Nutzung ist nicht immer zugänglich. Dem soll durch den Einsatz von Open-Source-Bibliotheken begegnet werden. Die Kombination von Jupyter Notebook und Python hat großes Potenzial für die interaktive Datenverarbeitung. In dieser Arbeit wurde ein Workflow für die Verarbeitung von Bathymetrie- und Rückstreudaten auf Basis von Kongsberg EM122-Daten entwickelt und auf seine Anwendbarkeit hin getestet. Mit Fokus auf die XYZ- und seabed image-Datagramme wurden zunächst die zugrundeliegenden Daten analysiert. Anschließend wurde ein Python-Modul zur Generierung attributierter Punktwolken entwickelt und darauf aufbauend ein interaktives Notebook zur Filterung und Korrektur der Daten mit den Open-Source-Tools PDAL, Entwine und Potree erstellt. Die Validierung erfolgte mit Referenzdaten, die mit kommerzieller Software an Bord verarbeitet wurden, sowie mit unabhängigen Daten von einem Kongsberg EM 710 Fächerlot. Generell konnte die Anwendbarkeit des Ansatzes mit der gewählten Implementierung bestätigt werden. Marginale Unterschiede wurden bei der Vorprozessierung gefunden. Insgesamt hat sich Jupyter Notebook als effektiv erwiesen, allerdings müssen die Verarbeitungsschritte für spezifische Anwendungen klar definiert sein und können nicht beliebig hoch skaliert werden.

Contents

List of Figures

List of Tables

1. Introduction

Nowadays, multibeam echo sounders (MBES) are the most common and efficient method to conduct hydrographic surveys. Thereby MBES systems have the intrinsic capability to not only measure the geometry of the seafloor (bathymetry) but also its reflective properties (backscatter) which to some extent provide information about the characteristics of the seafloor. Due to the challenging conditions of the marine environment and the complex setup, inherent to MBES measurements, both, bathymetry as well as backscatter data, require various processing steps in order to provide reliable results.

Software suites that address the needs of MBES data are already available. In general, they can be divided into two large groups: commercial software and software originating from the work of scientific institutes. Software suites of the former usually offer very comprehensive solutions with the flexibility to adapt the differing demands. However, the licenses are most often very expensive. On the other hand, the available scientific software ranges from very specific implementations, to toolboxes as extensive as the commercial software. Often these toolboxes are modular and have a command-line or script-based user interface rather than a graphical one. Although they often have free licenses, they usually lack accessibility in terms of platform requirements and the level of knowledge expected from the user.

Python stands out among the variety of programming languages available due to its accessibility in terms of its simplicity, but also its fast implementation cycle and the wide variety of specialized libraries and open source packages offered by the scientific community. The Jupyter project was a fork of IPython (Interactive Python), specifically to support data science and scientific computing, with the stated aim of promoting the development of open source software and services for interactive computing. These efforts have resulted in, among other things, the Jupyter Notebook, a web application that allows code, as well as other rich, interactive output, to be shared, providing a unique way of integrating code and documentation.

Students in particular struggle with the disadvantages of the available software, as they usually cannot afford the commercial licenses, nor do they have the skills to use the scientific software. The combination of Jupyter Notebook and Python has great potential to bridge this gap. First, Jupyter Notebook per se provides a graphical user interface that allows the combination of executable code and explanatory documentation. Python is one of the programming languages that is said to be easy to learn. Therefore, in combination with the documentation, it has a high potential to provide easy accessibility while using tools and algorithms offered by the large Python community. The goal of this thesis is to explore the capabilities of MBES bathymetry and backscatter data processing using Jupyter Notebook and Python. The focus will be primarily on Kongsberg EM122 data collected during RV Sonne cruise SO268-3 to assess its general suitability.

2. Background

A multibeam echo sounder (MBES) is a type of active sonar[1] that is used to efficiently map the seafloor. As all active sonars, MBES systems emit short acoustic pulses called pings towards a target, usually the seafloor, and listen for the returning echoes. The ping of a MBES consists of multiple acoustic beams arranged in a fan-shape called swath. With one ping, a MBES swath can simultaneously measure up to a few hundred soundings covering a narrow strip of seafloor in athwart ship direction with a width of up to several times the water depth. Nowadays, MBES systems are widespread in all kind of applications ranging from nautical charting over research to industrial applications. They are meanwhile deployable from various support platforms such as ships, unmanned underwater or surface vehicles, or towed containers (towfish).

While MBES systems first appeared in the 1970s (Lurton, 2002), the need for reliable hydrographic information is much older. First traces of primitive depth measurements date back to 3,700 years old Egyptian tomb paintings showing depth measurements with poles, spades and ropes for the location of potential navigational hazards nearshore (Grządziel and Wąż, 2018). Such and similar techniques were used over centuries, eventually evolving in complexity to mechanical sounders. First practical devices utilizing the good propagation properties of sound in water were passive sonar systems used by Allies during World War I (Lurton, 2002). Although the idea of sonar systems actively transmitting acoustic energy was not new then, it was not until the Titanic disaster in 1912 which finally triggered the development of active sonar systems. One of the first patents came 1913 from the German physicist Alexander Behm who was initially searching for a method to detect icebergs but eventually found that the seafloor was an excellent acoustic reflector (Grządziel and Wąż, 2018). First hydroacoustic surveys were carried out in the 1920s using single-beam echo sounders (SBESs). In the following years, the development of sonar technology was mainly driven by the upcoming World War II and further enhanced by improvements in computer performance. In the 1970s first commercial deep water MBES models appeared increasing the efficiency of seafloor acoustic mapping drastically by multiplying the number of simultaneous soundings. These days, a variety of MBES systems exist, covering a wide range of applications. Generally, they may be distinguished into two groups: Swath and sweep systems. Swath MBESs produce multiple beams from a single transducer whereas sweep MBESs consist of an array of SBESs mounted on sideward pointing carriers (de Jong, 2002). Within this work, only swath MBES systems are addressed, and therefore the term MBES refers to swath systems only.

2.1. Multibeam echo sounder theory

Most of the modern (swath) MBES systems use the Mills cross technique to form a swath of acoustic beams (Fig. 2-1). A Mills cross consists of two perpendicular transducer arrays, a projector array aligned along-track which transmits sound waves and a hydrophone array in across-track direction which receives the echoes. A transducer converts an electric signal into an acoustic and vice versa. That is because the diaphragm of the transducer starts vibrating when exposed to electric power. In contact with water, the vibration generates an acoustic wave. Likewise, an incoming acoustic wave causes a vibration of the diaphragm which can be translated into an electric current (de Jong, 2002). Even though the same transducer can be used to transmit and receive acoustic waves, one may

[1]SOund NAvigation and Ranging

distinguish transducers which are only used for sound transmission, called sound sources, and those only used for reception, called hydrophones. As a sound pulse expands spherically from its source, an individual sound source cannot transmit acoustic energy in a specific direction. By arranging multiple sound sources in a line array, the pattern of sound interference can be used to achieve some degree of directivity. Thereby the basic idea is that constructive interference focuses a main lobe of acoustic energy perpendicular to the transducer array, while the spread of sound in other directions (side lobes) is strongly suppressed by destructive interference. That way the projector array forms a transmitting beam that ensonifies an area of seafloor which is wide across-track and narrow along-track. The same beam pattern applies to the hydrophones of the receiving array which are in contrast sensitized to receive sound from a certain direction. By introducing a time delay to the individual hydrophone readings, the main lobe of the beam pattern can computationally be shifted such that its axis is steered to some angle from the perpendicular. That way, the hydrophone array can receive parallel beams at different angles that are narrow across-track and wide along-track without being physically changed. The intersections of the area ensonified by the projector array and the multiple strips observed by the steered beams of the hydrophone array are the discrete locations (footprints) of the soundings in each ping (SeaBeam, 2000).

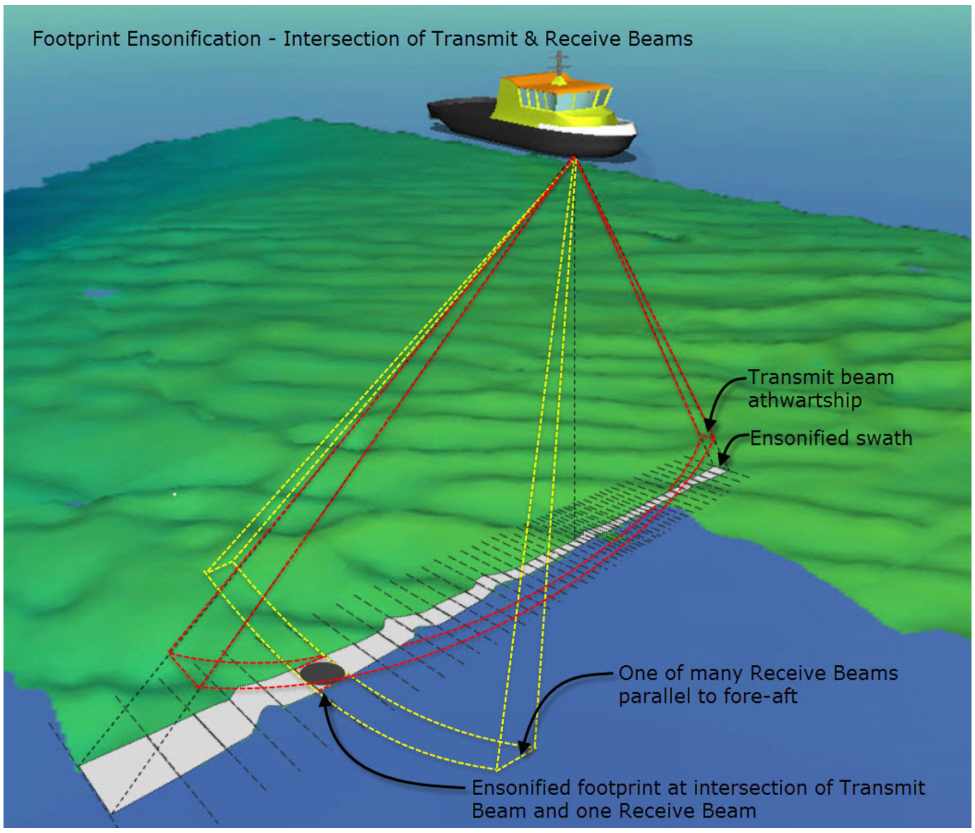

Footprint Ensonification - Intersection of Transmit & Receive Beams
Transmit beam athwartship
Ensonified swath
One of many Receive Beams parallel to fore-aft
Ensonified footprint at intersection of Transmit Beam and one Receive Beam

Figure 2-1: Mills cross arrangement of projector (transmit) and hydrophone (receive) array to form a swath of beams which is narrow along-track and wide across-track (*How-to-FMGT: FMGT Supported Data Formats* 2020).

Depending on the use of a MBES, the desirable characteristics of a system vary. One of the most essential properties is the acoustic frequency. It determines the range and sediment penetration of echo sounders. This is primarily because the attenuation of the acoustic signal in water is proportional to its frequency. Attenuation is the combined loss of energy caused by absorption, spherical spreading and scattering by particles in the water column (Lurton, 2002). The higher the frequency, the stronger the signal is attenuated and thus the lower its range and sediment penetration. The acoustic frequencies of the different MBESs range from:

- Deep-water systems (deeper than 1,500 m) with 12 to 50 kHz,
- over shallow-water systems (shallower than 1,500 m) with 50 to 200 kHz,
- up to high-resolution systems (shallower than 100 m) with frequencies higher than 200 kHz.

The acoustic frequency of the MBES also influences the beam width[2] of individual beams in the swath. To produce a signal with a lower frequency but the same beam width, a larger transducer is required or, conversely, using the same transducer, a decrease in frequency will result in a wider beam (de Jong, 2002). Typical beam widths are in the order of less than one degree up to a few degrees per beam. For high-frequency echo sounders, the same transducer array can typically be used for both transmission and reception so that the transducer size is small enough to be mounted e.g. on underwater vehicles such as ROVs. However, low-frequency echo sounders require physically separated transmission and reception arrays with a size of up to several meters. Only sufficiently large vessels are able to carry the heavy equipment in the hull of the ship. For those systems, the transmission array alone defines the along-track resolution while the across-track resolution is imposed by the reception array. The footprint of an individual beam typically has an ellipsoidal shape and can be computed as the product of the beam pointing angle θ, the beam width of the transmission ϕ_{T} and reception array ϕ_{R}, the (mean) water depth z and the mean slope of the seafloor. Under the simplification of a flat seafloor, the length of the ellipse in athwart direction a_y may be approximated by (IHO, 2005):

$$a_y = \frac{2z}{\cos^2(\theta)} \tan\left(\frac{\phi_{\mathrm{R}}}{2}\right) \tag{2-1}$$

And the width of the footprint, which can be reasonably approximated as a set of overlapping beam ellipses, in fore-aft direction a_x is roughly given by (IHO, 2005):

$$a_x = \frac{2z}{\cos(\theta)} \tan\left(\frac{\phi_{\mathrm{T}}}{2}\right) \tag{2-2}$$

Assuming a MBES with a swath opening angle of 140° (max beam pointing angles of ± 70°) and a TX x RX beam width of 0.5 x 1°, one can exemplary determine the approximate footprint of a beam at any given beam pointing angle and depth using equations 2-1 and 2-2. For a depth of 1,500 m, the inner beam (beam pointing angle of 0°) has a footprint length of 26 m and a width of 13 m. The outer beam at a beam pointing angle of 70° has a length of 224 m and a width of 38 m. At a depth of 3,000 m, the inner beam already has a footprint of 52 x 26 m and the outer beam 448 x 77 m. The width of the entire swath for a flat seafloor can by computed by:

$$S_{\mathrm{W}} = 2z \tan\left(\frac{\Delta\theta}{2}\right) \tag{2-3}$$

Whereas $\Delta\theta$ is the swath opening angle. For the same MBES system as mentioned above, the

[2]Considered at –3 dB

swath width at a depth of 1,500 m would be roughly 2,100 m, at double the water depth (3,000 m), the swath width also doubles to 4,200 m.

While the horizontal resolution of a MBES is determined by the beam width, the target depth (and geometry) and the beam pointing angle, the vertical resolution is given by the pulse length. The pulse length is the product of the pulse duration and the sound velocity in sea water which is approximately 1,500 m s^{-1}. The variation of pulse length helps to overcome attenuation losses in the water column. However, with an increase of the pulse length, the vertical resolution decreases as two objects are only distinguishable if they are more than ½ the pulse length apart. For a signal with a pulse duration of 0.5 ms and thus a pulse length of 0.75 m, the vertical resolution is roughly 0.38 m. For a pulse duration of 2 ms, the vertical resolution already decreases to 1.5 m (de Jong, 2002).

As the sound pulse travels trough the water to the seafloor and back again, the propagation properties in water play an important role in the measurement process. However, the sound propagation properties are not constant but actually vary throughout the water column. They depend on temperature, salinity and depth, which in combination define the density of water. For the upper water column between the surface and the thermocline[3], temperature is the dominant factor. Temperature is highly variable depending on geographical and oceanographic seasonal changes. Below the thermocline, depth becomes the major impact. As depth increases the hydrostatic pressure grows, which causes an in first approximation linear increase of sound velocity (Lurton, 2002).

Generally, water layers with very different densities tend not to mix because of an effect called stratification. At the borders of two media with different propagation properties, an acoustic ray gets refracted and reflected. Refraction occurs due to changes of the acoustic impedance (product of water density and sound velocity) either continuously or at distinct bounding surfaces. According to Snell's law, refraction causes the path of a sound wave to bend (Hovem, 2013). After the reflection at the seafloor, the sound wave propagates back to the MBES and gets refracted again. This effect occurs especially at the outer beams of the swath where the acoustic paths are longer and the incidence angles at bounding surfaces larger. To correct the acoustic path, sound velocity profiles (SVPs) are required. Two of the mostly used devices to acquire SVPs are sound velocity and conductivity, temperature, depth (CTD) probes. Both devices are lowered through the water column and take measurements on their way to create the SVPs. Sound velocity probes directly determine the sound velocity by measuring the time, which a ping needs to travel a short known distance, whereas CTDs measure each parameter of the sound velocity individually (conductivity/salinity, temperature and depth/pressure) to calculate the sound velocity. Another sound velocity probe is usually permanently mounted at the transducer face for sound velocity measurements to improve the beam forming (IHO, 2005).

Other ancillary measurement systems required are a motion sensor and a position system to compensate for the ships motion and to transform the MBES measurements from the vessel coordinate system to a global one. Usually the vessel coordinate system is a right-handed system with x pointing towards the bow, z pointing downwards and y completing the right-handed coordinate system (pointing towards starboard). Generally, a ship can rotate around and translate along three axis. Thereby, the three rotations pitch (around the y-axis), roll (around the x-axis) and yaw (around the

[3]Water layer between the warm surface water and the cooler deep water below

z-axis) which are referred to as the ships attitude, and the vertical translation heave are relevant for MBES systems. To convert the measured beams of the swath from polar coordinates relative to the MBES transducer to the vessel coordinate system, the ship's attitude and heave need to be determined. Inertial measurement units (IMUs) consisting of three accelerometers and three angular rate sensors are the most common sensors for roll, pitch and heave. Yaw, when measured in degrees from north (true, magnetic or compass), is referred to as heading. Heading is especially important for swath systems. It can be measured by gyro- or fluxgate compasses or differential GNSS[4]. GNSS is also the preferred system for positioning. The position system is required to transform from the vessel coordinate system to a global system. Commonly, GNSS sensors are used for positioning either in pseudo-differential or in RTK[5] mode (IHO, 2005; de Jong, 2002).

2.1.1. Bathymetry

The bathymetry measurement is the primary functionality of any hydrographic echo sounder, including MBESs. It aims to retrieve accurate information on the geometry of the seafloor. The fundamental principle is the measurement of the time delay t between signal transmission and reception and its association with the (slant) range R between transducer and seafloor (Fig. 2-2). Provided that the beam pointing angle θ and the sound velocity c are known, the two way travel time can be converted to depth z and distance y in athwartships direction from the MBES transducer (Lurton, 2002):

$$\begin{aligned} z &= \frac{ct}{2}\cos\theta = R\cos\theta \\ y &= \frac{ct}{2}\sin\theta = R\sin\theta \end{aligned} \tag{2-4}$$

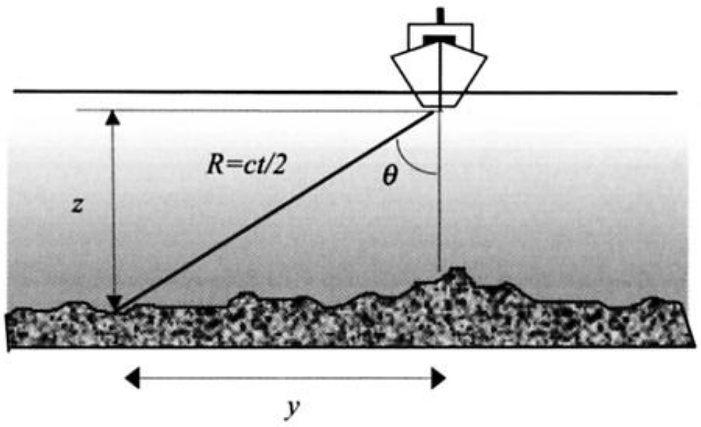


Figure 2-2: Fundamental principle of MBESs: Water depth z and (across-track) distance y as derivatives of range R to the seafloor and beam pointing angle θ (Lurton, 2002).

2.1.1.1. Measurements

There are different techniques which can be used to determine the time-angle couples of each beam bottom detection. The two main divisions being (IHO, 2005):

[4]Global Navigation Satellite System
[5]Real Time Kinematic

- Amplitude detection: The returned signal is sampled in time for each beam angle. The sample amplitudes belonging to a beam are then used to determine the travel time of a depth point. There are different methods how to use the sample amplitudes for detection (leading edge, maximum amplitude or center of mass of the reflected signal). Typically, amplitude detection algorithms are applied to beams around normal incidence with high amplitudes and fewer samples.
- Phase detection: For outer beams with more grazing angles, the sample amplitudes decrease and the number of samples may become very large. The amplitude gets thus more blurred resulting in poor results when applying amplitude detection algorithms. The phase detection algorithm artificially subdivides the transducer array of a beam in two sub-arrays. Each of the two forms a beam in the same direction (beam angle). The time of arrival is then defined by the zero crossing of the returning sequence of phase differences “corresponding to the arrival of a signal from a target exactly on the beam axis” (Lurton, 2002).

Generally, inaccuracies in determining the time-angle couples of each beam decisively influence the accuracy of the beam coordinate estimation. The errors induced by an erroneous time measurement are given by (Lurton, 2002):

$$\begin{aligned} \delta z_t &= \delta R \cos\theta \\ \delta y_t &= \delta R \sin\theta \end{aligned} \tag{2-5}$$

For angular error the impact on the y, z coordinates is given by (Lurton, 2002):

$$\begin{aligned} \delta z_\theta &= R \sin\theta \,\delta\theta \\ \delta y_\theta &= R \cos\theta \,\delta\theta \end{aligned} \tag{2-6}$$

So far the x coordinate in fore-aft direction was neglected based on the assumption that the sensor is perfectly aligned with the ships axis and hence x = 0. It is at this point noteworthy that angular errors cause x to deviate from 0.

The overall accuracy is thereby not only influenced by errors in the acoustic measurement itself but also by inaccuracies in the sound velocity correction and movement of the support platform. The latter, namely the attitude motion parameters roll, pitch and yaw and the vertical motion parameter heave are measured in real-time. The accuracy demand depends on the performance requirements of the survey. If high performance is pursued, roll and pitch are usually required with an accuracy of 0.05° and heave with an accuracy of 0.05 m to 0.1 m (de Jong, 2002). Modern MBES systems mostly compensate for the attitude of the support platforms in form of a pitch, roll and sometimes also yaw stabilization. However, stabilization does not eliminate residual angular biases.

Equation(s) 2-4 are based on the assumption of a constant sound velocity throughout the water column, and hence, a rectilinear acoustic path. But as explained above, the sound velocity in reality varies throughout the water column causing the path of a sound pulse to bend. To reconstruct the actual acoustic trajectory of a beam, ray[6] tracing is applied. Based on sound velocity profiles, sounding point positions are estimated starting with the initial beam angle by following the sound trace as

[6]Normal to the wavefront

a function of time along the water column. When the function reaches the measured time t, the computation is stopped and the outermost extremity of the estimated path is set to be the sounding position (Lurton, 2002). Another component of the sound velocity induced uncertainty arises from variations of sound velocity at the transducer face. In order to steer the receiving beams, the MBES uses a sound velocity which is typically measured close to the transducer face. Knowing the exact sound velocity is paramount to determine the right time delays for the individual hydrophone readings along the receiver array. Deviations from the actual sound velocity cause an error in the beam pointing angle (IHO, 2005).

Hare (1995) describes the total error budget of the MBES measurement in great detail. Without stressing to much on excessive explanations, it may be summarized as follows:

- Depth error budget:
  - MBES system error, in fact range and beam angle measurement error
  - Error of support platform's attitude (especially roll and pitch) caused either by measurement errors, possible stabilization errors and/or angular misalignment of sensors
  - Error of measured or induced (by roll and pitch) heave
  - Errors caused by refraction due to sound velocity variations affecting range, beam angle and beam steering
- Error budget for the sounding position:
  - Positioning system (GNSS) error
  - Latency error or rather the lack of knowledge of latency
  - Error of the relative sounding position to the transducer due to MBES system errors, error in attitude measurement or sensor alignment, or refraction errors
  - Heading error due to measurement error or transducer yaw misalignment
  - Error of the relative positions of the GNSS antenna(s) and the transducer

For the sake of completeness, the depth error budget has to be extended to include dynamic draft errors which are caused by squat and load, and water level errors such as the measurement and prediction errors of tidal effects (Hare, 1995).

2.1.1.2. Processing

The MBES bathymetry measurement is a very complex process due to the swath geometry, the many ancillary sensors required and the data volume which they produce. Moreover, the initial interest of hydrographic surveys evolved from the safety of navigation. Accordingly, the accuracy and also the reliability demands on those surveys are high.

The International Hydrographic Organization (IHO) published and continuously develops the S-44 Standards for Hydrographic Surveys which aim "to provide a set of standards for hydrographic surveys primarily used to compile navigational charts essential for the safety of navigation, knowledge and the protection of the marine environment" (IHO, 2020). It thereby has to be understood as minimum standards which have to be specified for the intended task. Generally, IHO (2020) defines different (safety of navigation) survey orders depending on the water depth, geophysical properties, and expected shipping types.

There are three different types of errors which can be found in bathymetric data sets (Artilheiro, 1998):

Blunders are large errors caused by a machine, either a defective mechanical or electronic device. Another term often used in a surveying context is outlier. Outliers are defined as errors of random quantity which lie outside some arbitrary statistical limits. The term covers both blunders but also the set of actually existing hazards or minimum depths which fall out of the expected distribution of data points. Within this thesis, the terms are rather used in an equivalent manner.

Systematic errors are the second type of error which might occur in bathymetric data sets. They are mainly caused by constant offsets (fixed) or biases (variable) either spatially or in time. Examples for this are time offsets of sensors, sensor misalignment or a wrong sound velocity profile. Those errors can be minimized by proper calibration procedures and good surveying practice (Le Deunf et al., 2020).

Finally, after removing blunders and systematic errors, stochastic deviations or also random errors remain. Those errors are intrinsic to measurement processes. Usually they follow a normal distribution and can be analyzed statistically.

For hydrographic purposes it is vital to correctly remove blunders without mistakenly deleting soundings actually representing solid objects above the seafloor, such as wrecks or boulders. There are different possible causes for the occurrence of blunders. They might be subdivided into environmental factors and system internal problems. Environmental factors include all possible circumstances under which a reflection is caused in the water column. For example shoals of fish or kelp, abnormal sound velocity variations due to sudden jumps in temperature or salinity, or multiple reflections and paths. On the other hand, system internal problems with the bottom detection algorithm or equipment malfunction can also be a potential cause of blunders (Artilheiro, 1998). Generally, outer beams are more prone to errors as inner beams as the signal has a longer way through the water column, more grazing incidence angles and errors occurring in the MBES system itself show stronger impact.

In the previous section the MBES error budget was briefly described comprising the depth and position error. IHO (2008) outline the generic steps necessary to address and minimize all those error sources. It should be mentioned though that in the latest edition (6.0) the guidelines for data processing (annex B) are already outsourced, as their are meant to be included in the next edition of the Manual on Hydrography (IHO M-13).

Generally, it is proposed to use all information sources available to ensure the presence of " navigationally significant soundings" (IHO, 2008). Also, it is not recommended to delete any kind of raw data but to rather flag data as rejected if it is doubtful.

At first all the individual sensor raw data should be checked and if required corrected and/or cleaned. A summary of the guidelines is presented in table 2-1.

Subsequently, the post-processed raw sensor data should be checked for any sensor latency and merged to retrieve the final x (longitude), y (latitude), z (depth) coordinates of the soundings. Thereby the spatial offsets of the system set up have to be considered.

The next step is the detection of blunders in the bathymetric point cloud. One may distinguish two

Table 2-1: Post-processing of individual raw sensor data in the MBES system set up following the guidelines outlined in IHO (2008)

Position data	Merging of position data from different sensors (if necessary) Checking the position data Eliminating/correcting position jumps
Attitude data	Checking the attitude data (heading, pitch, roll and heave) Eliminating/correcting data jumps
Sound velocity data	Checking sound velocity profiles Correcting two-way travel time and refraction Real-time correction should be possible to override in post processing
Depth data	Corrections for potential water level changes Corrections for motion of support platform Corrections for draft of ship It should be possible to re-process data which has been corrected in real-time

major types of data cleaning, namely automatic (non-interactive) versus manual (interactive). For a long time it was common practice to manually clean the data. The detection of blunders completely relied on the subjective decision of a trained hydrographer. The manual cleaning procedure is very lengthy and time consuming. Often it takes double the processing time compared to the acquisition time (Le Deunf et al., 2020). Also it does not guarantee that all blunders are detected, respectively, all outliers representing actual objects are preserved. That is why generally the use of automated or semi-automated approaches is recommended by IHO (2008) on condition that the applied algorithm(s) have been documented, tested and demonstrated to produce repeatable and accurate results. Suitable algorithms comprise statistical techniques such as robust estimation but also simple threshold values. It is furthermore recommended to review the automated result for validation and/or to resolve potential ambiguities. Thereby it is helpful to display all available information such as backscatter or detection information in a suitable manner, at best interactively, to support the decision making.

2.1.2. Backscatter

While the bathymetry relies on the time and angle of the returning echo to retrieve geometrical information of the seafloor, backscatter is based on the intensity of the returning signal to determine seafloor (acoustic) reflectivity. The reflective properties of the seafloor can directly be related to the nature of the seafloor and some of its physical characteristics. Generally, the term backscatter refers to the part of a sound wave which is scattered back to the direction it came from. Usually backscatter is caused by a diffuse reflection at a rough surface whereby, indeed, most of the incident acoustic wave is reflected away or scattered in other directions (Fig. 2-3). The sole exception is a sound wave at normal incidence, where backscatter is caused by the (specular) reflection. Backscatter does not only occur at the seafloor but also at particles in the water column such as fish or gas bubbles. Water column backscatter itself is an active area of research, however, it is not addressed in this work and therefore the term backscatter refers to seafloor backscatter only (Weber and Lurton, 2015).

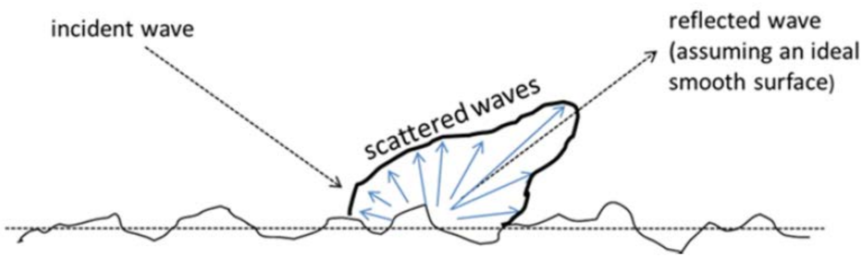


Figure 2-3: Backscatter is the part of an incident (acoustic) wave which is scattered back whereby most of the wave is reflected away or scattered in other directions (Weber and Lurton, 2015).

2.1.2.1. Measurements

The interest of (seafloor) backscatter focuses on the acoustic response of the seafloor. Unfortunately, the intensity of the backscattered acoustic wave does not only depend on the effect of the seafloor but rather on the continuous interaction with its environment from signal transmission until reception. The active sonar equation is commonly used to quantify the performance of an echo sounder in consideration of the environmental conditions. Following an implementation of the sonar equation by Weber and Lurton (2015), the intensity of the backscatter (echo level EL) is the sum of the source level SL, the transmission loss TL and the target strength TS:

$$\mathrm{EL} = \mathrm{SL} - 2\mathrm{TL} + \mathrm{TS} \tag{2-7}$$

The MBES transmits a signal with a source level SL (in dB re 1 µPa at 1 m). Afterwards the acoustic wave propagates through the water, where its signal intensity decays/attenuates because of spreading and absorption. The total effect of both is the transmission loss (TL). It depends on the range to the seafloor R and the absorption coefficient α. The absorption coefficient (α in dB m^{-1}) of water depends, just as the sound velocity, on the water properties but also the signal frequency. Assuming that the transmission loss is identical on the way to the target and back to the receiver, the impact of transmission (TL in dB) loss is doubled (Hammerstad, 2000):

$$2\mathrm{TL} = 2\alpha\mathrm{R} + 40\log_{10}\mathrm{R} \tag{2-8}$$

There are generally three different kinds of noise in the water column, ambient noise caused by, among others, breaking waves, marine mammals or ships, self-noise of the sonar itself or its platform, and reverberation which is the effect of backscatter from features of no interest (Weber and Lurton, 2015). However, equation 2-7 neglects the contribution of noise assuming that the echo level is sufficiently high to significantly exceed the noise level.

Finally, the target strength (TS) is the effect of a target such as the seafloor where some part of the acoustic wave is scattered back to the MBES receiver. Target strength (TS in dB re 1 m^2) is the relation of the intensity backscattered at a target to the incident intensity (SL – TL). It is the quantity of the sonar equation relevant for backscatter measurements. The target strength of the seafloor depends on the scattering cross section σ and the ensonified area A (Weber and Lurton, 2015):

$$\mathrm{TS} = 10\log_{10}(\sigma\,\mathrm{A}) = \mathrm{S_b} + 10\log_{10}\mathrm{A} \tag{2-9}$$

$\mathrm{S_b}$ is referred to as the bottom scattering strength. $\mathrm{S_b}$ is independent of the sonar properties (except the frequency), but it depends on the incidence angle at the seafloor and on some seafloor dependent parameters including its impedance (product of density and sound velocity), interface roughness and heterogeneities within the sediment volume (biological, mineral, gas, etc.). In colloquial language, one may say that the bottom scattering strength depends on the "hardness" and "roughness" of the seafloor and possibly also the degree of homogeneity.

Hardness is thereby defined as the contrast between the characteristic impedance of the seafloor compared to the water above. Water has roughly a sound velocity of 1,500 $\mathrm{m\,s^{-1}}$ and a density of 1,000 $\mathrm{kg\,m^{-3}}$ which yields an impedance Z of 1.5x $10^{-6}\,\mathrm{kg\,m^{-2}\,s^{-1}}$. In contrast, clay has an impedance of 1.68x $10^{-6}\,\mathrm{kg\,m^{-2}\,s^{-1}}$ and coarse sand has an impedance of 4.18x $10^{-6}\,\mathrm{kg\,m^{-2}\,s^{-1}}$ (numbers extracted from Weber and Lurton, 2015). The impedance contrast between water and clay is hence much smaller than between water and coarse sand, and thus, coarse sand is acoustically speaking harder than clay. That means that comparably more of the acoustic energy is reflected from the coarse sand as from the clay, and conversely, less of the acoustic energy is transmitted into the seafloor. The amount of energy which is reflected at the boundary between seawater (medium 1) and seafloor (medium 2) is given by the reflection coefficient R (Weber and Lurton, 2015):

$$\mathrm{R} = \frac{\mathrm{Z_2}\cos\theta_\mathrm{i} - \mathrm{Z_1}\cos\theta_\mathrm{t}}{\mathrm{Z_2}\cos\theta_\mathrm{i} + \mathrm{Z_1}\cos\theta_\mathrm{t}} \tag{2-10}$$

Where θ_i is the incident angle and θ_t is the transmitted angle. These two are related by Snell's law (Weber and Lurton, 2015):

$$\frac{\sin\theta_\mathrm{i}}{c_1} = \frac{\sin\theta_\mathrm{t}}{c_2} \tag{2-11}$$

Considering equations 2-10 and 2-11, the reflection coefficient increases with the incident angle and hence less acoustic energy is transmitted into the seafloor. This is valid until the so called critical angle, which is so grazing that no energy is transmitted into the seafloor.

The seafloor roughness is the effect actually enabling swath sounding systems (with combined transmitters and receivers) in first place, as otherwise all the acoustic energy would simply be reflected away. Roughness in this context has to be understood in relation to the signal wavelength. Typical MBES wavelength range from circa 0.38 cm for 400 kHz high-resolution systems to 12.5 cm for 12 kHz deep-water systems. The seafloor interface is considered rough if the interface regularities are much greater than the wavelength and smooth if the interface regularities are much smaller. Figure 2-4 shows the backscatter level (dB) for a signal with a high (HF) and a low frequency (LF) as a function of the incidence angle. The scenario on the right side shows an acoustically rough seafloor compared to HF as well as the LF. The backscatter level is governed by interface roughness scattering with almost no specular reflection. Even at grazing angles, the backscatter level is high. In contrast, the left scenario shows a low-moderate acoustic roughness. In that case, most of the scattered wave is still focused around the specular direction and a lesser part is scattered in other directions. For near-normal incidence where backscatter is mainly caused by the specular reflection, the backscatter level is high. The interface roughness scatters only little acoustic energy at grazing angles back to the receiver. The effect is stronger for the low frequency (hence longer wavelength) as the interface is comparably smoother (Weber and Lurton, 2015).

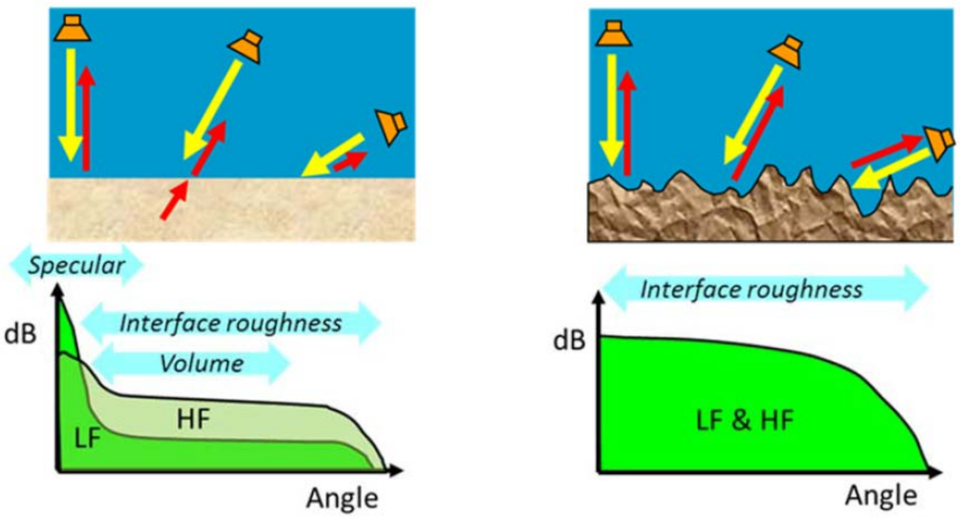


Figure 2-4: Backscatter (in dB) as function of incidence angle. The acoustically rougher and harder the seafloor is, the stronger the specular reflection and volume scattering are dominated by interface roughness scattering (Weber and Lurton, 2015).

In addition to interface scattering, the left scenario in figure 2-4 also shows volume scattering from within the sediment volume itself. Volume scattering occurs at inhomogeneities in the sediments such as gas bubbles, benthic animals etc. or heterogeneities of the sediment itself such as a change of the sediment composition. Those features are hit by the part of the sound wave which is transmitted into the seafloor and generate a reflection themselves. The sediment volume backscatter is maximal at the intermediate oblique incidence angles between the specular reflection in the normal direction and very grazing angles close to the critical angle. Also, the effect is stronger for softer sediments (reflection vs. transmission coefficient) and deeper frequencies (Weber and Lurton, 2015).

2.1.2.2. Processing

Processing of backscatter data has generally two possible outcomes, backscatter mosaics and the less common angular response. Backscatter mosaics are georeferenced images, typically in grayscale, representing the (seafloor) backscattering strength (BS) or a related variable. Angular response describes the variation of BS as a function of the angle of incidence at the seafloor (Schimel et al., 2015). Both representations aim to enable the identification of different seafloor types and where applicable corresponding regions. To produce any meaningful output from the raw backscatter data, a sequence of processing steps has to be applied (Fig. 2-5). The actual implementation of backscatter processing highly varies depending on the MBES system and the used processing software. For this work, the processing steps and order, outlined by Schimel et al. (2018), are used as a generic approach.

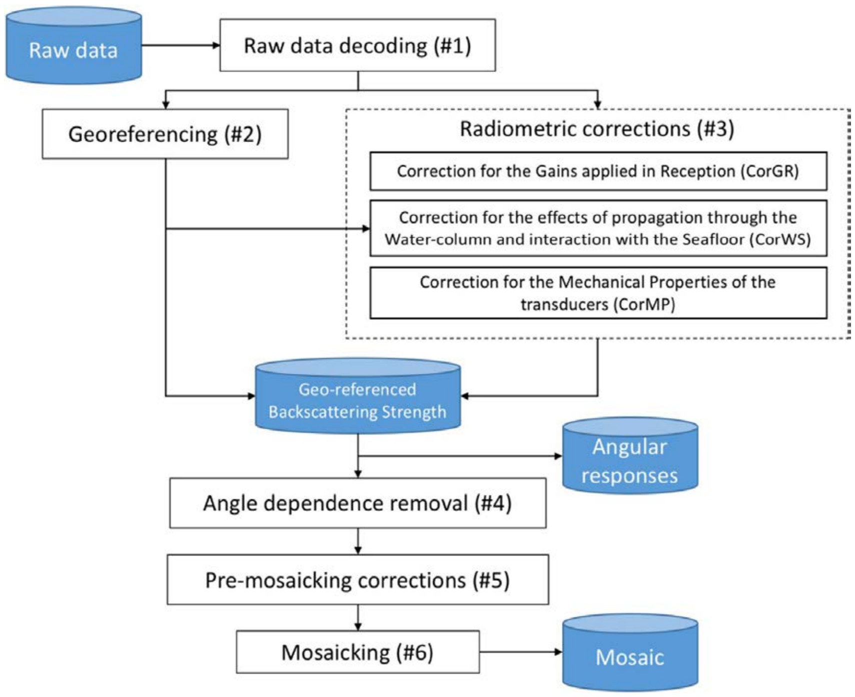

Raw data
Raw data decoding (#1)
Georeferencing (#2)
Radiometric corrections (#3)
Correction for the Gains applied in Reception (CorGR)
Correction for the effects of propagation through the Water-column and interaction with the Seafloor (CorWS)
Correction for the Mechanical Properties of the transducers (CorMP)
Geo-referenced Backscattering Strength
Angular responses
Angle dependence removal (#4)
Pre-mosaicking corrections (#5)
Mosaicking (#6)
Mosaic

Figure 2-5: Backscatter processing workflow (Schimel et al., 2018).

Raw data decoding

In a first step, the relevant raw sonar data needs to be decoded. There are three main types of delivery for raw backscatter data (Schimel et al., 2018):

1. “Single value per beam” whereby one single intensity value is recorded for each beam which corresponds either to the amplitude value of the bottom detection or an average amplitude of all samples in the beam footprint.
2. “Beam time series” consist of series of intensity samples for each beam mimicking the propagation of the acoustic signal over the seafloor (Fig. 2-6).
3. “Half-swath time series” which consist of an uninterrupted intensity sample series for each side of the swath, respectively.

Depending on the sonar manufacturer, different types of backscatter data may or may not be available and also the actual implementation might vary.

Georeferencing

In a second step the intensity samples are georeferenced. That means that the geographical location of each sample, whether it is an individual value or part of a series, has to be determined based on the available bathymetric information. For the single value per beam samples, the implementation is very much straightforward. The location of an individual sample is directly provided by the bathymetry of its associated sounding point. For the other two types of backscatter data it is not as simple. In general, there are two approaches to georeference beam time series data. The first is to find the sample in each beam time series which corresponds to the bottom detection and georeference it by using the respective bathymetric information. The other samples in the time series are then georeferenced by interpolation (Schimel et al., 2015). This approach appears simple at first,

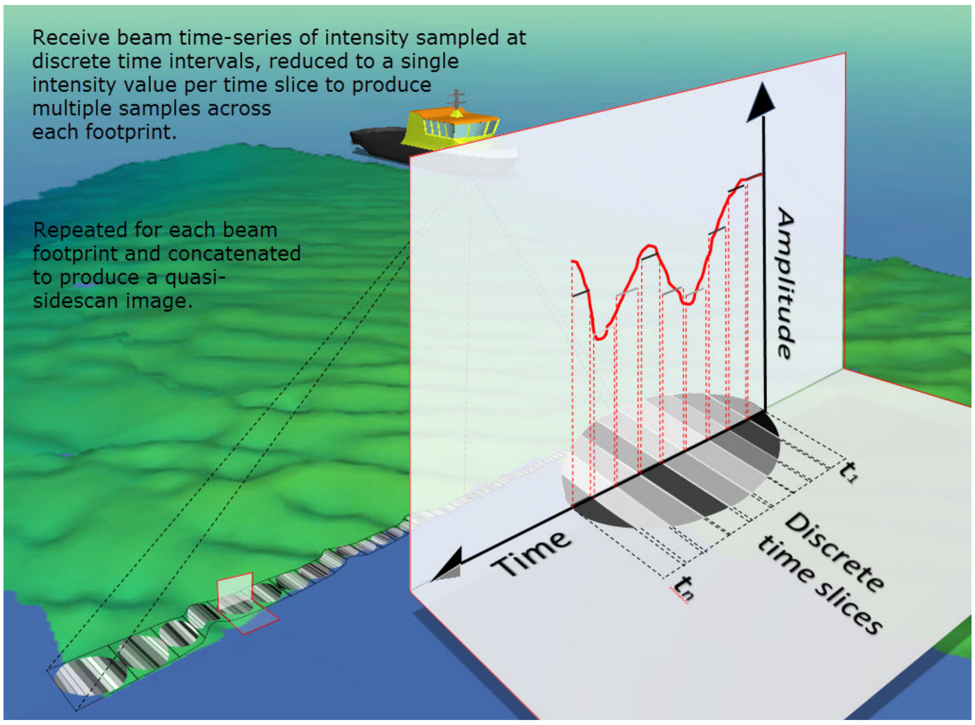

Receive beam time-series of intensity sampled at discrete time intervals, reduced to a single intensity value per time slice to produce multiple samples across each footprint.
Repeated for each beam footprint and concatenated to produce a quasi-sidescan image.
Amplitude
Time
t_1
t_n
Discrete time slices

Figure 2-6: Schema of beam time series backscatter with discrete amplitude samples in consecutive beams (*How-to-FMGT: FMGT Supported Data Formats* 2020).

however, depending on the specific data (manufacturer and model) it requires some more intermediate processing. The second approach is especially suitable for beam time series data which has been "formed specifically as to create a continuous trace along the seafloor when concatenated" (Schimel et al., 2018). Those data sets should be transformed into half-swath time series and georeferenced as such. Beaudoin et al. (2002) suggest to georeference half-swath time series with an improved slant-range correction based on two-way travel times and the across-track offsets of the beams per time series.

Radiometric corrections

As the active sonar equation (Equ. 2-7) shows, the recorded echo level has significantly been altered on its way to the seafloor and back. Therefore the raw backscatter data recorded by the MBES itself is not directly exploitable but first needs to be corrected for those undesirable dependencies. Radiometric corrections (step three) are meant to adjust the raw backscatter to some meaningful BS value which only depends on the signal acoustic frequency, the angle of incidence at the seafloor and the characteristics of the seafloor (Lurton, 2002). Schimel et al. (2018) classify three 'themes' of radiometric corrections:

Corrections for gains applied during reception

This correction addresses all modification which are applied to the backscatter level between signal reception and its recording in the raw data file. Typically, the analog signal from the receiver unit is amplified using a static gain and may optionally be further amplified by a dynamic gain to compensate for the transmission loss in the water column. Usually, further gains are applied after the analog to digital conversion. Often those gains are dynamically fitted to range, depth or other measurement parameters. The digital gains typically address the physical parameters of the sonar equation such as the ensonified area or angular dependence, and hence, are in principle desirable. However, especially the analog but also the digital gains are based on generic assumptions or estimated parameters from, for example, previous pings. That is why removing them and applying accurate corrections in post-processing might improve the result (Schimel et al., 2015).

Corrections for the interaction with the water column and seafloor

These corrections are the compensation for the physical interaction of the sound pulse with its environment, namely the transmission loss in the water column and the extent of the ensonified seafloor area. The two way transmission loss (Equ. 2-8) depends on the range to the seafloor and the absorption coefficient of the water. Theoretically, if the absorption through depth was known, its effect could directly be modeled for the path of the signal. But even though the absorption coefficient might be estimated from CTD measurements, it is still commonplace to use a single value approximated for an average depth in order to compensate for transmission loss (Schimel et al., 2018).

The compensation for ensonified seafloor requires the estimation of the area ensonified by the pulse for each time sample in the return signal. The area ensonified by an individual beam depends on the geometry of the beam itself but also the geometry of the seafloor. If the seafloor was flat (and horizontal), the areas ensonified by beams near normal incidence are said to be beam-limited, whereas beams at oblique incidence are pulse-limited (Brown et al., 2015). Beam-limited means that the whole area of seafloor within the beam is ensonified simultaneously, and hence, the along- and across-track beam width (beam pattern) determine the extent of the area. At oblique incidences, the sound pulse gradually ensonifies the area of seafloor covered by the beam. Then the extent of ensonified area is given in along-track direction by the the along-track beam width and in across-track direction by effective pulse length projected onto the seafloor (Weber and Lurton, 2015). While simple models use the assumption of a flat seafloor to determine the extent of the ensonified seafloor area, more advanced ones such as described by Beaudoin et al. (2002) also consider the geometry of the seafloor from the available bathymetric information.

Corrections for the mechanical properties of the transducer

Those corrections are supposed to compensate for the mechanical characteristics of the transducer. Included are corrections for the source level and the transmit and receive beam patterns. Information of the source level is usually available as a nominal value for different MBES models and modes of operation. It is typically compensated at an early stage along with the reception gains. However, the actual transmit and receive beam pattern is usually not known. Sometimes there is a combined transmit and receive beam pattern available which was tested on a prototype under laboratory conditions. Though individual sensors usually vary slightly from this prototypical behavior, especially as they age. To accurately compensate for the mechanical properties of the transducer a calibration would be required. Those calibration values could be applied in post-processing (Schimel et al., 2018).

Angular responses

So far the recorded amplitude values are geometrically and radiometrically corrected to a backscattering strength which only depends on the signal frequency, the angle of incidence at the seafloor and the seafloor characteristics itself. At this state of processing the angular response analysis takes place. The aim of analyzing the angular response is to correlate the seafloor characteristics to the backscatter variation with incidence angle (at a given frequency). Therefore backscatter and incidence angle couples are compiled over seafloor areas which appear to share common characteristics. Thereby the aim is to cover a wide range of the incidence angles. Afterwards two approaches to analyze the gathered information are possible, either a geophysical or an empirical. The former is based on geophysical models of angular responses which are fitted into the data. The latter approach is more pragmatic and relies on simple empirical models instead (Schimel et al., 2015).

Angle dependence removal

Stage four in the processing chain is the removal of the angular dependence of the backscattering strength. What is of interest in the angular response analysis, actually hinders the interpretation of backscatter mosaics as angle-dependent variations to some degree mask the seafloor-dependent differences. Yet this step is insofar delicate as the angular dependence itself hinges on the type of seafloor. Most often there is no prior knowledge of the seafloor type and its spatial variability available (hence the survey). Instead a procedure has to be found which overcomes the lack at best. Schimel et al. (2018) describes the standard technique comprised of two basic steps:

1. For each sample subtract the expected BS level according to the incidence angle.
2. Substitute a reference BS value corresponding to a reference incidence angle or interval.

There are two influencing factors which significantly determine the performance of the algorithm, namely how the expected BS values are derived and what angle or angular interval is used as reference. For typical seafloor types there are generic angular responses available, either measured or generated from physical models. But again, as usually the seafloor type is unknown, the more common approach is to determine the expected BS curves from the data itself. Therefore the angular response for a subset of data is computed. The larger the subset, the more likely statistical variations are smoothed out which would otherwise cause across-track artifacts. On the other hand, the probability increases that it overlays different types of seafloor, mixing up their angular responses which produces along-track artifacts (Schimel et al., 2018). The complexity of different implementations strongly varies from the choice of static subsets up to adaptive algorithms using moving window functions to optimize the expected angular response curve (Schimel et al., 2015).

On the other hand, also the reference BS level can significantly influence the result. While some implementations use relatively wide angle intervals between 20° to 60°, others suggest to use explicit angles 45° where the angular response differs most strongly between different seafloor types (Schimel et al., 2015).

Pre-mosaicing corrections

This step comprises possible operations which are meant to enhance the visual appearance of the final mosaic. Those corrections could optionally also be applied to the final mosaic. A desirable correction might be to downsample the backscatter data. Depending on the chosen backscatter type, the data might be present in a very high resolution exceeding the grid size of the final mosaic. To avoid aliasing effects it can be helpful to downsample the data to approximately the final grid resolution.

Another possible possible correction is to despeckle the backscatter data. Speckle noise is caused by stochastic fluctuations of the signal sampled at the transducer head (Fonseca and Calder, 2005). The removal of speckle can improve the interpretability of the backscatter data.

Mosaicing

In a last step the georeferenced and corrected backscatter is mapped into mosaics. The term comes from the time when backscatter data was delivered in individual images which had to be combined in a mosaic. Nowadays this step rather consists of gridding algorithms. Thereby special attention is paid on the choice of suitable grid parameters and how to combine overlapping data seamlessly (Schimel et al., 2018).

2.2. Kongsberg EM series data formats

Hydrographic raw data is typically delivered in a binary format with the data being structured in datagrams from the individual sensors such as the MBES itself but also external senors like the motion or position sensor. Each output datagram has a header including some supplemental information and the actual data body which is also called payload. Kongsberg EM series echo sounders write data in files with the suffix *.all. Each datagram includes two control characters, STX[7] and ETX[8], the datagram type, a time tag and a checksum. The date related information in all Kongsberg (EM series) data is given as 10,000 · year + 100 · month + day and the time is given in milliseconds from midnight. All information regarding the datagram format was extracted from the specification described in Kongsberg (2018).

MBES parameter

Some of the more generic information transmitted by Kongsberg MBESs are datagrams containing the system parameters. Among others they can be distinguished in runtime and installation parameters. The latter is stored in the eponymous datagram ‘installation parameters’. It is generally issued once when logging is turned on and also when it is turned off i.e., at the start and end of a survey line. For each sensor in the system the datagram logs the installation parameters. Among others it contains the installation offsets and angular orientation of each sensor, any time delays or the time system used, etc., but also some survey related information. Commonly, the installation parameters are automatically extracted by the post-processing software as required and will not be inspected by the operator. However, sometimes it may be necessary for troubleshooting purposes to manually check those parameters.

The information logged in the ‘runtime parameters’ describes how the MBES has been operated during the survey. It basically contains all the settings which are defined. This includes the different modes, possible filters applied in real-time, min and max depth, and the swath configuration such as the coverage on both sides of the swath. Some of the settings are thereby more relevant for the backscatter processing. These include the absorption coefficient applied, the transmit pulse length, the transmit and receive beam width, transmit power, and receive band width.

[7]Start of text character
[8]End of text character

Bathymetric data

Kongsberg delivers the bathymetric data in two different formats, the raw MBES data and some preprocessed data. The raw MBES data is written in ‘raw range and (beam) angle’ datagrams. There are minor differences between different EM series models but generally, the raw MBES data includes detailed information on the transmit sectors and reception beams. For each transmit sector of the ping there is information given about the characteristics of the transmitted signal such as the signal length, the center frequency, the band width, etc. For each receiver beam, the beam pointing angle, the associated transmit sector, some information about the detection, the two way travel time and an average reflectivity value are given (Kongsberg, 2018).

The preprocessed bathymetry data is depending on the MBES model either stored in the ‘depth’ or the ‘XYZ’ datagram. The two basically hold the same information but have nonetheless slight differences. The basic data contained in both datagrams is the depth (z), across-track distance (y) and along-track distance (x) for each beam. Thereby the beam data is given vertically from the transmit transducer and horizontally from the location of the active positioning system’s reference point. The beam data is already corrected for attitude (heave, roll and pitch) of the support platform, sound speed at the transducer depth and ray bending through the water column. To get the depth with respect to the water line, the transmit transducer depth at transmission time, which is also included in both datagrams, has to be added to the depth value. In contrast to the XYZ datagram, the depth datagram also contains the beam repression angle, the beam azimuth angle, and the range for each beam. Those values are only corrected for the support platform’s attitude and the sound velocity at the transducer depth and can thus be used for new ray bending calculations with revised SVPs without any need for attitude data. However, if the depth data needs to be processed with a new sound speed at the transducer depth, new attitude values or XYZ data is used instead, full reprocessing starting with the raw range and beam angle data is required. Additionally, both datagrams yield information about the quality, the detection window length and an average reflectivity value of each beam, though with slightly different implementations. While the depth datagram only contains beams with a valid bottom detection, the XYZ datagram generally includes data of all beams which is to be able to store backscatter data also for beams lacking a valid detection. The detection information of each beam is a 8 bit number indicating among other things the type of detection (amplitude or phase detect) for valid beams and the method used to determine the x, y, z values (interpolation, estimation, rejection or no data) for invalid beams. Additionally, a real time data cleaning module is applied to flag out beams (Kongsberg, 2018).

Backscatter data

Kongsberg traditionally stores two types of backscatter data, single value per beam data in the depth or XYZ datagrams (alongside with the bathymetric information) and beam time series data in the ‘seabed image’ datagram. Both datagrams use scaled units of decibel typically in 0.1 dB resolution. The single BS values are “an average value of the sample amplitude values. Short averaging lengths are used and the maximum average level within a beam is chosen to represent the beam BS” (Hammerstad, 2000). For very short echos, which might appear in shallow waters near normal incidence, the maximum sample amplitude is taken instead. The beam time series data provided in the seabed image datagram is “picked from the beam amplitude samples in such a way that when fitted together the total array of samples represent a continuous set along the bottom with a fixed interval in range according to the range sampling rate of the multibeam echo sounder and the mode it is used in" (Hammerstad, 2000). Attention is paid to layover effects and beams lacking a valid detection. The amplitude samples corresponding to the bottom detection are identified in the datagrams to allow for georeferencing.

Kongsberg applies one of the more complex dynamic gains (TVG) in the attempt to produce (radiometrically) corrected BS values. Thereby the applied gain is meant to compensate for:

- Static gain applied to maximize the dynamic range
- Dependence on the angle of incidence at the seafloor
- Mechanical properties of the transducer
- Transmission loss in the water column
    - Frequency dependent attenuation in the water column
    - Spherical spreading
- Ensonified seafloor area

Hammerstad (2000) describes the implementation of the dynamic gain. Cutbacks made to enable real-time processing include the initial assumption of a flat seafloor and the estimation of required parameters based on previous pings. To remove the angular dependence, a distinction is being made between small and large incidence angles. For larger angles ($> 25^\circ$) Kongsberg applies Lambert's law which (under the assumption of a uniform flat seafloor) describes the backscattering strength as angular variation of a mean backscattering coefficient BS_O. For smaller angles ($< 25^\circ$) a linear change is assumed from the mean backscattering coefficient at normal incidence BS_N to the coefficient at oblique incidence BS_O. The crossover angle between the two models was later found to be quite variable so that it now can be adjusted between 5° to 30°. Nominal values are used for the source level and the receiver sensitivity. After beam forming, the values are corrected for beam pointing angle dependent variations. Thereby the different frequencies of different transmit sectors (of deep water systems) are considered. Afterwards bottom detection errors in estimated parameters are corrected (Hammerstad, 2000).

External sensor data

Datagrams including data of external sensors are inter alia the 'attitude' datagram, the 'position' datagram and the 'sound speed profile' datagram. The attitude datagram includes roll, pitch, heave and heading information of the motion sensor. The data is not required if the preprocessed bathymetry data is used, but would be necessary for the raw data processing. The position datagram contains obviously latitude (negative if southern hemisphere) and longitude (negative if western hemisphere) but also speed and course over ground, heading and a quality measure of the position fix. It also includes a position system descriptor, indicating the source system of the position and the time used, either (MBES) system time or input datagram time. In contrast to the attitude data, the position data is in any case required to process the MBES data. It is noteworthy that the position and MBES sensors are asynchronous for Kongsberg EM series systems. Typically, the position (and attitude) sensor have a much higher measurement frequency as the MBES and need to be interpolated at ping time.

2.3. Jupyter Notebook and Python

On its official website Python is described as an “interpreted, object-oriented, high-level programming language with dynamic semantics” (PSF, 2020). To understand the full meaning of this loaded sentence the various components should be disassembled. The following information is based on Löwis and Fishbeck (1997).

An interpreted language, as opposed to a compiled language, uses an interpreter to parse the commands of a program and then executes them by sending the corresponding machine language instructions. On the other hand, compiled languages use a compiler which directly translates the program to machine code before execution. A compiled program will only work on the platform it was designed for, whereas interpreted programs can be executed on any platform if the suitable interpreter is available. Typically, interpreted languages are easier to debug and review, though they tend to be slower.

In Python the main programming paradigm is object-oriented. An object is a data entity which may have metadata in form of attributes and associated functionality called methods. Other programming approaches include for example functional programming which in comparison emphasizes on functions. While different programming approaches have different strengths and weaknesses, at some point it becomes rather a question of personal preference what approach to use.

A high-level programming language is a language with a high degree of abstraction from the actual machine instructions towards human language. On that scale, a low-level language would rank at the opposite end, with a lower degree of abstraction. The abstraction itself eases the use of a language and may automate computing system tasks as for example memory management.

The PYPL[9] index asses the popularity based on how often language tutorials are searched on Google. According to this index, Python is not only the most searched language but also the language whose popularity grew the most in the last 5 years with 18.4 % (Carbonnelle, 2020, status as of December 2020). On the globally popular development platform Github in 2019 Python had ranked on place two of the most popular languages after the first placed language JavaScript and just ahead of Java which has for a long time been placed second. When Python was for the first time released, it was welcomed since it needed comparably fewer code to express the same functionality as compared to Java. This improved the efficiency of developers and provided code with a good readability (Skywell Software, 2020). On the other hand, Java was very popular for its free runtime on popular platforms and an enhanced, configurable security.

According to Skywell Software (2020), one of the biggest differences is that Java in contrast to Python is a statically typed language, meaning that variables have to be declared with a specific type. As a consequence, Java code is often wordier and therefore less readable and efficient. Another difference is that Python compiles implicitly on-the-fly while Java has to be compiled in advance. Consequentially, Java code tends to be faster than Python. However, especially for non-professional programmers, the readability and efficiency of code implementation in Python is comparably more productive than the faster code execution in Java. Simply speaking, programs which are written in Python are developed faster. According to inVerita (2020) Python is by far the most popular

[9]PopularitY of Programming Language

language for data science and machine learning, since 83 % of the data professionals use Python. This introduces another aspect that should be considered: The open source community for Python in data science is huge. For many different applications, there are already libraries, packages and/or modules to solve the task. Furthermore, Python can be used to extend and embed other languages such as C or C++. In this context, Python can be understood as a glue language. Performance-critical parts of a program can then be written in or adapted from the faster languages, while Python is used for control and customization (van Rossum, 1998).

A development environment generally includes the procedures and tools for building a program. The term is often used as a synonym for an integrated development environment (IDE), which is the software development tool used to write, test, and debug a program. The most basic IDE would be a simple text editor, while more advanced IDEs provide the programmer with a graphical user interface to aid in development. One of them is the Jupyter Notebook. It is the graphical user interface to the IPython (Interactive Python) shell which itself is meant to extend the Python interpreter in terms of interactivity, and exploratory data-intensive computing (Perez and Granger, 2007).

> “The notebook extends the console-based approach to interactive computing in a qualitatively new direction, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results.” (Jupyter Team, 2015)

A notebook has various dynamic display capabilities which not only allow the execution of Python/IPython statements but also inter alia the integration of formatted text, static and dynamic visualizations, mathematical equations, JavaScript widgets, etc. (Vanderplas, 2017). All of those elements are saved within the notebook document and can thus be easily shared and distributed. In order to execute code the notebook must be connected to a kernel. A kernel can be understood as a program that runs and introspects the code of the user. IPython includes a kernel for Python code, but there are meanwhile also kernels for several other languages in Jupyter Notebook available.

The Anaconda distribution is the recommended way to install Python for use in data science, partly because it is operating system independent. There are two variants to use it: Miniconda, which includes the Python interpreter and the conda package (and environment) manager, and Anaconda, which additionally includes a bundle of pre-installed Python packages suitable for scientific computing.

3. Methodology

The Kongsberg EM 122 is a deep water MBES system with a nominal frequency of 12 kHz and an angular coverage sector of up to 150°. Over a flat seafloor, the swath width can be up to six times the water depth. During RV Sonne cruise SO268-3 the swath opening angle was set from 130° to 140° which roughly resulted in a swath width of around 22 km in 5,000 m water depth. The separate transmit and receive array of the EM 122 are both linear arrays set up in a Mills cross configuration. The model deployed on RV Sonne (Fig. 3-1) has a beam width of 0.5° x 1°. The transmit fan is split into different sectors which are independently steered according to the vessel roll, pitch and yaw. The soundings are placed on a line perpendicular to the survey line ensuring a uniform sampling of the seafloor. Normally, the sounding spacing is equidistant but an equiangle mode is available. In high density equidistant mode the detections are derived from more than one point within a beam, resulting in up to 432 soundings per swath. In dual swath mode, two swathes with a slight alongtrack tilt are transmitted per ping recording up to 864 soundings per ping. The tilt applied is chosen in consideration of depth, coverage and vessel speed to ensure a constant sounding separation alongtrack. The ping rate is mainly determined by the two way travel time of the pulse in the water column but is limited to ca. 5 Hz (Kongsberg, 2011; Kinne et al., 2019).

Figure 3-1: RV Sonne (www.bmbf.de)

The EM 122 logs two types of bathymetric data: ‘Raw range and angle’ and ‘XYZ’ depth datagrams. As outlined in section 2.2, the latter can be understood as a preprocessed version of the former as it already provides (vessel) coordinates per beam instead of beam opening angle and pulse travel time information. The x, y, z coordinates are already corrected for vessel attitude (heave, roll and pitch), sound speed at the transducer face and ray bending through the water column. The choice of which of the two datagrams is initially used, essentially determines what kind of corrections may be applied in postprocessing. IHO (2008) recommends to correct the raw data of each sensor in the MBES system (position, motion, SVP and MBES) individually before merging the corresponding data. Those corrections are only possible when using the raw beam data and the other respective raw datagrams. However, the processing implementation is very complex and tedious. Assuming that the data acquisition has been conducted diligently in a sense that any potential sensor misalignment or time offset has been properly checked and calibrated in advance and enough sound velocity profiles have been taken underway, it seems justified to assume that the real-time corrections for vessel motion and sound velocity applied to the depth data are sufficient. Hence, despite the limited possibility for postprocessing corrections, the XYZ datagrams are used as starting point for the bathymetry processing. To accommodate data where the above stated assumption is violated, the bathymetry processing and (data) cleaning are kept separate with an open ASCII interface. That way, any necessary corrections can be for now applied separately, while the (raw) bathymetry processing can be implemented at a later point in time.

A similar decision is faced for the backscatter data. It is also logged in two different formats: Single value per beam reflectivity is stored along with the bathymetry in the 'XYZ' datagram and beam time series data is separately stored in the 'seabed image' datagram. Since it is the more common choice when working with backscatter data, the beam time series data is used as starting point for the backscatter processing. While the single value per beam samples are georeferenced by their associated bathymetry information, the seabed image data requires an additional processing step to georeference all samples of the series. Generally, Kongsberg applies a quite complex TVG correction to the raw backscatter which is meant to radiometrically correct the value to some meaningful level related to the seafloor scattering properties. However, to correct the data in real-time, some simplifications have to be presumed which downgrade the accuracy of the applied correction. To achieve the optimal outcome, the TVG would have to be taken out in postprocessing and re-calculated using the available bathymetric information. As those computations can become very comprehensive, they are not attempted within this thesis.

3.1. Available tools and algorithms

The initial idea is to find a suitable combination of available open source libraries to facilitate the required processing of multibeam data. As mentioned previously, Python was chosen in part because of its large open source community, particularly in the data science (and machine learning) community. When looking for customizable libraries, it is important to also consider licensing. Open source in the software context means that the source code is open for inspection, modification, and extension. Depending on the specific license, modification and distribution may be subject to conditions, such as that the modified source code must be made available to the community (*What is open source?* 2020). In the search for suitable libraries, the licensing did not really result in any major restrictions. All of the tools mentioned below are published under an OSI[1] approved license (Open Source Initiative, 2020).

In examining the available tools and algorithms themselves, it was found that the point cloud processing community is heavily rooted in LiDAR[2]. The hydrographic community was not extensive, at least on GitHub. However, there are contributions: HydroOffice in particular seems to provide open source code for water-related geoscience, though there was no real overlap with the objectives of this work. Some fairly extensive Matlab scripts were also found, some of which even dealt with MBES data processing. Unfortunately, these were not compatible with the structure of the thesis. The libraries and modules that were finally selected are briefly explained below. While more than the mentioned Python libraries are used in the thesis, the less essential ones are introduced in the

[1]Open Source Initiative
[2]Light Detection and Ranging

respective section of the text.

pyALL

pyALL is a Python module which reads Kongsberg ALL files. It was written by Kennedy (2016) and published under the Apache license. Files in the ALL format are binary, hence they cannot directly be read. For that purpose pyALL uses the struct module of Python which converts between strings of bytes and Python native data types. The pyALL module is built up on an ALLreader class. The class has among others a method which reads the header of each datagram in a file. If the header corresponds to a datagram which is intended to be used, the according datagram is passed for reading. Each datagram type is thereby associated with an individual class. The datagram classes have each a read method which finally decodes the binary string corresponding to the datagram dependent class attributes. To illustrate the workflow it is exemplary assumed that the task is to extract all position datagrams from a given ALL file. As a first step, the ALLreader is initiated. While there are still datagrams in the file, the reader decodes the header of each datagram. If the header information corresponds to the position datagram, the still encoded datagram is passed to the initiator of the position class. By calling the position class's read method, the datagram is finally decoded and the values are stored as position class attributes (of that specific position class instance) according to their Python data type. That specific position instance can then be further processed in custom code. pyALL supports all the required datagram types used by Kongsberg and furthermore provides some handy function. One example is a time helper function which converts the Kongsberg date and time format into Python's datetime format.

PDAL

PDAL (Point Data Abstraction Library) is a C++ library for point cloud processing released under the FreeBSD (*PDAL* 2019) license. Its concept is based on modules that can be individually chained together in pipelines to achieve a particular point cloud processing task. Since it has its origins in LiDAR processing, many of the underlying modules are based on the basic principles and requirements of LiDAR. Unlike an alternative, monolithic approach, it has the main advantage of being able to cover many more different situations by simply adjusting the modules and sequence, rather than rewriting the entire program until it can solve that one, very specific task. In a sense, PDAL can be seen as the point cloud data equivalent of GDAL (Geospatial Data Abstraction Library), intended for processing vector and raster data. Since vector/raster and point cloud data are all kinds of geospatial data, they share a common basis. Point cloud data, however, typically has an even larger volume of data that requires "specialized processing and management techniques [...] to handle so much data efficiently" (PDAL, 2020). As briefly mentioned earlier, for this purpose it is also advantageous to offload performance-critical code to C++, as this can usually be executed faster. For this, PDAL offers possibilities for embedding as well as extending Python.

Comparing PDAL to other available point cloud processing tools, there are some generic features where PDAL is particularly well suited to the intent of this thesis: First, it can be freely used, redistributed, and modified. In addition, it is not exclusively based on the LiDAR point cloud format LAS, which other point processing software from the LiDAR community partially are. In addition, its clear pipeline structure allows for the rapid construction and modification of different processing chains. Much of the other open source point cloud software (CloudCompare, libLAS, etc.) is more desktop

GUI oriented than library oriented. PDAL, on the other hand, allows an (experienced) programmer to customize and extend the functionality of the library.

Before ultimately deciding on PDAL for the outlined task, two other libraries were tested: PCL[3] and Open3D. PCL (Rusu and Cousins, 2011) appeared to be more specific to the range of steps required for the processing chain of point clouds with photogrammetric or LiDAR origins, such as registration, filtering, segmentation, and visualization. PDAL attempts to work with the point clouds themselves, regardless of their origin. Similar observations have been made with Open3D (Zhou et al., 2018). It also focuses a lot on point cloud registration, segmentation, etc. It was found that Open3D has a very convenient 3D visualization tool. However, in the end, PDAL, especially in combination with the tools mentioned below, proved to be the most suitable for the purposes of this thesis.

Entwine

Entwine (Manning, 2016) is a data organization library designated to point cloud data of large volumes. For this purpose it uses a specific indexed structure, the Entwine Point Tiles (EPTs). Entwine is closely linked to PDAL and vice versa PDAL supports the Entwine data format. EPT is a “ simple and flexible octree-based storage format for point cloud data” (Hobu, 2020). An octree is a hierarchical tree data structure in which each node has exactly eight children or non. To index a point cloud with an octree, the 3D cubic boundary (root node) is divided into eight octants of which each is further subdivided into octants until a certain threshold is met. Typically the threshold is defined as the maximum depth or the number of points (Han, 2018). An example of an octree structure can be seen in figure 3-2. The yellow boxes indicate the voxels at the different depths (levels). The outermost yellow box is the root node.

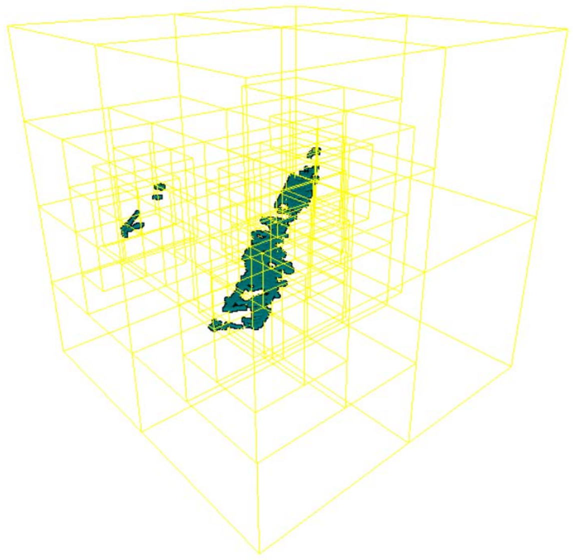


Figure 3-2: Potree viewer of an example data set (https://potree.entwine.io). The yellow boxes indicate the octree levels.

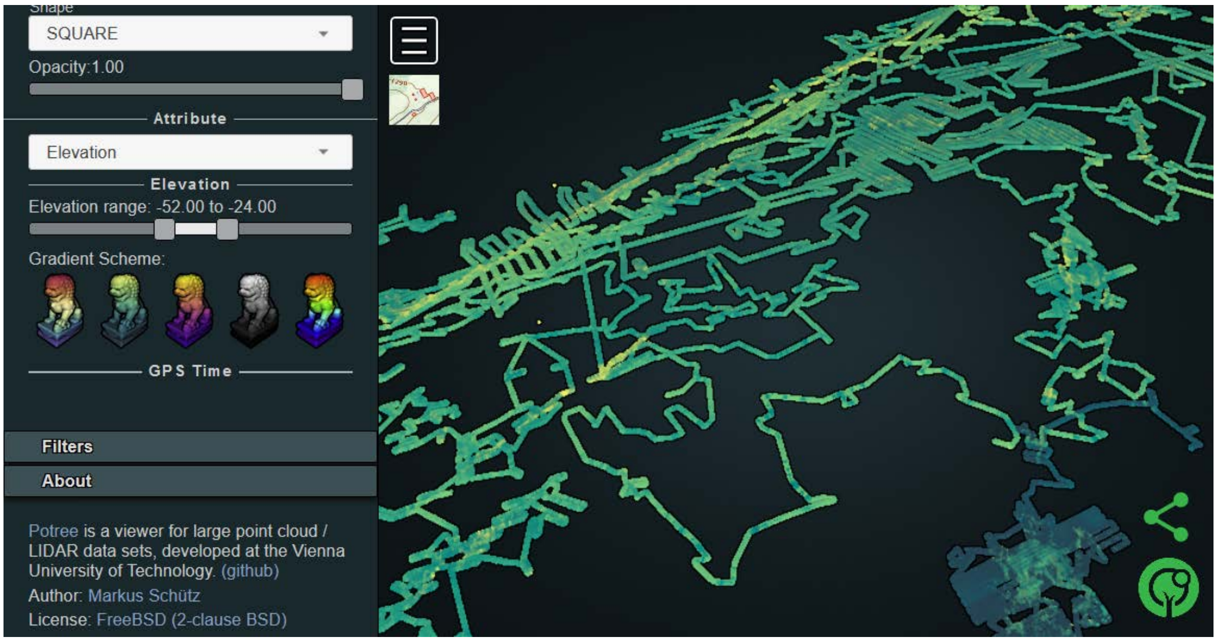
An EPT data set consists of metadata in JSON and the structured binary point data. A specific description of the individual files can be seen in Hobu (2020). Entwine uses similar mechanisms as

[3]Point Cloud Library

the ones applied in raster tiling schemes but extended to 3D. In raster tiling schemes, the coarse resolution data is typically exchanged by higher resolute data when traversing its sub-tiles. EPT in contrast can be understood as an additive scheme (Hobu, 2020). Rather than replacing data, it adds data at a higher resolution. Octrees are generally popular for its "memory efficiency, query speed and structural simplicity" (Han, 2018).

Potree

Potree (Schütz, 2016) is a web-based point cloud renderer that allows users to view large point could data sets in a web browser. While raw point clouds are often converted into triangle models, i.e. meshes, or two-dimensional images/grids for visualization purposes, Potree directly visualizes the points. Based on a modifiable nested octree structure in the background, distant regions are rendered at a lower level of detail. Additionally, points outside the field of view are culled. By this means even large point clouds can be rendered at a fast speed. Potree itself is based on WebGL[4] which is a JavaScript API for rendering high performance interactive graphics. Among others WebGL takes advantage of hardware graphics acceleration, e.g. visualization processes can be offloaded on a graphics card (MDN contributors, 2020). With WebGL, the distribution of 3D content over web browsers has become increasingly popular. Especially as it allows to visualize content without the need to install a third-party viewer. Potree is an application entirely on the client side. The server only hosts files i.e. JavaScript configurations but does not execute any code. To use Potree, a point cloud first has to be converted to the octree structure. While Potree directly offers a converter, called PotreeConverter, also an Entwine EPT data set can be used to provide the structure as it is itself a octree-based storage format. This data set is then locally served and the Potree configuration fetched from the Potree server by opening the special Potree URL which allows to take the localhost URL. The Potree user interface can be seen in figure 3-3.



Potree is a viewer for large point cloud / LIDAR data sets, developed at the Vienna University of Technology. (github)
Author: Markus Schütz
License: FreeBSD (2-clause BSD)

Figure 3-3: Potree viewer of an example data set from the search for the Malaysia Airlines Flight 370 (https://potree.entwine.io)

[4]Web Graphics Library

On the sidebar (left side), Potree additionally provides tools for the exploitation of and measurements in the point cloud. Among others, one can choose different measurement tools such as angle, point, distance, height, area, volume etc. Also it is possible to clip specific areas of the point cloud and change the visualization in terms of colorization, point size, etc.

3.2. Project structure

The aim of this thesis is to implement a project structure which enables an adaptive workflow in Python. The workflow can be subdivided in three tasks:

1. Compute the bathymetry point cloud from the raw Kongsberg beam data (Bathymetry preprocessing).
2. Use the bathymetry to georeference the raw backscatter time series (Backscatter preprocessing).
3. Filter and correct the data to clean undesired data artifacts (Data filters and corrections).

The outlined tasks are approached based on the RV Sonne SO268-3 cruise data. The cruise started on the 30/05/2019 in Vancouver, crossed the Northern Pacific westwards and ended at the 05/07/2019 in Singapore. During the cruise, the MBES has constantly been operated and the data saved in Kongsberg raw data files (*.all) of 60 min duration. Underway, eight CTD measurements were used to apply sound velocity corrections to the data. Before the first CTD station, synthetic profiles were extracted from Sound Speed Manager (Kinne et al., 2019). The outlined tasks are subdivided in individual steps as shown in appendix A. Starting with the raw Kongsberg EM 122 data, in a first step the needed datagrams have to be decoded. For this purpose, the pyall module is used. Datagrams found to be of relevance are the (P) position datagram, the (X) XYZ datagram, the (Y) seabed image datagram and the (R) runtime datagram.

Bathymetry preprocessing

Subsequently, the raw positions and depths are used to compute the bathymetric point cloud. Since the Kongsberg datagrams are not synchronized, the latitude and longitude of the position datagrams first need to be interpolated to ping time. Afterwards the MBES position at ping time and the transducer depth are merged with the beam coordinates to transform them from the vessel to a global geographic coordinate system (WGS 84). This transformation requires a solution to the direct geodetic problem which is integrated in Python's geographiclib (Karney, 2015). The final result of this step and the first task is a bathymetric point cloud in geographic coordinates. The implementation is described in section 3.3.

Backscatter preprocessing

To solve the second task, the raw seabed image data and the previously derived bathymetry are required. The seabed image data contains time series of backscatter sample for each beam which have to be georeferenced. Therefor the bottom detection sample of each series has to be identified and associated with the corresponding sounding of the bathymetry. The other samples are then concatenated and interpolated between two adjacent bottom detections. The result of this step and the second task is a dense point cloud attributed with the backscatter amplitudes. The implementation is described in section 3.4.

The data stored in the runtime datagram is from a processing point of view mainly interesting for the backscatter data. Different survey settings can have a major impact on the radiometric level itself but also on the geometric distribution of the samples along the swath. There are approaches which thoroughly address the individual settings in an attempt to automatically adjust the backscatter processing. However, since a radiometric correction is not implemented in this work, those settings are mainly interesting to understand the data and potentially for future implementations. On the other hand, the important settings for the georeferencing did not change throughout the cruise and hence their impact could not be detected.

Data filters and corrections

The third task relies heavily on the strengths of Entwine, PDAL and Potree for both, the bathymetry but also the backscatter processing. During the thesis, it was found that the data amount of point clouds exceeds the capability of simple Python functions. For example, even finding the nearest neighbor of a point becomes a lengthy task when working with unstructured point clouds on a point by point basis. The combination of the above named libraries addresses exactly this need for efficient point cloud handling. Entwine is a data organization library meant for massive as well as desktop-scale point clouds. It can essentially index anything that is PDAL readable and it is completely lossless in terms of points, metadata and precision. The Entwine EPT[5] is an octree-based storage format based on structured binary files in the ASPRS[6] LAS format. It was found that PDAL point cloud processing from EPTs is very fast compared to e.g. the same processing with ASCII files. In the documentation of PDAL it is said that the PDAL EPT reader supports “spatially accelerated [...] and file reconstruction queries” (PDAL Contributors, 2020). On the other hand, Entwine is suitable for real-time rendering in Potree. Potree is a web-based point cloud viewer which, just as PDAL and Entwine, emerged from endeavors of the LiDAR open source community. LiDAR typically does not focus on hydrographic data (even though there are bathymetric LiDARs), therefore some of the basic constructs do not work perfectly for the given data set and need workarounds which are not the most elegant. For example, the predefined object classes in LAS and Potree include trees, street furniture etc. which are not needed underwater. However, since LiDARs, just as MBES systems, attempt to build models of real world features, there is a consent of what processing tools should be capable of such as outlier filtering. Also, all the tools have the general flexibility to be adjusted to the user/developer needs. They can be tailored to the personal demands, if it is considered worth the work it takes to get into the various components involved in the process.

The bathymetry and backscatter data is converted to the Entwine format. As PDAL works in units of the data coordinate system, the point clouds have to be projected to UTM. Subsequent filters and corrections for data cleaning are then applied in PDAL using Potree as supervision tool. Thereby the Python extension of PDAL is planned to be used to conduct the processing. For the bathymetry cleaning, a fixed sequence of filters is applied to the data. A Jupyter Notebook is used to substitute the individual filter parameters to the data. The suitability of the parameter settings can be verified in Potree. Once the bathymetry has been sufficiently cleaned, it can be exported as an ASCII point cloud or in gridded form. The backscatter processing approach is structured similarly. The corrections are also parameterized in the notebook and then exported to ASCII or grids.

[5]Entwine Point Tiles
[6]American Society for Photogrammetry and Remote Sensing

3.3. Bathymetry processing workflow

For each beam in a XYZ datagram there are Cartesian vessel coordinates x, y, z given which are vertically referenced to the MBES transmit transducer and horizontally to the active position system's location. The test data set contains position data from two systems which are specified by the 8 bit descriptor field:

11xx xx11	The first to digits indicate that the position system is active, using the input datagram time and the last two that the input data came from position system number 3.
xxxx xx01	The last two digits indicate that the input datagram came from position system number 1.

That means although both position systems are sending data to the MBES system, the beam data is only referenced to system 3. Figure 3-4 shows a section of the position data of both systems with the same start- and end-time, whereby the ship proceeds from the lower right to the upper left corner. While the forms of the respective tracks correspond, there is a constant spatial offset whereby position system 1 is further ahead.

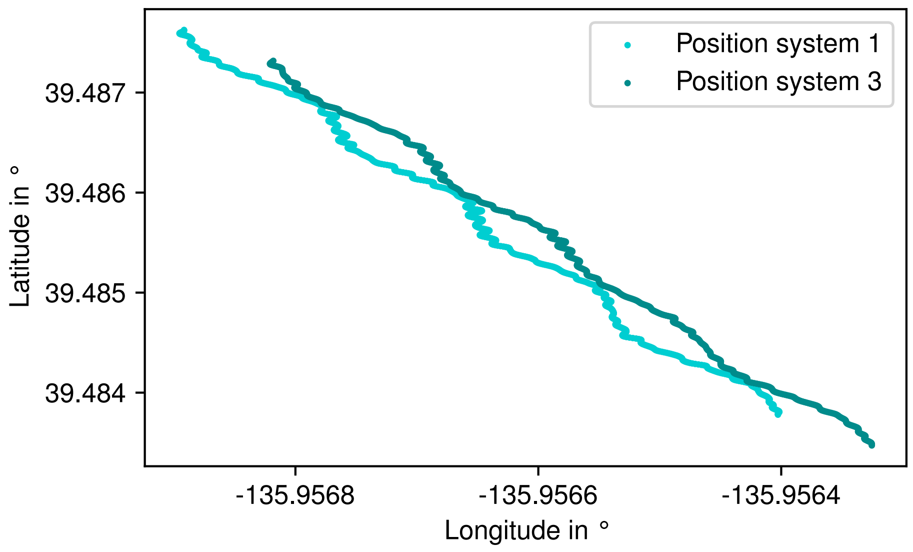


Figure 3-4: Spatial offset between position system 3 (active) and 1

And indeed, the installation parameters of the two position systems (Tab. 3-1) show that the location of position system 3 coincides with the origin of vessel coordinate system whereas the additional position system is horizontally located roughly 35.1 m further to the bow of the ship and ca. 2.6 m to the port side.

Inspecting the position data itself, it can be seen that the data from system 1 and 3 arrive in an alternating sequence whereby two subsequent datagrams share the same position counter. Meaning that after each system 1 position datagram, there follows a system 3 datagram with the same position counter. The time difference between a 1-3 datagram sequence (with the same position counter) is 311 ms and it repeats after 189 ms.

As the MBES data is referenced solely to the location of the active positioning system, the data of the other system can not directly be used. A straightforward approach to solve the ambiguity is to use

Table 3-1: Installation parameters of position systems

	System 1	System 2	System 3 (active)
Quality check	on	on	on
Motion compensation	on	off	off
Time stamp used	input datagram	system	input datagram
Vertical location	–15.604 m	0.000 m	0.000 m
Along location	35.119 m	0.000 m	0.000 m
Athwart location	–2.607 m	0.000 m	0.000 m
Time delay	0.000 s	0.000 s	0.000 s
Geodetic datum	WGS 84	WGS 84	WGS 84

data from the active positioning system only. When skipping the system 1 data, the measurement frequency of system 3 would still be 2 Hz, and hence, the maximum time lag possible between a depth and position datagram would be 0.25 s. Usually the depth measurement frequency is much lower as sound can only travel ca. 375 m through water in the same time which corresponds to a depth of 187.5 m (actually less, since the outer beams have a longer way due to the swath geometry). Typically it would be even less through hardware induced time delays. For the section of test data with a mean depth of roughly 5,000 m and 24 s per measurement (for a dual swath to repeat), the position measurement frequency would be sufficient. Though it has to be kept in mind for shallow water systems.

To allow a generalized distinction of position data from the active and potential other position systems, the 8 bit descriptor needs to be utilized. As shown above, it mainly indicates the source position system number of the datagram and for the active system also which time has been used. But regardless of the system number and time, the first digit of the code has to be 1 for the active position system. Thus, in order to identify whether a datagram comes from the active system, only the first digit of the descriptor needs to be checked. Or if decimal numbers are used instead, the number has to be > 128 as all higher numbers will definitely start with a 1 when converted back to binary.

3.3.1. Interpolation of positions to ping time

Another complication when working with Kongsberg MBES data is that the sensors and hence their data are asynchronous. At ping time of the MBES transmit transducer there is (probably) no instantaneous position measurement available. Therefore, after filtering the position datagrams for the active system data only, the positions need to be estimated at MBES ping time. For this purpose, latitude and longitude are independently interpolated as functions of position datagram time and then estimated at XYZ datagram time. In an attempt to do so, it turned out that for all of the 670 files of the test data set, the position datagrams started with a time lag of on average 20 s compared to the XYZ datagrams. Likewise, the time lag at the end of each file was on average 40 s. As the position descriptor states, the timestamp of the position datagrams was taken from the input datagrams. The general MBES system time, following the system clock information given in the installation information, had as clock source the NMEA ZDA format datagram and a 1 PPS clock synchronization using the rising edge detect was turned on. Following Kongsberg (2018), those are suitable settings when the timestamp supplied in the position input datagrams is used. This suggests that the time lag did not arise from an actual delay of the position and MBES sensor but is rather caused by how the data is fed into the data file. At a later point it was found that this is a typical phenomenon of Kongsberg deep water echo sounders. Consequentially, following challenges arise which the choice of the interpolation method has to take into account:

- How to handle data at the beginning and end of a file?
- How to handle potential position data gaps?

Actually both questions are insofar associated, as the time lag at the beginning of a file can cause an artificial data gap. It is difficult to find a universal solution as the reasons for time gaps are various. For example the position sensor could have a temporal malfunction or its data transmission is interrupted, or the different sensors have time lags which cause systematic deviations. Generally, one should be careful to directly interpolate large numbers of consecutive ping positions as it indicates a somewhat larger data gap. Therefore, an interpolation limit is introduced which restricts the maximum number of positions that are interpolated between to position measurements. For dual swath system as the Kongsberg EM 122 used for the test data set, the value should not be lower than 2 as the two swathes of a ping are received in a very short period of time and thus the limit would be reached with almost every ping. If the source of the data gap is known and rated as 'non-critical', the limit can manually be raised. To handle systematic effects arising from actual time lags between the position sensor and MBES system, it should also be possible to apply a manual time offset (positive or negative) to the position data before the data is interpolated to ping time.

3.3.2. Determination of sounding positions

After the interpolation of positions to ping time, the successive task is to combine the position and MBES (beam) data to determine latitude φ, longitude λ and depth from the waterline of each sounding. For that purpose the XYZ transducer depth at ping time has to be added to each beam depth z, and the horizontal coordinates x, y of each beam have to be transformed from the local Cartesian vessel to the global geographic system. The latter requires a solution to the direct geodetic problem which deals with the determination of the geographic position of a point by using the distance and azimuth starting from the position of an initial point. On an ellipsoid (as the GNSS reference ellipsoid WGS 84) the shortest distance between two points is a geodesic. However, a common approximation for such kind of navigational problem is the combination of local flat and ellipsoidal or spherical calculations. In a first step, ranges, bearings and/or courses are converted to ΔX and ΔY increments in a local rectangular coordinate system with the y-axis pointing north. Afterwards, ΔX and ΔY are converted to geographic coordinate increments $\Delta\varphi$ and $\Delta\lambda$ on the reference ellipsoid. These can then be added to the geographic coordinates φ and λ of the initial point (Lenart, 2011). While this approach is a computationally efficient method, the distortion caused by the approximations was strongly visible as artifacts in the bathymetric data which was significantly skewed. An often used solution which uses accurate geodesic calculations is Vincenty's formula. It is an iterative method which is meant to cover the entire range from very short geodesics in the centimeter range up to 20,000 m with a reasonable computational effort. Inaccuracies were found to be in the millimeter range (Vincenty, 1975).

To transform the vessel coordinates to geographic coordinates, Vincenty's formula is applied to each beam in the swath. The interpolated MBES position at ping time is used as starting position for the direct geodetic problem. Subsequently, the distance and azimuth to each beam of the swath are calculated. Thereby, the distance can be computed by applying Pythagoras's theorem to the x, y beam coordinates. The azimuth is calculated by adding the atan2 function of y and x to the heading of the swath. Together, the MBES position and the distance and azimuth of each beam can be inserted in Vincenty's formula to determine the geographic coordinates of the swath.

3.4. Backscatter processing workflow

As described earlier, the Kongsberg beam time series data is formed in a way that the sample sets of each beam form together a continuous trace along the seafloor when concatenated. This implies that samples overlapping at the edges of two beams are already resolved. The two generic approaches to georeference the samples are either to georeference the corresponding sample of each beam with the bottom detection and then interpolate the others or secondly, to transform the data into half-swath times series and proceed with the data as such. In Schimel et al. (2018) it is recommended to use the second approach for this type of continuous beam time series data. Following e.g. the methodology of Beaudoin et al. (2002), an across-track array is formed (with the desired spatial resolution) and then filled with the according two way travel times of the corresponding beams. The time index is subsequently used to populate the array with the amplitude samples. This approach was initially used by the OMG[7] software library to process data from Atlas systems and was then modified by Beaudoin et al. (2002) to adapt SeaBat data. However, due to the form in which Kongsberg provides the seabed image data, the first approach seems to be more obvious as there is no two-way-travel-time available but the center sample number. To asses the feasibility, following information should be acquired:

1. How to find the amplitude sample corresponding to the bottom detection?
2. How are the samples spatially distributed within the beam?
3. How are XYZ and seabed image data associated?
4. How are beams handled without a valid bottom detection?

In Kongsberg (2018) the structure of the seabed image datagram and some supplementary information is described. For each beam, there is a center sample number given which corresponds to the detection point of the beam. When inspecting an arbitrary file of the test data set, it can be seen that the center sample number does not necessarily correspond to the actual center of the sample series in a sense that it coincides with half the number of samples per beam.

Hammerstad (2000) describes that the continuous set of samples along the bottom is spaced in fixed intervals according to the range sampling rate of the MBES and the mode it is used in. It should be possible to derive both parameters from the information given in the runtime datagram. The mode logged in the runtime datagram was changed a couple of times throughout the acquisition of the test data set (Tab. 3-2). It generally describes different settings for the dual swath mode, the TX pulse form and the ping/depth mode of which all are meant to optimize the measurement for the local water depth.

The dual swath mode is meant to increase the along track resolution in deeper waters by transmitting two swathes per ping whereby the transmit beams are slightly tilted in along-track direction.

[7]Ocean Mapping Group of the University of New Brunswick

Table 3-2: Different modes used in the test data set.

Dual swath mode	TX pulse form	Ping mode
dynamic	CW	medium
off	CW	deep
dynamic	CW	deep
off	mixed	deep
dynamic	mixed	deep

However, it is only available in waters deeper than 50 m to ensure enough separation between TX pulses (Kongsberg, 2011). In dynamic (dual swath) mode, the tilt angle between the two swathes is automatically determined based on vessel speed, ping rate and depth in the attempt to provide a uniform along-track resolution (Kongsberg, 2019). The TX pulse form is meant to extend range where required by water depth. The two available pulse forms are either continuous wave (CW) or frequency modulated (FM) chirp. There is also a combined mode, where the outer transmit sectors use a significantly longer FM chirp to optimize the signal-to-noise ratio. However, those modes do not seem to have an impact on how the amplitude samples are distributed in across-track direction.

During acquisition of the test data set, the beam spacing was constantly set to high density equidistant which is essentially a signal processing technique applied to control the effective acoustic footprint and keep it constant for all soundings derived from phase detections (Kongsberg, 2011). In fact, this is valid for almost all soundings except a few at normal incidence which are usually detected by amplitude. Figure 3-5 shows the average distances determined based on x, y (2D) and x, y, z (3D) differences of adjacent beams along the swath. When comparing the 2D and 3D plots, the former seems to fit better to the above described spatial spacing of the footprints. Around the center beams, the distances are shorter with higher standard deviations which could be caused by the amplitude detection. Also on the outer beams, the standard deviation gets worse.

Figure 3-6 shows the average number of amplitude samples grouped by their corresponding beam index. It can be seen that the number of samples per beam increases the further outside a beam is. For the two utmost beams at port and starboard side, the values suddenly rise to values above 900 samples. When considering that the effective acoustic footprint for most of the beams is processed to be constant, it would actually imply that the sample density increases towards the outer beams.

Though as mentioned above, the center sample number does not always correspond to half the number of samples per swath and this could actually be the reason, why the samples are spaced equally, despite the increasing number of samples per beam. While the effective beam footprint is processed to be constant, the actual beam form is very likely still equiangular according to the RX beam width. That is why the actual (across-track) footprint increases and hence the number of equally spaced samples. Consequently, when matching the amplitude samples which spatially coincide with the (real) equi-angular footprint and the (effective) equidistant sounding/detection points, the bottom detection sample would relatively move to a position closer to the swath center, the further outside a beam is.

XYZ and seabed image datagrams belonging to the same ping/swath are associated by a common sequential ping counter. In both datagrams, there is data given for each beam of the swath including

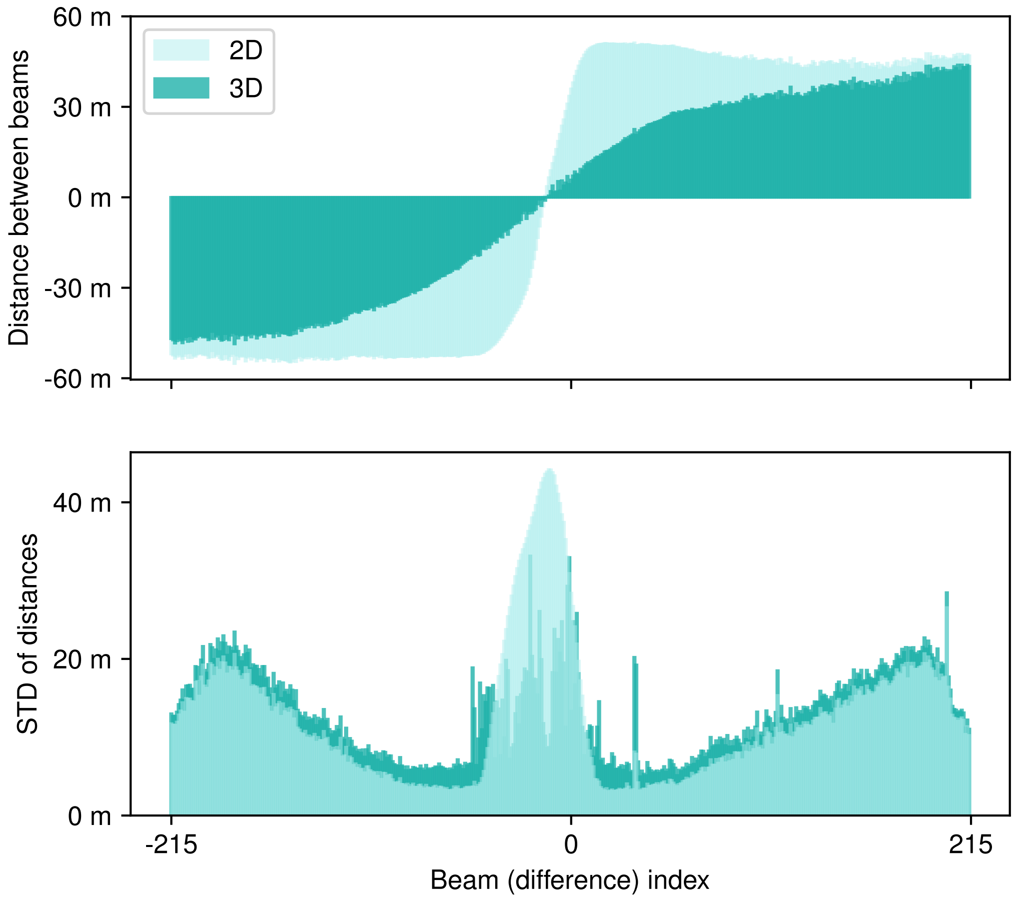


Figure 3-5: Average 2D and 3D distances between adjacent beams based on an arbitrary file (upper figure) and the corresponding standard deviation STD (lower figure).

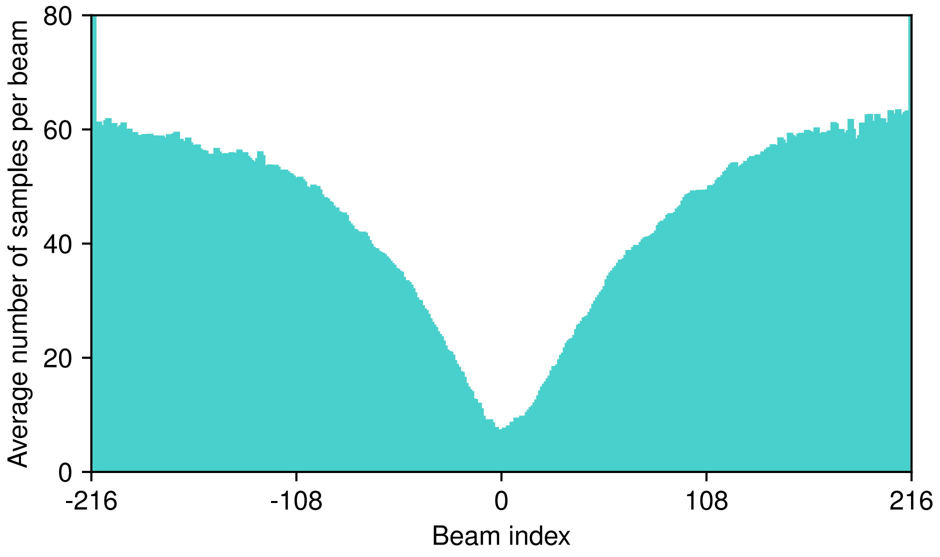


Figure 3-6: Bar plot of average number of sample amplitudes per beam along the beam index. For the outer beams at beam index -161 and 161 the values are rounded 993 and 987.

beams lacking valid detection, and hence, the respective beams in the different datagrams can be allocated by generating a beam index. For the 100,260,288 beams in the entire data set, following detection types were found:

- 85.87 % valid phase detections
- 7.90 % valid amplitude detections
- 6.23 % invalid but inter-/extrapolated from neighbor detections

For the latter type there is no bottom detection available, even though a center sample number is provided. Therefore, no special care has to be taken for beams with an invalid detection when georeferencing the bottom detection samples. The beam samples time series themselves are sorted differently depending on what side of the swath they are. The sorting direction is provided and indicates whether the first value in a series has the lowest or furthest range (Kongsberg, 2018).

Considering the above findings, following backscatter georeferencing procedure is developed:

1. Associate processed bathymetry and raw seabed image swaths and beams using the ping counter and beam index.
2. Georeference the sample of each beam time series which corresponds to the bottom detection.
3. Concatenate the samples between to adjacent bottom detection samples.
4. Interpolate the newly formed sample time series by equally spacing the latitude and longitude (and depth, eben though it was found to not be of relevance for the equidistant sample spacing) differences according to the number of samples.
5. The two sample series outside the outermost beams are not georeferenced as there is no justified assumption on how the bathymetry continues.

3.5. Visualization of bathymetry and backscatter data

Potree (Schütz, 2016) is a web based point cloud renderer which can directly visualize EPTs. This has the big advantage, that one can share a point cloud data set without the need to install any third-party software. However, Potree was not the first attempt for the data visualization. Python itself comes with some more or less static visualization tools. It was whatsoever found that interactivity of the visualization is crucial to inspect any data processing result. Therefore an alternative to the classic matplotlib of Python was searched. Project Jupyter itself offers some interactive plotting capabilities such as ipyvolume. Ipyvolume is specifically designated to 3D point data and may be seen as the equivalent of imshow extended to the third dimension. While ipyvolume works well in Jupyter Notebook and can be easily integrated with ipywidgets, it was in the first place designed to work with arrays stored in the working memory (RAM). Thus this approach quickly reached its limit due to the large amount of data. Finally, Potree solved both the interactivity as well as the performance issues of the previous approaches.

Since it is web-based just as Jupyter Notebook, the integration of Potree in a notebook can be achieved with a simple iFrame (inlineframe) which is a HTML object that can be embedded in a fixed frame on an existing website. A typical example of an iFrame object is a YouTube video embedded in another website. To initialize Potree, a HTTP server has to be started in the directory of the EPTs. In the Entwine documentation the NodeJS http-server is used to serve the data to the remote Potree page. In Python the http-server can be started as a subprocess which can be terminated once the data has been delivered. It was noticed that when the subprocess was canceled before the data was fully transmitted and the Potree page loaded, the higher level nodes containing the high resolution data were missing for some EPT tiles (Fig. 3-7).

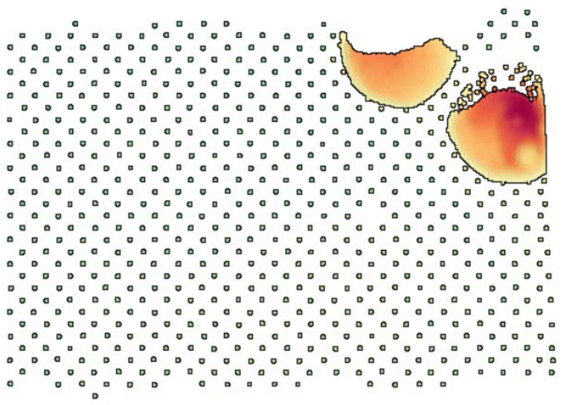


Figure 3-7: Resolution issues of individual EPT tiles which appeared when the server was canceled to early.

While the two ‘hills’ on the right side of the image are fully loaded at the scale, the major part lacks that resolution level. To avoid that, a timer was set in place which delayed the termination of the HTTP server for a couple of seconds. Another issue (which was at least noticed under Windows 10 using the Firefox Browser) was that the Potree viewer did not update even though an updated EPT data set was provided. While it at first seemed to be an issue of the NodeJS http-server, it was later found that the phenomenon is simply caused by the browser cache. The problem can be solved by changing the browser cache configuration. Alternatively it could be tested if another server (such as live-server) behaves differently.

The specifications of the data to be visualized in Potree are strongly influenced by the LiDAR requirements and standards. In first attempts of integrating Potree, the default decimal precision imposed issues as it was set to 2. With the point cloud data in geographic coordinates it resulted in a pattern which can be seen in figure 3-8.

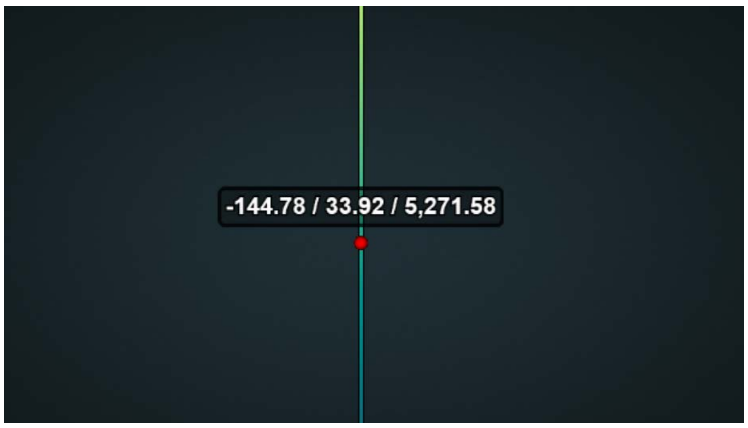


Figure 3-8: Point cloud is pressed together due to an insufficient decimal precision.

The point cloud appeared as if it was pressed together. This pattern was simply caused by the effect that small scale differences in geographic coordinates are only noticeable at later decimal places than the second. There are two possibilities where the decimal precision could have been lost:

During the Entwine build in the conversion from ASCII to EPT or in the Potree Viewer itself. In its build command, Entwine offers to set the scale of the data which defines how many decimal places are preserved. Alternatively, the absolute parameter can be set to true. While scaled values with a fixed precision are preferred, the absolute parameter defines double-precision values for x, y, and z instead. With both options, the LAS files in the EPT octree preserved the defined or initial precision so that the issue was actually caused by the Potree configuration. While Potree can generally be configured, it would diminish the simplicity of using a remote server without the requirement to keep any customized code locally. Also it was found that the use of metrical coordinates such as projected UTM coordinates provides significant advantages for subsequent processing stages. Therefore it was decided to project the geographic coordinates to the suitable UTM zone (determined on the northwestern extent of a data set). For the projected metrical coordinates, the decimal precision was found to be suitable.

Challenges were generally faced with the data types of the bathymetry and backscatter attributes. Potree is intrinsically bound to the LiDAR attributes, e.g. point clouds can be colored by RGB, elevation, intensity, classification etc., most of which are useful in a hydrographic context as well. Thereby Potree expects attributes to be formatted in a specific way in order to detect and visualize them. In PDAL and Entwine the point attributes are expressed as dimensions. Dimensions are thereby a construct used to structure the point data. All points are defined by a set of dimensions, the most basic being the x, y and z coordinates. As Entwine uses PDAL readers for data input and conversion, the required configuration information generally relates to PDAL. A PDAL reader (stage) attempts to automatically detect a known dimension if the header infers it. To represent the backscatter amplitudes, a suitable dimension has to be found. Unfortunately, there was no perfect match available. Generally, there are two potential options of which both have their advantages but also respective drawbacks: Firstly, the intensity dimension could be used. The expected data type of intensity is an unsigned 16 bit integer which corresponds to a range from 0 to 65,535. However, the backscatter data is usually negative as it represents the relation of the intensity backscattered at a target to the incident intensity and therefore it requires a signed representation. When passing negative values to the PDAL reader, the backscatter intensity is whatsoever not shown in Potree. At this point it is not sure where exactly this loss occurs. It was observed, that the backscatter values are stored as double if the header is changed, so that the data is stored in a custom dimension. Though, Potree does not recognize it as intensity then. If the header is changed to ‘Intensity’ the data is stored as unsigned 16 bit integer, so that the values cannot be negative. This seems to cause an internal invalidation, because the points are not shown then. Generally, Entwine itself provides the possibility to set the schema in which the dimensions are stored. When passing a suitable data type (signed 16 bit integer), the points do not appear either. Hence, in order to show the backscatter values as attributes in Potree, it would be required to convert the data type to a representation which is technically incorrect. For following processing this is highly prone to failure. The second option is to use the amplitude dimensions. The amplitude dimension in PDAL is stored as float and can therefore represent the true backscatter values. However, one sacrifices the representation of backscatter in Potree since it does not (by default) support amplitude attributes. As the x, y, z points are still visualized in Potree, it was decided to use the amplitude dimension for the backscatter samples and prioritize on the technically correct representation.

3.6. Filters and corrections

The implementation of the bathymetry filters and backscatter corrections is founded on PDAL's pipeline concept. A pipeline basically consists of a data 'reader' stage which handles the data input, one or more 'filter' stages which process the point cloud and a 'writer' stage in order to generate the desired output. PDAL pipelines are formulated as JSON arrays or objects (depending on the syntax applied). Their use for data processing has two significant advantages: Firstly, they provide simultaneous record of how the data was processed, which is, considering the IHO (2008) processing guidelines, an important step when working with hydrographic data. Secondly, they allow to define a suitable processing structure which can be individually adjusted to the requirements on a parameter basis. This is because one can predefine a pipeline in the JSON array, while on the other hand the specific parameters of each stage can be individually substituted to fit the application. For this purpose PDAL provides among others its Python extension. Since Python itself has an interface to JSON objects with the json library, a pipeline can be build and adjusted in Python and then executed via the extension in PDAL. In this sense, PDAL can be seen as the data format and processing handler which is optimized to handle large point cloud data and Python is used as the pipeline constructor and initiator. Thereby the JSON pipelines are build in Jupyter Notebook using widgets instead of plane Python. Widgets are simultaneously very useful but also restrictive. It is very convenient to provide a clean interface to the user so that no parts of the raw Python code are touched which could cause unwanted and/or uncontrolled behavior. However, it is vital for the functionality that the widgets do not restrain the use of a pipeline. A basic example is an integer slider with an inappropriate range. While a simple remedy would be to only use integer text boxes, it was simultaneously found to be quite prudent to restrict behavior which could actually cause malfunction. It is therefore a trade off to restrict as much behavior as necessary, while allowing the user to have as much flexibility as possible. Another aspect is that a user who does not know the suitable settings for its data can be guided by provision of starting values for the pipeline, which were found to be useful for similar kinds of data. For MBES data, the major difference between data sets is caused by depth and bathymetry. The initial idea was to introduce depth classes i.e. shallow, medium and deep water pipeline configurations of which the user can choose. According to the set depth class, the default values are adjusted. While this is a very convenient solution for the user, it becomes confusing when programming the widgets. To implement such an approach, a more comprehensive widget management would be needed.

The pipelines itself are build up starting with the PDAL EPT reader. It simply requires the path to the ept.json file of the EPT data set. That file includes the "core metadata required to interpret the contents of an EPT dataset" (Hobu, 2020). This includes the data set bounds, the data and hierarchy type, the number of points, the schema, the span of voxels in each spatial dimension, the spatial reference system and the version. The EPT reader provides all stored dimensions to the pipeline. The next section of the pipeline are PDAL's filters which operate on that data. Generally, they "can remove, modify, reorganize, and add points to the data stream as it goes by" (Bell et al., 2020). The result of those modifications is then in the last pipeline stage written to the desired output. For example when classifying outliers as for the bathymetry cleaning, the result of the filter stage(s) is the point cloud itself with a new or modifies dimension 'Classification'. While PDAL does not provide an EPT writer which updates the input EPT directly, it does have an EPT addon writer which supports writing additional dimensions to an EPT data set. However, those add on dimensions are stored separately from the original EPT data set. On the one hand this is thorough since the raw data is not altered, though it comes with the drawback that Potree does not visualize them. In order to visualize any modifications to the data, the EPT needs to be updated, the HTTP server has to be initiated again and then Potree can be updated to show the processing result. A workaround found to update the EPTs is to use an intermediate file from which the EPT is build again. It is insofar redundant as the entire hierarchy has to be build again, but it enables the data visualization. Also it was found that the use of a temporary BPF (Binary Point File) file provides the capability to rapidly update the EPT.

3.6.1. Bathymetry filters

For the bathymetry filtering pipeline, three PDAL noise classification filters are used: The extended local minimum filter, the radius outlier filter and the statistical outlier filter. The individual filters are described and tested in the sections below according to their order in the pipeline. Additionally it was found to be helpful to have a depth window filter that allows to define a min and a max depth. Points outside this window are rejected. Therefor PDAL's assign function is used. It simply assigns a defined classification value to points where a certain criterion (where Z > min depth or Z < max depth) is met. As recommended by IHO (2008), points which are suspicious should not be deleted but instead flagged with an appropriate indication. Thereby flags which were applied automatically instead of manually, e.g. by a filter, should be carefully verified by a hydrographer. To meet this criteria, a discard-continue-apply cycle is introduced which is meant to iterate the bathymetry cleaning until the hydrographer is satisfied.

The cycle starts with the bathymetry data whether it has already been classified previously or not. Thenceforth the PDAL filter pipeline is configured: The user can decide what filters to turn on or off and adjust the respective filter parameters using the provided widgets. Afterwards the pipeline is run through Python. Initially, the PDAL Python extension was used for this step. In this connection however, the extension unfortunately has a bug (the bug was already reported). The 'where' clause of the filter stage, which is used to define what points should be considered during the filter, does not work properly through the extension. In order to only consider points which are at this point unclassified, the where clause is required. Therefore this pipeline has to be invoked in a Python subprocess instead. Afterwards, in order to visualize the result, the EPTs are updated and the data is shown in Potree. In the first place, the classification was limited to the two classes 'never classified' and 'low point (noise)'. The latter is the default classification of all PDAL noise classification filters and corresponds to a classification value of 7. However, it was found to be interesting to understand which point was classified by what filter stage in order to allow a more precise parameter adjustment. As an instance, if the filter pipeline flagged too many points, it is beneficial to know what exact filter was too aggressive and should be reduced. Hence, the individual filters were each assigned different classification values. Unfortunately, those classifications are again predefined based on the LiDAR requirements. In the LAS specification (ASPRS, 2018), the point classification values and their meaning are outlined. Potree follows those specification and innately uses them when interpreting the delivered classification dimension. In a hydrographic context, those classes such as vegetation, building etc. are not applicable. Generally, the specification reserves a defined range of value for customized classes. Though again, the default Potree version does not account for them. As before, the workaround found to be comparably convenient is to reinterpret the classes known by Potree and accept that the label does not fit perfectly.

After the outlier classification state, the hydrographer can decide whether to discard or keep the classification. To discard the classification, a checkbox ‘Reject any classification’ is provided. It should be enabled prior to the next iteration in order to ignore any previously defined classes and treat all points as if they were never classified. If the classification is accepted, the hydrographer can decide whether to continue with another iteration. Then, the filters of the pipeline are set to ignore points which are already classified and solely filter the others. Compared to discarding the classification which is necessary if it was too strong, it can be effective to continue the classification rather than restarting it. For example, if certain outlier patterns are caught better if the default order of filters is changed. Finally, the iteration ends once the operator is satisfied. The result is a point cloud with the rejected outliers being classified accordingly. The outliers can then be set to ‘ignore’ in the export of the cleaned bathymetry.

3.6.1.1. Extended local minimum filter

The extended local minimum (ELM) filter is implemented in PDAL based on the method described in Chen et al. (2012). It is meant to only discard points as noise which are considered low with respect to their neighbors. Basis for this is a raster of the point cloud. Within each cell of the raster, the lowest point is marked as noise, if the second lowest point lies more than a given threshold λ higher (Fig. 3-9).

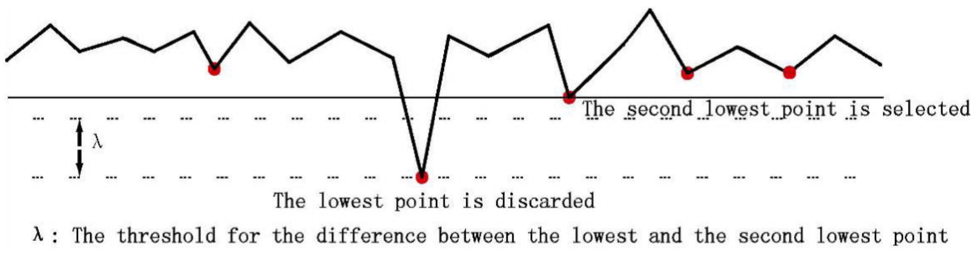
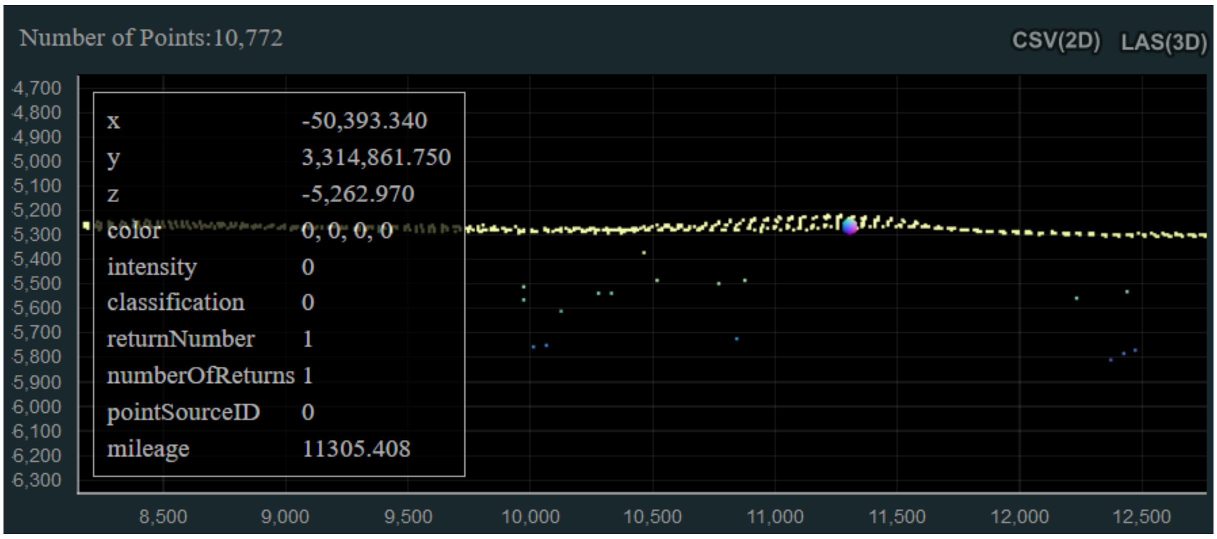

The second lowest point is selected
λ
The lowest point is discarded
λ : The threshold for the difference between the lowest and the second lowest point

Figure 3-9: Schematic representation of ELM filter (Chen et al., 2012)

If the lowest point is discarded, the previously second lowest point is considered and the procedure repeated. This is done until the current lowest point lies within the threshold. Then the iteration is stopped and the next raster cell is inspected for outliers. For the lowest point in the raster cell $\text{outlier}_{\text{lowest}}$, one may formulate:

$$\text{outlier}_{\text{lowest}} = \begin{cases} \text{true}, & \text{if the height difference to second lowest point} > \lambda \\ \text{false}, & \text{if the height difference to second lowest point} < \lambda \end{cases} \tag{3-1}$$

In PDAL the ELM filter can be adjusted to the given data by choosing a suitable raster cell size (default 10.0) and a threshold value below which points are identified as noise (default 1.0). Values are always given in coordinate units (PDAL Contributors, 2020). It comes in valuable for bathymetric data insofar as significantly low points typically have a high probability of being blunders since they cannot represent any objects in the water column. Hence when using the ELM filter, there is a low risk of rejecting outliers which actually represent real features. Also, the ELM filter, if applied properly, provides a good basis for successive filters by rejecting doubtless blunders which consequently cannot disturb downstream filter stages anymore. Regarding the error types detected in the test data set, especially the deeper lying soundings around the nadir beam could be addressed with the ELM filter.


Number of Points:10,772

x	-50,393.340
y	3,314,861.750
z	-5,262.970
color	0, 0, 0, 0
intensity	0
classification	0
returnNumber	1
numberOfReturns	1
pointSourceID	0
mileage	11305.408

Figure 3-10: Center section of across track profile (with metrical axes) created with the Potree height profile measurement tool. The descriptive data relates to the colored ball.

As figure 3-10 shows, those soundings are essentially separated from the seafloor and violate the hypothesis of a smooth and continuous seafloor. To see how the ELM filter generally behaves, it is tested on a small data subset. Figure 3-11 shows some measurements of along- and across track distances of neighboring soundings within the data set. Very roughly, the along-track distance at a water depth of 3,500 m is 80 m and the across-track distance is 40 m.

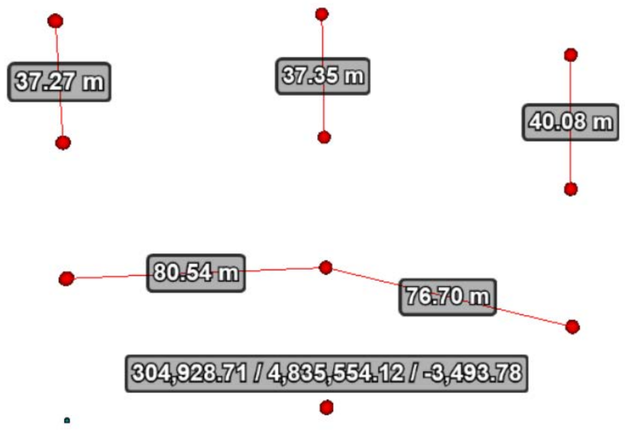


Figure 3-11: Along- (vertical) and across (horizontal) distances measured using the Potree distance measurement tool and a single point measurement.

In order to have a sufficient number of soundings per raster cell, an initial raster cell size of 200 m is passed to the ELM filter. To determine a suitable value for the threshold, some height measurements are taken at the steepest slope of the data set (Fig. 3-12). No regular height differences exceeding 20 m were detected, and thus, it is set as threshold.

From 689,904 soundings in the data, the applied ELM filter classified 868 as low outliers which

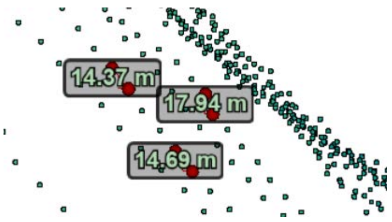
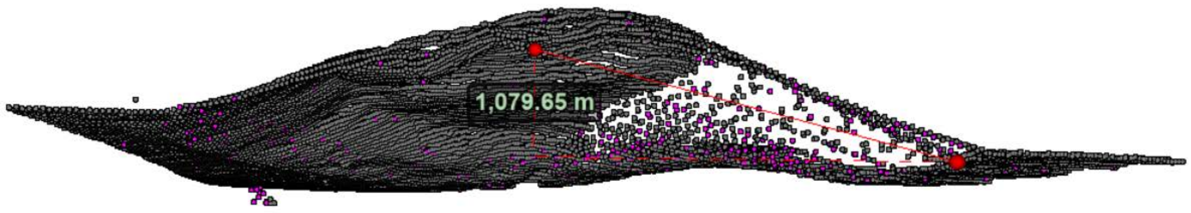
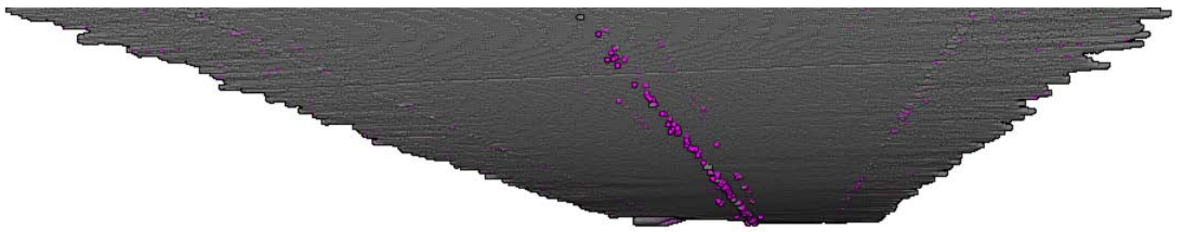


Figure 3-12: Height differences measured using the Potree height measurement tool.

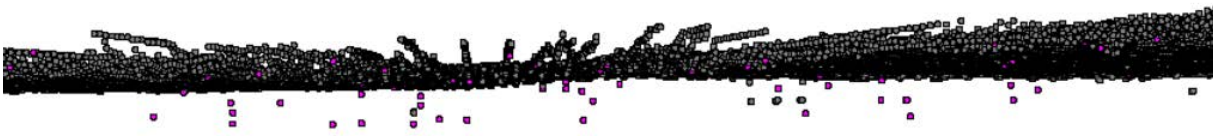
corresponds to 0.13 %. At first sight, the result looks promising. In figure 3-13 a it can be seen, that some false outliers were detected along the left slope. The right slope was facing away from the MBES so that it was shadowed by the mountain. The soundings are most probably interpolated and sparser. Under those circumstances the number of soundings per raster cell decreases, limiting the overall number of soundings considered by the algorithm. Figure 3-13 b shows a view of rather flat data in along-track direction. Most of the outliers were detected around the center beam and in the outermost beams. This pattern is insofar typical for bathymetric data, and thus the result plausible, as the accuracy of the outer beams is worse than towards the center. The center outliers were indeed detected by the algorithm. However as figure 3-13 c shows, especially outliers which appeared in groups decreased the filter result.



(a) Section showing half of the swath from inner beams on the left to outer beams on the right.



(b) Section showing part of the data in along-track direction.



(c) Section showing part of the data in across-track direction.

Figure 3-13: Result of ELM filter (cell size: 200 m and threshold: 20 m). Classified outliers are colored pink.

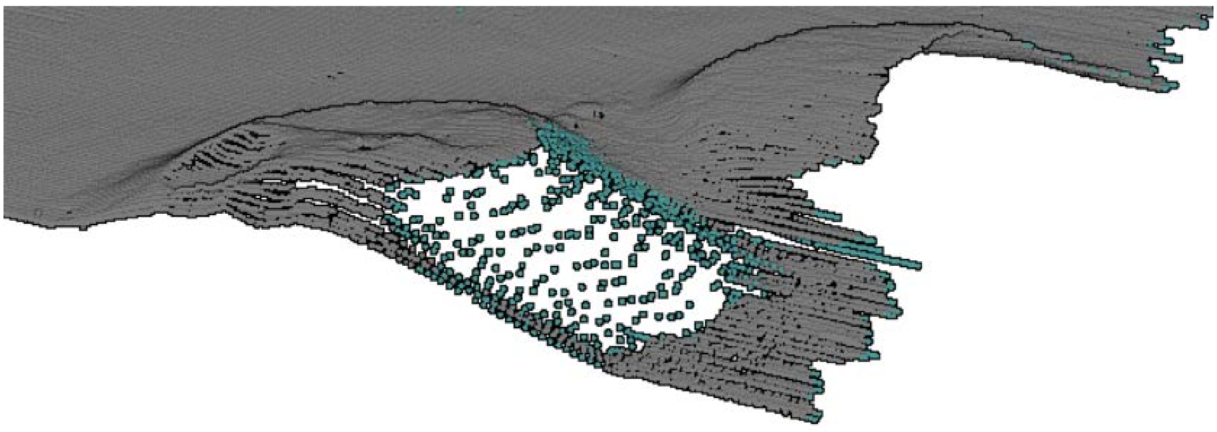
Altogether, the ELM filter seems suitable for bathymetry processing and a good choice as initial filtering stage. When choosing the parameters, the depth is determining for the raster cell size because it decides how dense the soundings are. If a rough bathymetry is expected, it should not be set too large to avoid strong smoothing effects. Thereby it should be chosen in a way that enough soundings are within a raster cell without covering too large areas. The threshold is mainly determined by the maximum expected slope. The steeper it is, the smaller the threshold should be since otherwise the algorithms takes out soundings at the deeper edge of the raster cell. The soundings which are rejected in the ELM filtering stage are assigned a classification value of 7 which corresponds to the LAS class 'Low point (noise)' (ASPRS, 2018).

3.6.1.2. Radius outlier filter

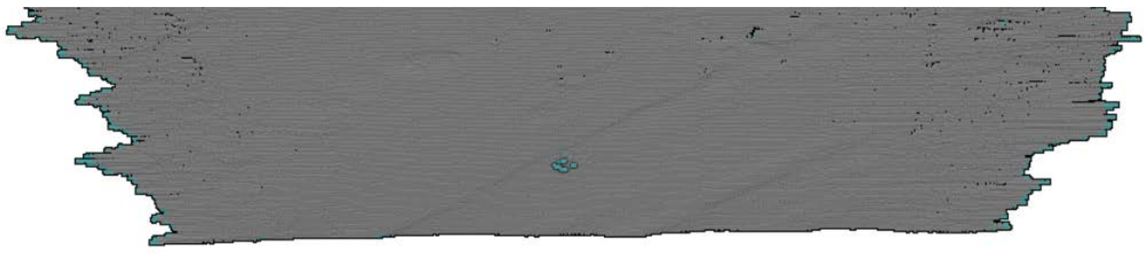
PDAL implements an outlier filter inspired by the algorithm described in Rusu, Marton, et al. (2008). The radius method of the outlier filter requires one pass through the point cloud. For each point P_i it counts the number of points k_i within a defined radius r. A point is discarded as outlier if k_i is smaller than a minimum number of neighbors k_{min} (PDAL Contributors, 2020).

$$\text{outlier}_i = \begin{cases} \text{true}, & \text{if} k_i < k_{min} \\ \text{false}, & \text{otherwise} \end{cases} \tag{3-2}$$

The radius outlier filter can be adjusted using the radius (default 1.0) and the minimum number of neighbors (default 2). It was found to be very suitable for the detection of both isolated outliers as well as outliers which appear in groups but in a structured way i.e. not as dense clusters. Based on the previous measurements, the parameters were set to a radius of 150 m and a minimum number of neighbors of 5. The result can be seen in figure 3-14.



(a) Section showing the edge of a survey line.



(b) Section showing the data from below along the sailing direction.

Figure 3-14: Result of radius outlier filter (radius: 150 m and k_{min}: 5). Classified outliers are colored light blue.

The radius outlier filter rejected 0.6 % of the soundings. Regarding the usual amount of soundings which are rejected (following Le Deunf et al. (2020) estimated percentage of outliers in hydrographic data lies between less than 1 % up to 25 %), the filter was set proportionally rather weak. Figure 3-14 a shows the same mountain that has previously been shown. Again, a lot of the interpolated soundings were rejected. Also the outer beams have a tendency to be filtered out. This can also be seen in figure 3-14 b. The light blue spot in the center of the swath was one of the remaining outlier groups after the ELM filter.

Altogether, the radius outlier filter is based on a very clear outlier definition: A point which has only very little near neighbors. However, it could be seen that this is comparably also valid for e.g. the outer beams. When the filter is adjusted to the given data set, it seems to be capable of providing a good outlier detection. The determining parameter of the filter is the sounding density. Thereby the two strategies are either to use a proportionally small radius and a small k_{min} or a larger radius and a larger k_{min}. The former seems to be better suitable for rougher bathymetry. The soundings which are rejected in the radius outlier filtering stage are assigned a classification value of 4 which corresponds to the LAS class ‘Medium Vegetation’ (ASPRS, 2018).

3.6.1.3. Statistical outlier filter

The statistical method of the outlier filter requires two passes trough the input point cloud. During the first pass, a threshold value is determined based on global statistics which is then used in the second pass to identify outliers. For the computation of the threshold value, the mean distance μ_i of each point p_i in the point cloud to its respective k nearest neighbors is computed. Afterwards the global mean of distances $\overline{\mu}$ and the global standard deviation σ are computed (PDAL Contributors, 2020):

$$\overline{\mu} = \frac{1}{N}\sum_{i=1}^{N}\mu_i \tag{3-3}$$

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\mu_i - \overline{\mu})^2} \tag{3-4}$$

The threshold value t is calculated allowing the user to set a specific multiplier m (PDAL Contributors, 2020):

$$t = \overline{\mu} + m\sigma \tag{3-5}$$

In the second pass, the statistical outlier filter iterates over the pre-computed μ_i values and discards them if the value exceeds the threshold. PDAL Contributors (2020) formulate it as:

$$\text{outlier}_i = \begin{cases} \text{true}, & \text{if} \mu_i >= t \\ \text{false}, & \text{otherwise} \end{cases} \tag{3-6}$$

The statistical outlier filter can be adjusted using the multiplier of the standard deviation threshold (default 2.0) and the mean number of neighbors (default 8). Compared to the previous two filters, the statistical is the only filter which considers global statistics and does not require any absolute distance declaration. During testing it was found that it reacts quite stable i.e. it is a rather robust method. A suitable setting found for the test data set, was a mean k value of 20 and a multiplier of 2.5. the result can be seen in figure 3-15.

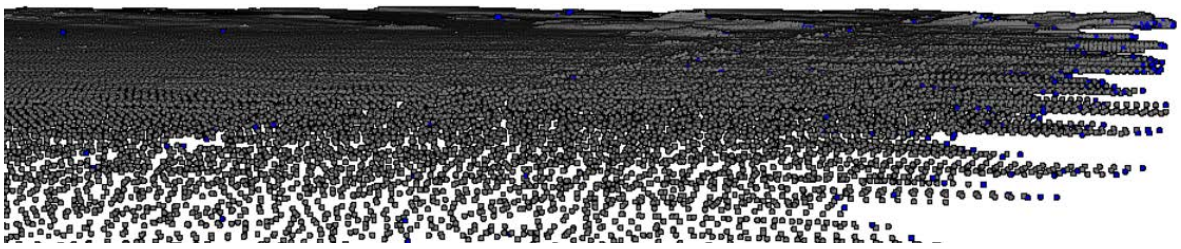
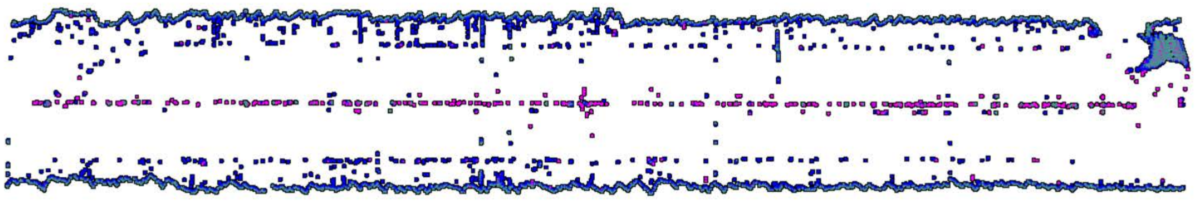


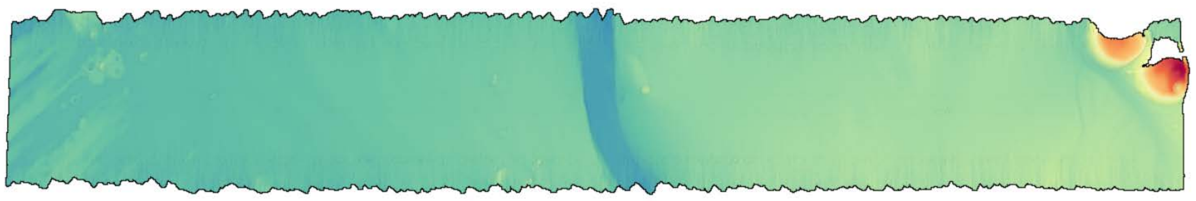
Figure 3-15: Result of statistical outlier filter (mean k: 20 and multiplier: 2.5). Classified outliers are colored dark blue.

Once the coarser outliers are already rejected by the ELM and the radius outlier filter, the statistical outlier filter seems to be proficient in detecting the remaining individual outliers which occur close to the most probable seafloor surface. This is generally desired, however when the threshold is set too low, it could indeed also falsely reject soundings representing small scale bathymetric features. In this sense, the multiplier can be understood to accommodate the structural characteristics of the bathymetry while the mean number of neighbors can be used to adjust the large scale structures. The soundings which are rejected in the statistical outlier filtering stage are assigned a classification value of 9 which corresponds to the LAS class ‘Water’ (ASPRS, 2018).

During the filter performance testing, the statistical outlier filter discarded 0.4 % of the soundings. Altogether, the three filter stages rejected roughly 1 % of the soundings. Generally speaking, it would still be reasonable to configure the filters stronger. The combined result of the three filters can be seen in figure 3-16.



(a) Soundings that were rejected (pink: ELM filer, light blue: radius outlier filer and dark blue: statistical outlier filer)



(b) Soundings that were not rejected i.e. accepted.

Figure 3-16: Combined result of ELM filter, radius outlier filter and statistical outlier filter.

Figure 3-16 a shows the soundings that were rejected and figure 3-16 b shows the soundings which were accepted. The deeper lying soundings in the nadir area were removed satisfactorily. All in all, it seems appropriate that more soundings were discarded in the outer swath, as these beams are typically less accurate. However, the outermost beams have a higher probability of being statistically rejected, only because they lack neighbors on the outside. Also, the interpolated values are almost completely rejected.

3.6.2. Backscatter corrections

As already mentioned, it is beyond the scope of this thesis to implement radiometric corrections. The corrections which are to be implement are rather meant to fit the data set to a given level by adding a constant offset or multiplier and visually enhance the final grid. As previously discussed, there was no perfectly suitable dimension found to represent the backscatter. Of the options identified, the dimension that is technically suitable but cannot be visualized in Potree was chosen instead of changing the backscatter values for visualization purposes. This introduces the dilemma that even though the points itself can be inspected in Potree, they cannot be colored according to their backscatter amplitude. For now, the processed backscatter data has to be exported as grid and then visualized externally. While working with the test data, the open source GIS software QGIS was used for backscatter grid inspection. As it does not require a license, everybody can download and use it. This was considered important as this thesis aims to enable accessibility.

The workflow of the backscatter processing is in principal the same as for the bathymetry processing. A PDAL pipeline is build and then executed in Python. The Jupyter Notebook is again used to parameterize the individual stages. During the bathymetry processing, the ‘Classification’ dimension is (repetitively) adjusted but the bathymetry dimensions (‘X’, ‘Y’ and ‘Z’) remain unchanged. This is the reason why the classification can be reset and started over. For the ‘Amplitude’ dimension it is slightly different. The filters which accept parameters beyond the geometry work directly on the dimension. When updating the EPT data set, the ‘Amplitude’ dimension is irreversibly altered. Hence, the update-reset-accept cycle used in the bathymetry processing cannot directly be adapted. At this time the backscatter results (grids) are inspected in an external software, and therefore, it was chosen to directly use a writer stage, instead of updating the raw data set.

Regarding the planned corrections, altogether four stages have been investigated. The first is the DEM filter. The DEM filter spatially culls points if they exceed a defined distance to a given raster. As the bathymetry data is handled separately, this filter could theoretically be used to keep the backscatter data in a reasonable distance to previously created bathymetry grid. Unfortunately, the given combination of backscatter data and bathymetry grids caused an exception in different tested combinations. The second correction being investigated is to add a constant offset value or multiplier to the data set. This was considered helpful for situations where different sensors with a varying backscatter level are used (as described by Schimel et al., (2015)). The mosaic enhancement corrections which are commonly applied to backscatter data are despeckle and/or antialiasing filters. For both a suitable PDAL filter could be identified. Despeckling is implemented using the median absolute deviation and the antialisasing with the Poisson sampling filter.

3.6.2.1. Constant offset or multiplier

To manually alter a dimension in PDAL, the assign filter stage is provided. The assign filter requires a defined syntax to express how the specific dimension is supposed to be altered. The following statement shows the syntax used to formulate an assign filter expression (Bell et al., 2020):

"Dimension = ValueExpression [WHERE ConditionalExpression)]"

Thereby the stated dimension is modified. The value expression defines how each value of the dimension is modified and the conditional expression can be used to express what specific values are modified. When applying this to the backscatter data, it could exemplarily be formulated:

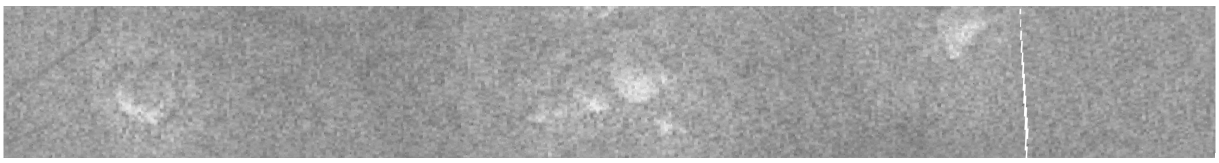
"Amplitude = Amplitude $*$ 0.7"

"Amplitude = Amplitude – 20"

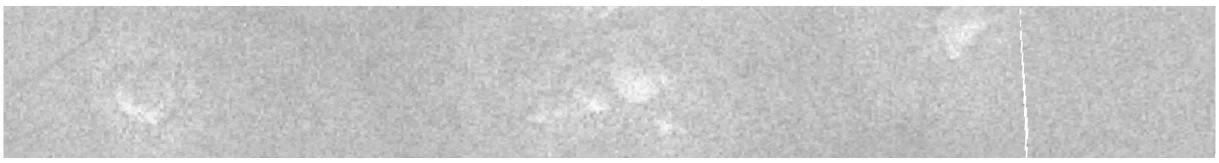
The first expression would multiply each backscatter amplitude value by 0.7 and the second statement would remove –20 dB from each backscatter value. A conditional could as an instance be used to apply the value expression only to a specific range of values.

"Amplitude = Amplitude – 20 WHERE OriginId = 0"

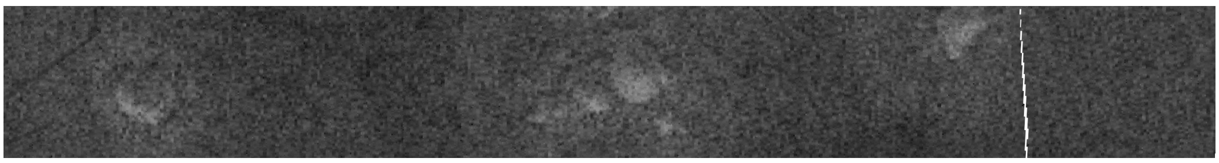
This statement only applies the value expression to amplitude values from the source file indicated by the OriginID 0. The results of the previously constructed expressions can be seen in figure 3-17. Thereby all grids are represented with the same gray scale.



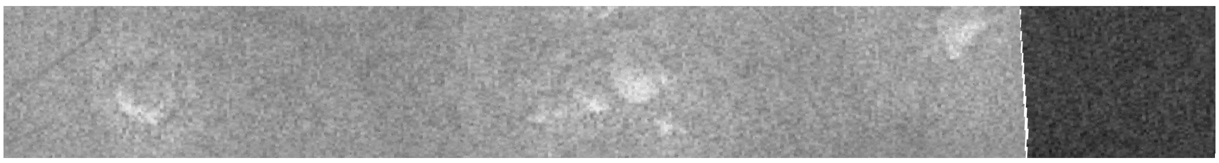
(a) Raw data set.



(b) Data set times the multiplier 0.7.



(c) Data set minus a constant offset of 20 dB.

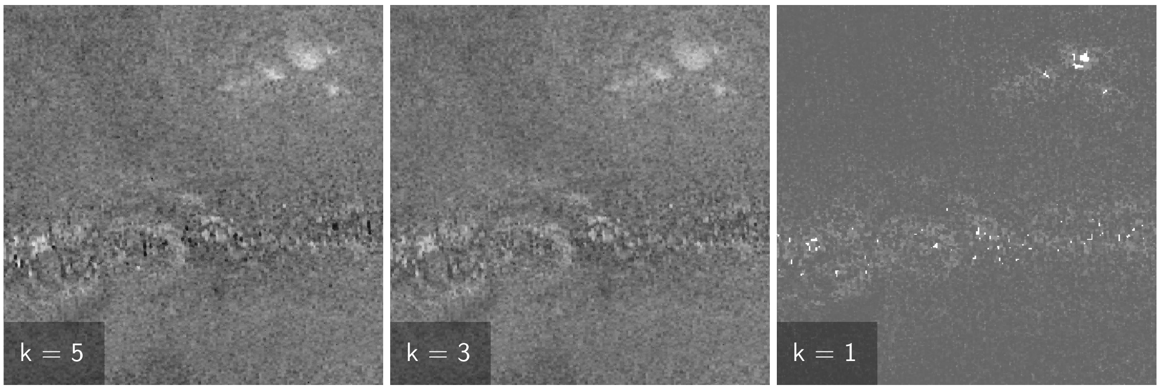


(d) Data set modified where OriginID = 0.

Figure 3-17: Backscatter grid multiplier/offset settings.

3.6.2.2. Despeckling

The backscatter despeckling correction is meant to reduce the high-frequency noise in the backscatter data in order to visually enhance the final mosaic (Schimel et al., 2018). In PDAL there are a variety of filters available which cull a point cloud based on a specific criterion. For the despeckling, it is important that the filter is not solely geometry based, such as the filters for the bathymetry outlier/blunder cleaning used above, but considers the amplitude dimension of the data set. Filters addressing specific dimensions are in PDAL accumulated under the umbrella term "conditional cull filters" (Bell et al., 2020). Among others PDAL offers the interquatile range (IQR) filter and the median absolute deviation (MAD) filter. The former culls points which lie outside the defined interquartile range (difference between the 25th and the 75th percentile) of a given dimension. The bounds can be customized by a multiplier. The MAD filter culls points based on how points are distributed within the defined dimension. Thereby the median absolute deviation is chosen as it is robust to outlier in contrast to the mean and standard deviation. The MAD filter is described in Bell et al. (2020). It uses the median absolute deviation to determine statistical outliers of the passed dimension. It is the method recommended by Schimel et al. (2018) for the despeckling correction. The filter takes essentially two parameters for its configuration: The dimension, which is set to 'Amplitude' when working with backscatter data, and the number of deviations k from the median. k is used to scale the strength of the MAD filter. To evaluate the effect of the MAD filter, it was applied to the same data set with three different settings. The result can be seen in figure 3-18. The gray scale of all grids were scaled to the maximum range of the grid with the largest range.



(a) MAD with k = 5. (b) MAD with k = 3. (c) MAD with k = 1.

Figure 3-18: Backscatter grid with different MAD filter settings (the grids have an equal gray scale).

Generally, the median of the data set was estimated to −26.8 dB and the median absolute deviation to 5.65 dB. The min and max values of the filtered points can be seen in table 3-3.

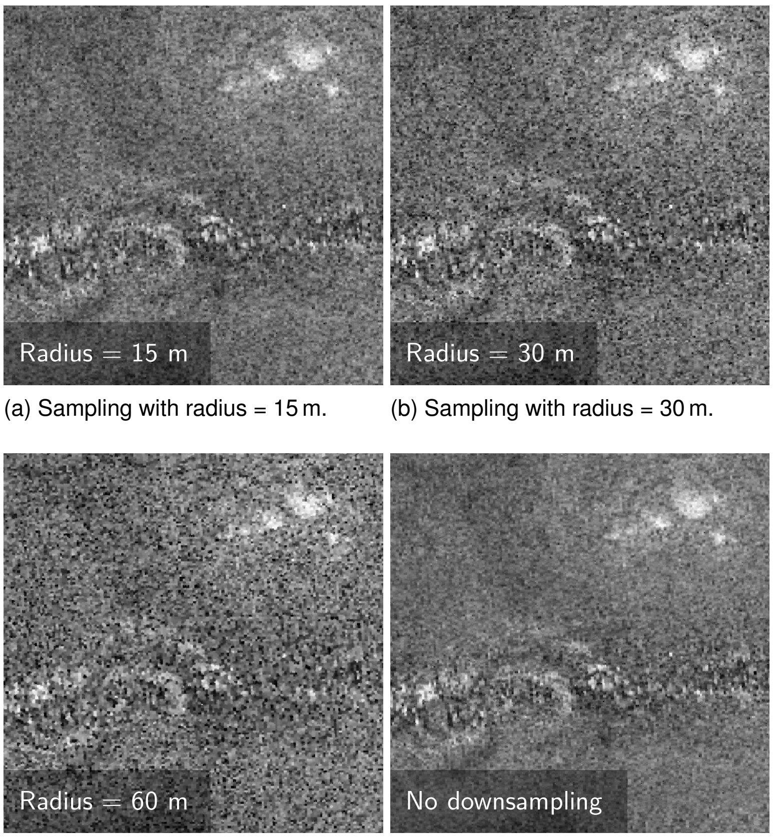
Table 3-3: Min and max amplitudes according to MAD filter settings.

k	Min value	Max value
5	−55.04 dB	1.44 dB
3	−43.74 dB	−9.86 dB
1	−32.45 dB	−21.15 dB

From a visual point of view, the grid shown in figure 3-18 looks the best. It seems as if the darker speckles in figure 3-18 a are almost fulls removed while not retouching the original structures. The grid in figure 3-18 c already lost most of the structures which are still visible in figure 3-18 b.

3.6.2.3. Antialiasing

Aliasing patterns in form of distortion (staircase) artifacts often appear when high-resolution data is represented at a much lower resolution. To reduce aliasing, it can be helpful to downsample the backscatter data prior to the mosaicing to a similar resolution. Also for this processing step, PDAL offers some suitable filters. The most basic resampling filter is the decimation filter, which simply culls every n-th point. However, this procedure acts randomly which may potentially be misleading depending on the point sorting. Another filter which could be utilized for downsampling is the sample filter which performs Poisson sampling. The algorithm culls all points within a given radius of the points to be kept (Bell et al., 2020). In contrast to the other filter it already assures that the retained points are spatially distributed with equal distances. Though, the other samples are simply culled and will thus be completely ignored in the final mosaic. To assess how this will (at least visually) impact the final grid, again a test data set is downsampled with different settings. The result can be seen in figure 3-19. The gray scale of all grids were just as previously scaled to the maximum range of the grid with the largest range. All grids have the same grid settings and a grid cell size of 30 m.



(a) Sampling with radius = 15 m. (b) Sampling with radius = 30 m.

(c) Sampling with radius = 60 m. (d) No sampling.

Figure 3-19: Backscatter grid with different Poisson sampling settings (the grids have an equal gray scale).

Of the total 14,275,618 points in the data set, following statistics were observed:

Radius of 15 m	: Retained 2,043,077 which corresponds to 14.31 %
Radius of 30 m	: Retained 1,055,105 which corresponds to 7.39 %
Radius of 60 m	: Retained 519,717 which corresponds to 3.64 %

Generally speaking, figure 3-19 d is visually the most appealing and smoothest. Figure 3-19 b was downsampled to a radius which corresponds to the grid cell size. Figure 3-19 c, with double the radius, looks rather coarse compared to the other figures. Only figure 3-19 a is visually comparable to the 'raw' data. As the degradation from figure 3-19 a to d shows, the general idea, to reduce aliasing by downsampling the data, has worked. However, it is assumed that at first discarding 85 % upward points has introduced aliasing effects which were then reduced. While this filter is generally functional, it should first be evaluated if an aliasing problem is present.

4. Results

The final program implemented consists of a Python module which includes the functions required for the bathymetry and backscatter preprocessing and a Jupyter Notebook which acts as user interface and constructor/initiator for the data processing (bathymetry filters and backscatter corrections). Both components are connected by an ASCII interface. The preprocessing module is subdivided in individual functions for the bathymetry and backscatter processing. Thereby the backscatter processing is dependent on the bathymetry. The bathymetry (pre-)processing is furthermore subdivided in three steps and corresponding functions. In a first step, the information required for the bathymetry processing (XYZ and position data) is decoded using the pyall module. The extracted data is then stored in respective pandas data frames which are indexed by their (date-)time stamp. These raw data frames are passed to the second step/function which processes the navigation. Therefore, the position data frame is first reduced only to the positions from the active system. Then the remaining (active) positions are interpolated to the XYZ time index. The interpolation can be configured via the function parameters, which allow to set the maximum number of consecutive pings for interpolation (default 2) and a possible time offset, which can optionally be added to the navigation time index. The interpolation method can also be passed as a parameter, but it is recommended to leave it at the default method since it is based on the index time values. Finally, the function passes a data frame which includes the combined navigation and beam bathymetry (swathes which lack one or the other are dropped). In the third step, the vessel coordinates of the beam bathymetry are located using the interpolated ping positions and heading information. Therefore, the *Direct* method of geographiclib's geoid class is vectorized to the rows of the previously combined data frame. The resulting data frame which contains the located soundings, can then be written to an ASCII file.

The backscatter (pre-)processing is based on the previously created bathymetry data frame as input, and then accumulates the backscatter information while reading the raw backscatter data from source. Initially, a data frame set up was chosen similar to the bathymetry processing. However, that approach required the data frame to have an index with three levels (swath, beam, and sample index). In order to fill the raw data frame, a double for-loop would have been required (for beam in beams of the swath; for sample in sample series per beam). This was considered to be computationally very inefficient. Thus, the nested loop was replaced by an accumulator function. For this purpose Python's reduce function is applied to each seabed image datagram in order to accumulate the georeferenced samples beam by beam for that respective swath. The extracted samples are then collected in respective numpy arrays and can be written to an ASCII file.

In order to continue with the processing notebook, the ASCII files from the preprocessing stage or potential other sources are converted to the EPT format. Thereby, geographic data has to be projected to the corresponding UTM zone. The data processing notebook then requires the path to the EPT directory to open the Potree viewer. The EPT directory will automatically be inserted as source in all PDAL pipelines, which are constructed in the notebook. Simultaneously, all outputs of the notebook are stored along with the EPT data set. The bathymetry filters and backscatter corrections are each build and applied separately. If the required data dimensions are not provided, the pipeline execution will throw an exception. For each of the two pipelines, widgets are provided which can be used to parameterize the stages of the respective pipeline. For the bathymetry filtering, the Potree viewer can be manually updated and the individual filter results can be inspected. Once the pipelines are run, the export can be used to produce a grid of the desired dimension (depth or amplitude) with the defined resolution, windowing etc. The grid automatically includes six bands: min, max, mean, IDW[1], count and standard deviation.

4.1. Comparison to data processed with QPS Qimera and FM Geocoder

During the RV Sonne cruise, 664 h of multibeam data were acquired along a track of roughly 6,350 nm covering an area of approx. 258,805 km^2. The measured water depths ranged from 500 m to 7,990 m (Kinne et al., 2019). Already onboard the Kongsberg raw bathymetry data was imported and processed in QPS Qimera (version 1.7.2). Blunders were manually deleted using the swath and slice editors. The cleaned soundings were then exported in ASCII and GSF format. Subsequently, the bathymetry data was gridded with a 3x3 weighted moving average filter to a raster cell size of 60 m, data gaps interpolated and the grid finally exported to GeoTIFF. Additionally, 100 m grids in grd format were created in GMT[2]. The backscatter data was processed in QPS FMGT[3] (version 7.8.10) with the default processing settings. Mosaics with a raster cell size of 30 m were generated and exported as GeoTiFF (Kinne et al., 2019). For subsequent evaluation, six data subsets are chosen which are meant to cover different bathymetric conditions (Fig. 4-1 and Tab. 4-1):

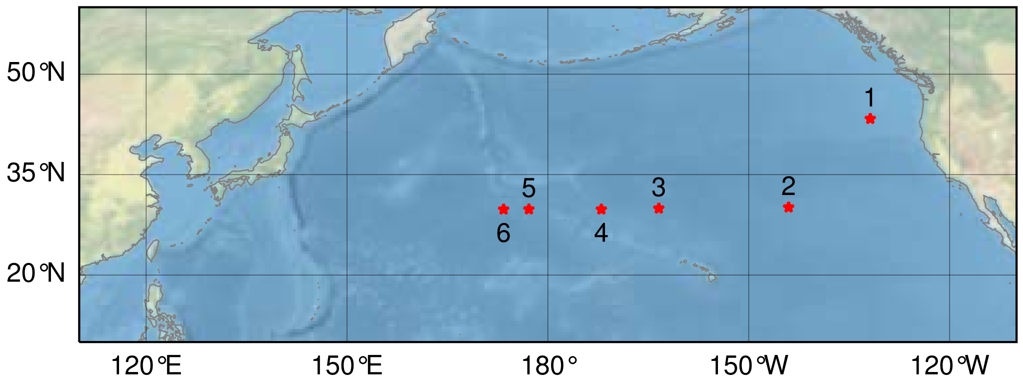


Figure 4-1: Positions of SO268-3 data subsets

To compare the results of the program implemented in this work with the results of the bathymetry and backscatter data processing performed on board, the resulting grids are compared quantitatively and qualitatively. For this purpose, the Kongsberg EM 122 raw data files of the six defined subsets are processed with the implemented program to produce grids that correspond to the grids produced on board. Since the reference grids were produced for each day of the cruise, they firstly have to be clipped to the extent of the respective subset raw data. For that purpose the open source GIS software QGIS (version 3.8 Zanzibar) was used. It simultaneously provides capabilities for the comparison of grids as well as cartographic tools for map production.

4.1.1. Bathymetry

For the production of the bathymetry grids, the raw data for each of the subsets was processed to X, Y, Z ASCII files using the workflow outlined above. The raw bathymetry without any outlier filtering

[1]Inverse Distance Weighted
[2]Generic Mapping Tools
[3]FM Geocoder Toolbox

Table 4-1: Summary of raw data subsets.

No.	Date	Period	Min depth	Max depth	Bathymetric features
1	01/06/2019	03:31 - 07:31	–2,282.64 m	–4,022.30 m	Extension of Columbia River
2	07/06/2019	08:56 - 14:56	–3,772.19 m	–6,143.55 m	Seamount chain
3	12/06/2019	18:56 - 23:56	–5.82 m	–6,775.52 m	Smooth bathymetry
4	14/06/2019	05:56 - 07:56	–3,906.34 m	–5,876.76 m	Steep small scale structures
5	16/06/2019	18:50 - 23:50	–4,576.32 m	–6,600.14 m	Variety of bathymetric features
6	17/06/2019	10:49 - 14:49	–2,042.12 m	–5,898.71 m	Large Seamount

and grid interpolation is presented in appendix B.1 as 100 m mean depth grids. The according standard deviation of the soundings in each grid cell is presented in appendix B.2. Using the bathymetry processing pipeline of the notebook, the bathymetric data was cleaned for outliers and exported to grids. The pipeline parameters used for each subset can be seen in table 4-2.

Table 4-2: Summary of bathymetry processing steps.

		ELM filter		Radius outlier filter		Statistical outlier filter	
No.	UTM zone	Cell size	Threshold	Radius	Min k	Mean k	Multiplier
1	9N (32609)	100 m	30 m	150 m	5	20	2.5
2	7N (32607)	200 m	30 m	200 m	15	20	2.0
3	3N (32603)	200 m	30 m	200 m	18	30	1.5
4	2N (32602)	200 m	30 m	100 m	4	8	2.0
5	1N (32601)	100 m	30 m	100 m	5	6	1.9
6	60N (32660)	100 m	40 m	80 m	2	10	2.5

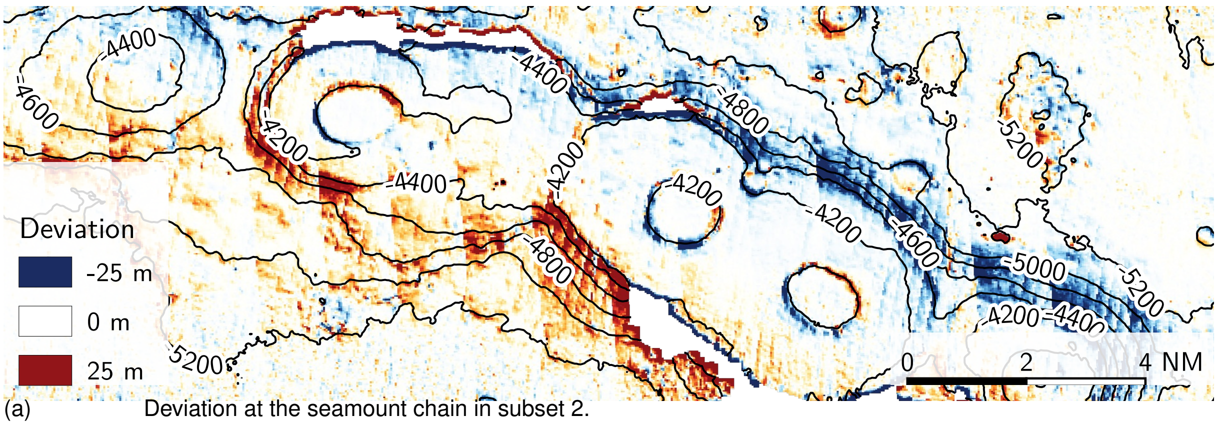
The gridding settings for all of the six subsets were set to a grid resolution of 60 m using a radius of $\sqrt{2}$ times the resolution ($\approx$ 85 m) and a window size of 1 which corresponds to a 3x3 array around cells to be interpolated. The overview of the resulting grids (IDW band) can be seen in appendix C.1. The difference to the reference grids (reference grids minus processed grids) can be seen in appendix C.2. The vast majority of grid cells lies within 25 m depth difference to the grids produced in Qimera. Table 4-3 shows some grid statistics of the difference grids.

The deviation of the grids processed by the implemented program compared to the grids processed in Qimera appears to be rather small. The average relative standard deviation (compared to the mean water depth of the corresponding subset) is 0.18 %. Generally, the quantitative comparison with an independent software suite indicates that there is no unidirectional systematic error in the processing workflow for the bathymetry data. However, the min and max values of table 4-2 show some larger deviations, which especially occurred in association with bathymetric features. Visually, two different occurrences were identified: Firstly, at the slopes of bathymetric features itself such as

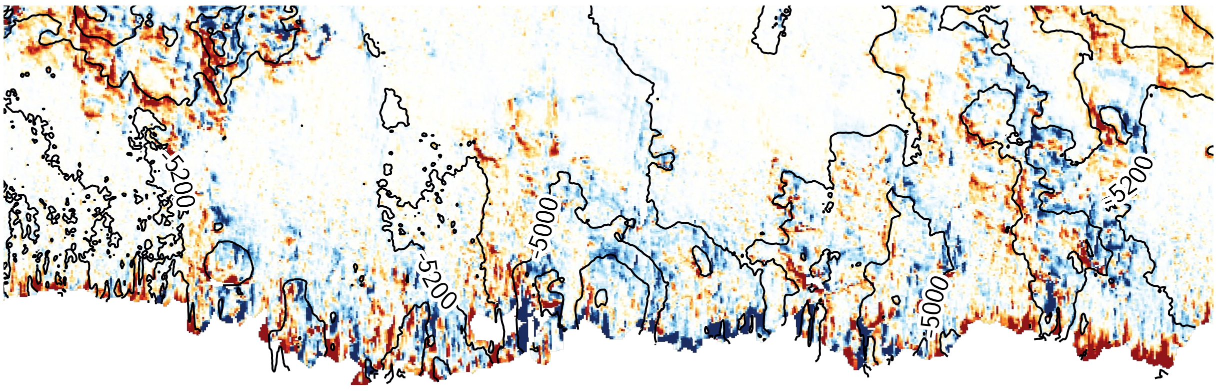
Table 4-3: Summary of bathymetry difference grid statistics for each subset.

No.	Mean	σ	Absolute mean	Absolute σ	Min	Max	Mean depth
1	−0.97 m	4.80 m	2.70 m	4.09 m	−71.14 m	114.16 m	−3,599.76 m
2	0.03 m	10.64 m	4.83 m	9.48 m	−317.19 m	142.10 m	−5,049.87 m
3	0.15 m	4.93 m	2.65 m	4.16 m	−132.62 m	66.15 m	−5,690.55 m
4	−0.04 m	8.53 m	3.59 m	7.74 m	−370.46 m	200.21 m	−5,265.92 m
5	−0.15 m	10.64 m	4.89 m	9.46 m	−223.53 m	202.60 m	−5,251.79 m
6	1.36 m	12.04 m	5.40 m	10.85 m	−243.19 m	173.90 m	−4,682.82 m

the seamount chain presented in figure 4-2 a, and secondly, in the outer beam areas. An example of this can be seen in figure 4-2 b.



(a) Deviation at the seamount chain in subset 2.



(b) Deviation at an outer swath section in subset 2.

Figure 4-2: Deviation of processed grids (implemented program and Qimera).

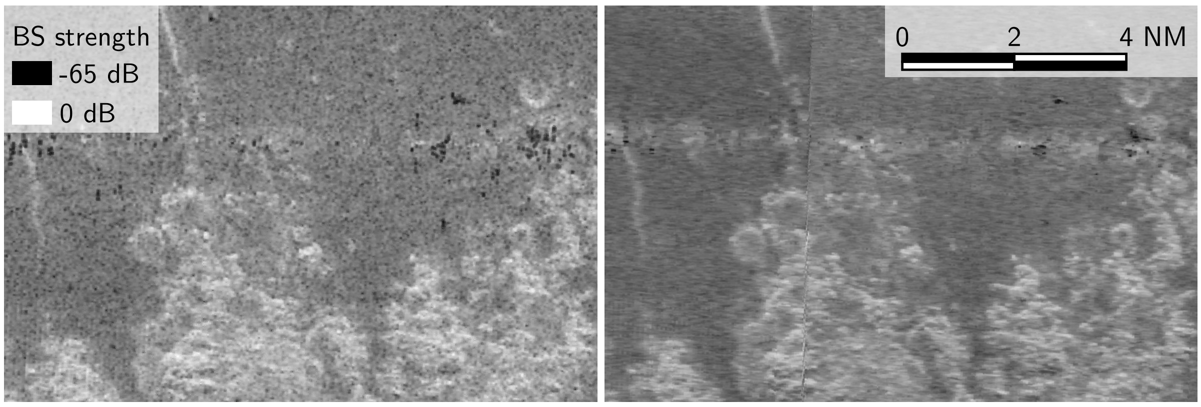
4.1.2. Backscatter

For the production of the backscatter grids, the raw Kongsberg beam time series data was processed to X, Y, (Z,) Amplitude ASCII files using the workflow outlined above. Appendix B.3 shows the raw backscatter data gridded to 30 m mean values with no interpolation applied. All the backscatter subsets were processed with the same parameters: Firstly, the backscatter point cloud was despeckled using the median absolute deviation of the amplitudes with a threshold of 5 deviations from the median. Afterwards, the point cloud was downsampled to a radius of 45 m using Poisson sampling. Finally, the point cloud was gridded to a cell size of 30 m using a grid radius of 80 m which was meant to smooth the amplitude samples. In hindsight it became apparent that other settings would have been more optimal. A 6x6 array was used to interpolate cells without data. All the corrections applied, visually enhance the final grid, nevertheless they are no comprehensive radiometric corrections. An overview of the resulting backscatter grids can be seen in appendix C.3. The processed backscatter of the implemented program and FMGT are compared similarly to the bathymetry data. The difference grids are attached in appendix C.4. As can be seen, the difference of subset 4 shows an implausible pattern. It can be suspected that this is caused by the reference grid. The grid is scaled to a range from 0 dB to 255 dB. Those values are not directly comparable to the ranges of the other grids that range from 0 dB to –65 dB and thus it seems reasonable to exclude subset 4 from the comparison. The difference statistics of the other subsets are listed in table 4-4.

Table 4-4: Summary of backscatter difference grid statistics for each subset.

No.	Mean	σ	Absolute mean	Absolute σ	Min	Max
1	–4.25 dB	4.23 dB	4.98 dB	3.34 dB	–23.77 dB	21.42 dB
2	–4.30 dB	4.52 dB	5.08 dB	3.63 dB	–26.05 dB	29.12 dB
3	–2.86 dB	3.81 dB	3.94 dB	2.67 dB	–22.98 dB	25.93 dB
4	–	–	–	–	–	–
5	–3.01 dB	3.93 dB	4.03 dB	2.87 dB	–23.79 dB	33.12 dB
6	–3.25 dB	3.92 dB	4.20 dB	2.89 dB	–26.75 dB	28.41 dB

The mean values imply a clear negative bias. This impression is strengthened when inspecting the overview of the difference grids (appendix C.4). There are only little white areas but large orange regions. Hence, the processed grids tend to have lower values than the reference grids. Thereby, the effect seems to increase towards the outer beams. In fact, the strongest positive deviations occur in the center beam area. Another pattern observed especially in subsets 2, 5, and 6 is that raw data files had abrupt transitions. Figure 4-3 shows an exemplary section of subset 2.

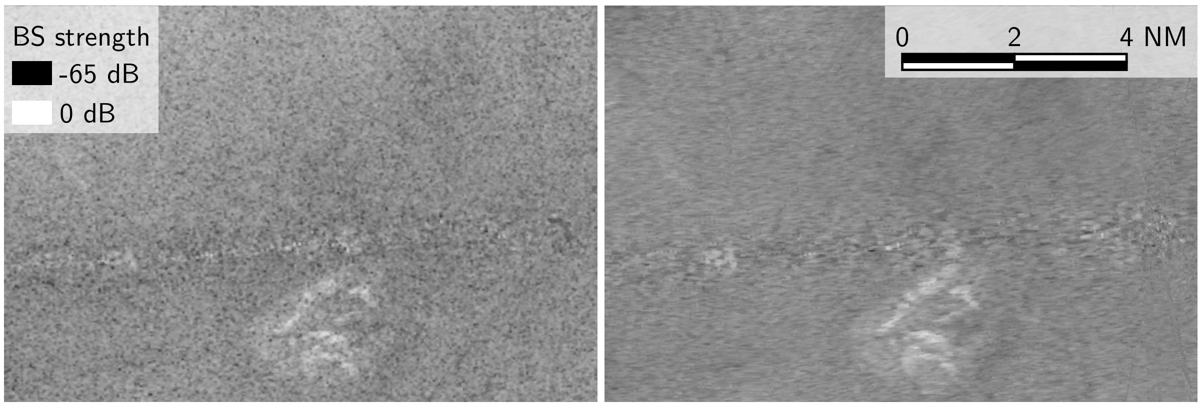


(a) Grid processed in the implemented program. (b) Reference grid processed in FM Geocoder.

Figure 4-3: Section of subset 2 backscatter grids.

Figure 4-3 a is the grid produced with the implemented program and figure 4-3 b was produced in FMGT. It looks as though the abrupt transition only occurs in the reference grid. In figure a there is no apparent difference between the left and the right side of the section. The reference grid looks a bit "smeared" in the along-track direction, which makes the features look comparatively blurry. In contrast, figure a looks rather speckled, but somewhat sharper. However, it does show some high amplitude artifacts in the center beam region that are barely visible in Figure b. The high amplitudes could possibly have been better corrected with other despeckling settings (MAD filter with 3 deviations from the median).

Similar observations were made in figure 4-4. Figure b appears again slightly smeared in along-track direction while figure a appears sharper but also more speckled. In comparison to the previous example, this section was taken from subset 3 in an area with a rather flat bathymetry, except a small elevation.



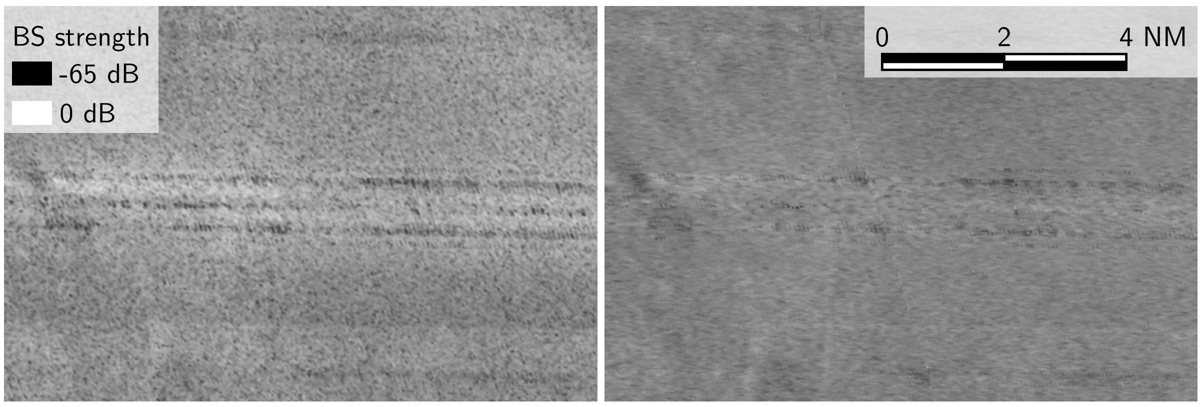
(a) Grid processed in the implemented program. (b) Reference grid processed in FM Geocoder.

Figure 4-4: Section of subset 3 backscatter grids.

Finally, figure 4-5 shows a section of subset 1. This subset seems to be strongly affected by artificial artifacts. It can clearly be seen that there are bands in the along-track direction. They are more strongly visible in figure a and seem to be partially corrected (radiometrically) in figure b. Those are probably artifacts caused by differences of the individual MBES transmit sectors. A detailed discussion of the observed patterns and artifacts can be found in chapter 5.

4.2. Validation with RV Heincke data

The RV Heincke (Fig. 4-6) is one of the smaller German research vessels which was used by a group of HCU Hamburg students in context of a university training survey on the North Sea. The vessel is equipped with a Kongsberg EM 710 multibeam echo sounder. In contrast to the EM 122, it is designed for high to very high resolution seabed mapping in water depths ranging from 3 m to 2,000 m, depending on the in situ conditions. Typically, it can achieve a swath width of up to 5.5 times the water depth. The MBES mounted on the RV Heincke has an along-track beam width of 1° and an across-track beam width of 2°. It operates with shallow water acoustic frequencies in the range of 70 kHz to 100 kHz. Just as the EM 122, it uses the concept of transmit sectors, roll, pitch and yaw stabilization for the transmit beams and roll stabilization only for the receive beams. It also offers equidistant and equiangle mode and dual swath measurements. The system has 128 beams / 200



(a) Grid processed in the implemented program. (b) Reference grid processed in FM Geocoder.

Figure 4-5: Section of subset 1 backscatter grids.

soundings per swath in the given configuration (Kongsberg, 2007).

Figure 4-6: RV Heincke (www.awi.de)

The EM 710 belongs to the same series as the EM 122. That is why both (as far as investigated) use the same datagram formats. Meaning, both MBESs use the XYZ, the same seabed image, and position datagram. Also, they both provide beam data for beams with a valid and an invalid detection in order to allow for backscatter processing. Hence, it is assumed that the implemented data preprocessing should be applicable to the EM 710 data. However, there could be problems related to the shallow water environment. In particular, the high data volumes caused by the much higher ping rate and the strong tidal effects in the North Sea could be noticeable. Similar to the approach used earlier, two subsets are chosen to test how well the implemented program can be applied to the EM 710 data of the RV Heincke cruise. Both selected data sets show wrecks which lie in the south of the German Island Helgoland (Fig. 4-7).

4.2.1. Feasibility for other Kongsberg EM series MBESs

For the RV Heincke data set, recorded with the MBES system Kongsberg EM 710, the same procedure is applied to the raw data as previously to the RV Sonne data set. In order to preprocess bathymetry and backscatter to attributed point clouds, the required raw datagrams are decoded and then merged, georeferenced, etc. As the development of the implemented program is founded on the EM 122 data set, a potential error source could arise from the raw data decoding. But once

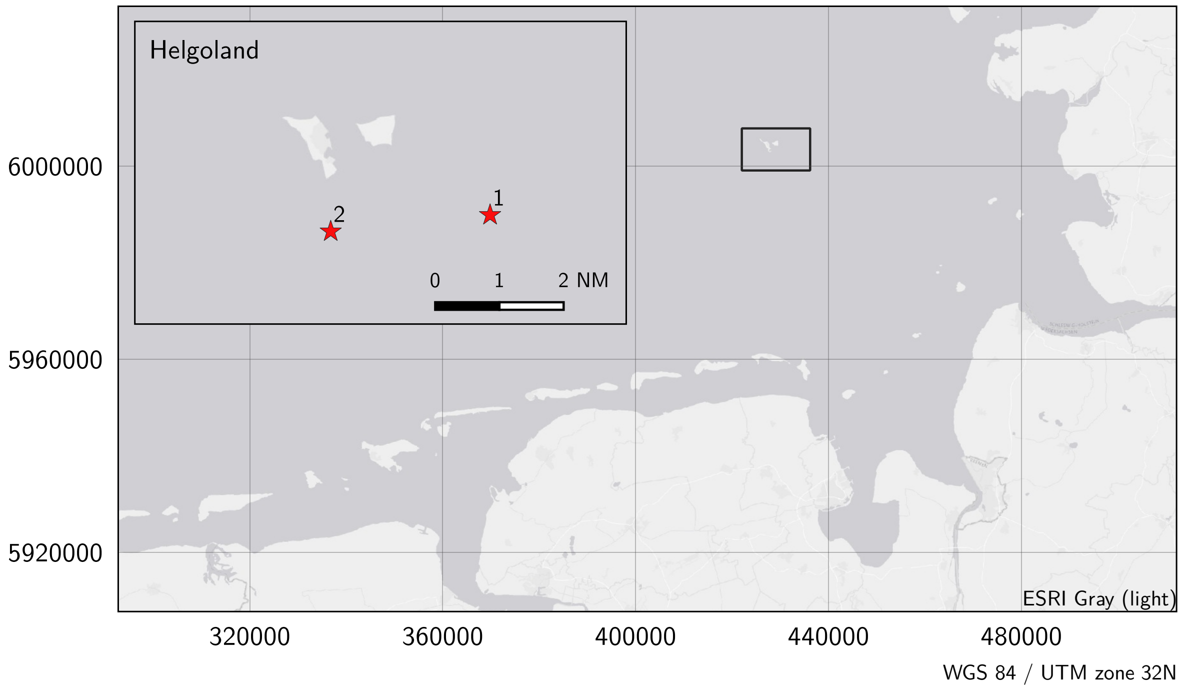


Figure 4-7: Wreck positions in RV Heincke data set.

the necessary information is retrieved, the subsequent processing algorithm should have no difficulties to work as expected. However, in a first attempt to import the EM 710 data, the program threw an exception due to an inappropriate beam index. The EM 122 in the given configuration has 432 soundings per swath. On the other hand, the EM 710 mounted on the RV Heincke only has 200 soundings per swath. Kongsberg has removed the beam index as it became redundant information when they started providing beam data for all beams, also the ones lacking a valid bottom detection (Kongsberg, 2018). Therefore, an artificial beam index was created in the preprocessing that indexed the bathymetry and backscatter information along the swath. It was automatically configured to use 432 soundings as in the EM 122 data. This actually caused the EM 710 data preprocessing algorithm to fail. In order to adapt the new sensor, the beam index was changed to automatically adjust to the number of soundings in a given swath. After this adjustment, the data set could be seamlessly decoded and preprocessed.

Since the application of the implemented program has worked with the EM 710 data, it is assumed that other Kongsberg MBES systems of the same series would also work. In examining the datagram formats described in Kongsberg (2018), it can be seen that there are two groups of Kongsberg MBESs. Of two identified possible combinations, each will use either one combination of bathymetry and backscatter datagrams or the other. Table 4-5 shows an overview of the findings.

In table 4-5 it can be seen that the upper four models are basically the successors of the lower six models. Following the information on Kongsberg's website (www.kongsberg.com/maritime), the latter six have already reached their end of life. With the replacement of the old EM series, the bathymetry and backscatter datagrams were probably also updated. Since the EM 122 and EM 710 could be straightforwardly read and processed by the same decoder, it is assumed that the other models of the new series (EM 302, EM 2040 and EM 2040C) can also be processed with the implemented program.

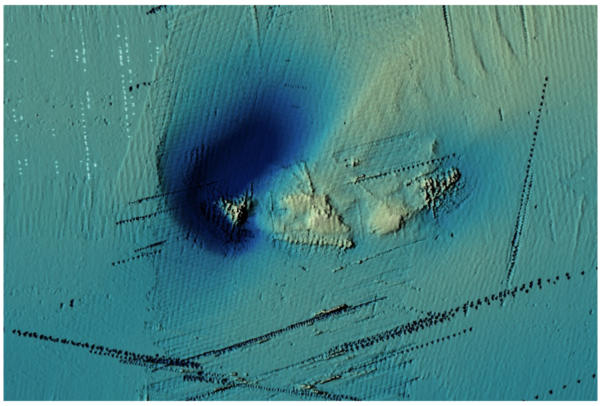
Table 4-5: Summary of Kongsberg EM series datagrams.

Model	Application range	Bathymetry	Backscatter
EM 122	Deep/Very Deep water surveys	XYZ	Seabed image data 89
EM 302	Deep/Very Deep water surveys	XYZ	Seabed image data 89
EM 710	Mid/Deep water surveys	XYZ	Seabed image data 89
EM 2040 (C)	Shallow water surveys	XYZ	Seabed image data 89
EM 120	Replaced by EM 122	Depth	Seabed image
EM 300	Replaced by EM 302	Depth	Seabed image
EM 1002	Replaced by EM 710	Depth	Seabed image
EM 2000	Replaced by EM 2040 and EM 2040C	Depth	Seabed image
EM 3000	*Replaced by EM 3002 ?*	Depth	Seabed image
EM 3002	Replaced by EM 2040 and EM 2040C	Depth	Seabed image

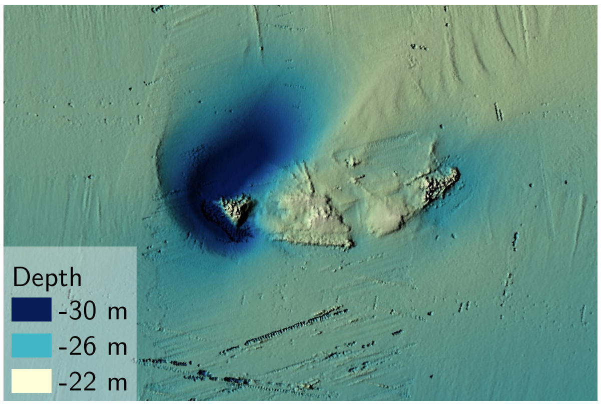
4.2.2. Feasibility in shallow water environments

The RV Heincke data set is located in extremely shallow water with depths in the range of approximately 15 m to 35 m. In contrast, the RV Sonne data set was located between several hundred to several thousand meters of water depth. While the general data structures and patterns of shallow water surveys are typically not much different to deep water surveys, the data amount is expected to increase. With an average sound velocity in water of 1,500 m s^{-1}, the round trip travel time in shallow water can be determined to $\frac{30\,\mathrm{m}}{1{,}500\,\mathrm{m\,s^{-1}}} = 0.2\,\mathrm{s}$ and in deep water to $\frac{5{,}000\,\mathrm{m}}{1{,}500\,\mathrm{m\,s^{-1}}} \approx 3.3\,\mathrm{s}$. Even though this cannot be directly transferred into a ping frequency, it does give an impression of the amount of data that is recorded in shallow water surveys compared to deep water surveys. In contrast to the RV Sonne data with a constant survey line/file duration of 60 min, the length of the files in the RV Heincke data set is variable. Consequently, the number of pings per file also varies. In addition, the EM 710 (unlike the EM 122) has slightly less than half the soundings per swath. Likewise, water level changes have a relatively stronger influence on shallow water surveys. Since the North Sea is a tidal body of water, this effect can be expected to have a significant impact on the outcome of the data processing, especially for wreck 1, which was surveyed on two different days.

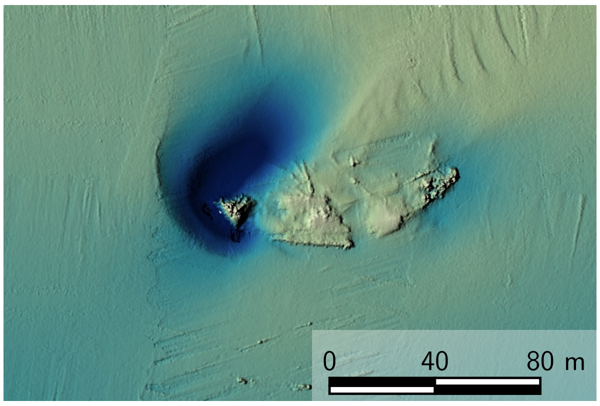
The result of the bathymetry processing can be seen in appendix D. Despite the expectations, the dense data did not impose a major issue. However, it should be emphasized that both subsets cover only small areas. As assumed, the tidal effect can be seen in the bathymetry processed with the implemented program (Fig. 4-8 a). When compared to a data set that was externally corrected for tides and then imported as ASCII (Fig. 4-8 b), it can be seen that the entire dataset is too deep, but also that the individual survey lines are not self-consistent. While the implemented program has been deliberately kept open for external corrections, it is definitely a desirable adaptation to add a tidal correction to the program. Figure 4-8 c shows the tide corrected data set after it has been cleaned for blunders. In general, the filters could also be adapted to the shallow water environment. It was observed that the survey line crossing vertically had very strong artifacts. It was reported that weather conditions were suboptimal at times, which may have resulted in poorer data quality. These artifacts were sometimes so dense that they could not be completely filtered out.



(a) Raw bathymetry (of implemented program).



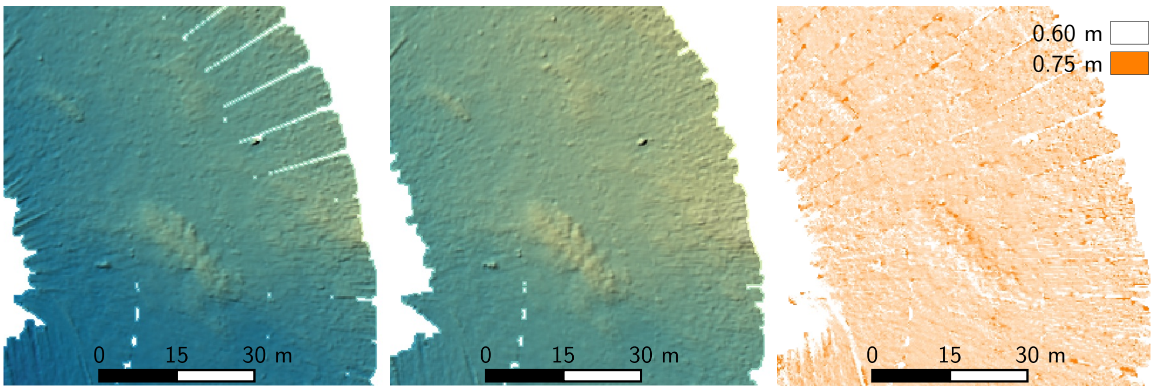
(b) Uncleaned bathymetry corrected for tide.



(c) Cleaned bathymetry corrected for tide.

Figure 4-8: Tidal influence in wreck 1 data set.

Furthermore, in the data set of wreck 2, a pattern was observed in the grid processed by the implemented program. Figure 4-9 shows the same section of the data set with different processings. Figure a was fully processed (preprocessed, cleaned, and rasterized) in the implemented program. In contrast, Figure b was externally preprocessed and corrected for tides, but cleaned and gridded in the implemented program. Figure c is the difference grid (b - a). Most of the difference is caused by the tide, which can be assumed to be constant for this section. However, focusing on the small-scale differences, a similar pattern can be seen as previously noted in the RV Sonne data set.



(a) Result implemented preprocessing. (b) Result of external preprocessing. (c) Difference of figure a and b.

Figure 4-9: Section of the wreck 2 data set.

For the small features in the center of the section, the difference is systematically smaller on one side and larger on the other than in the surrounding area. Also, the data that was preprocessed in the implemented program shows a fan pattern in the outer swath area that does not appear in the externally preprocessed grid. Since only the preprocessing differs in the two data sets, it was most likely caused in this step.

5. Evaluation and discussion

With respect to bathymetry processing, two patterns were identified when comparing with reference data processed onboard. The differences increased first around bathymetric features and second in the outer beam regions. The latter is not surprising in that the filters applied to the bathymetry were generally too weak rather than too strong. Typically, the outer beams have the highest inaccuracies as the errors get worse with increasing (oblique) range and angle of incidence. When manually cleaning bathymetric data, these values are often heavily cleaned and thinned. When using semi-automatic filters instead, one must weigh how aggressively to set the filter to discard all blunders while not erroneously discarding soundings that actually represent the seafloor. Often it is better to falsely accept a blunder than to falsely discard an actual sounding.

Regarding the second pattern observed: It appears that the bathymetry is systematically shifted, as it shows a repeating pattern of negative differences on one side of a bathymetric feature and positive differences on the other side. Further, it is assumed that this pattern does not correspond to an absolute heading, but actually occurs along the direction in which the ship is moving. Furthermore, the measured soundings are always too low in the uphill direction and always too shallow in the downhill direction, compared to the reference grid. Consequently, it must be a systematic effect that occurs along the track. In addition, a corresponding pattern can be seen in the RV Heincke data. The assumption is corroborated because the effect is inverted, for example, at the depression of the underwater river in subset 1. One possible cause could be a time delay between the navigation processing of the two programs, or a pitch effect that was treated differently. The implemented program used the XYZ data, which is already corrected for potential misalignments of the various sensors that are known to the system and defined in the installation parameters. Typically, misalignments such as a sensor tilt, if known, would be calibrated directly in the MBES system. Although this could be a possible source, it is excluded because it would likely have been set directly by the operators onboard and not corrected in the post-processing software. Another source of a pitch tilt could be the dual swath operation mode. If the pitch difference of the two swathes per ping is not considered, this could also cause a pitch error. To verify this theory, it would be necessary to perform the same comparison with a MBES without dual swath.

Regarding the time delay theory: As noted in 3.3, the GNSS and MBES times used, followed the recommended procedure for time synchronization and leave little room for reasonable doubt. Although it cannot be ruled out with certainty, it is rather suspected that it could be caused by the way, the navigation is interpolated to the MBES ping time. Since the effect is bidirectional, it does not show up in the mean of the differences. However, when using the mean of the absolute differences, the respective positive and negative differences do not cancel out and should be visible. Assuming that the absolute mean is mainly affected by this effect, except for a few slightly larger errors, the mean of the respective absolute differences of each subset is used to estimate the error. The mean absolute difference is 4.01 m. Assuming that the ship was mostly sailing at 13 nm, which is approx. 6.69 $\mathrm{m\,s^{-1}}$, the effect would have been caused by a time delay of about 600 ms. In reality, the value is probably somewhat higher because the flat areas reduce the effect of the time offset on the absolute mean. Since the position datagram (of the active position system) was sent in 500 ms intervals (section 3.3), this cause also seems to be reasonably excluded. It cannot be conclusively determined with certainty where exactly the error is induced. By validation with the RV Heincke data, the error can be narrowed down to the preprocessing. A processing error in connection with the dual swath measurement mode could not be excluded at this point.

Another aspect that was noted in connection with the interpolation can be seen in Figure 5-1. It is known that there were some issues with the position datagrams being slightly shifted compared to the XYZ datagrams. This caused some of the XYZ datagrams to pile up at the end of the raw data files with no position information left. The problem was noticed quite early and a solution to the problem was obtained by extrapolating these ping positions. However, as figure (5-1) shows, this did not completely close the gap between two adjacent files. Sometimes, such as on the right side of the figure, it worked better, but the gap on the left side is still quite obvious.

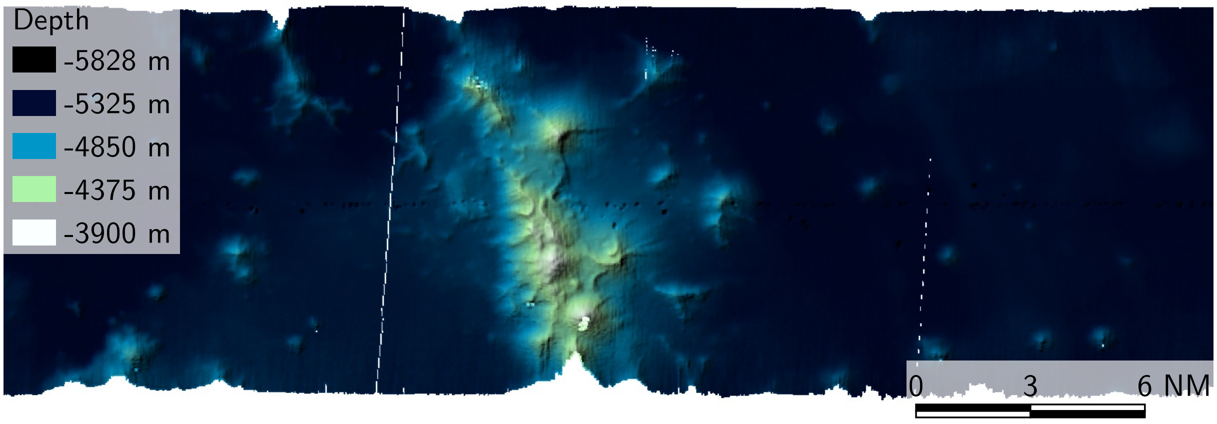


Figure 5-1: Data gaps between adjacent files caused by navigation interpolation.

A third aspect, which was found unrelated to the direct comparison, is that points along slopes that were facing away from the vessel position, were strongly discarded in the bathymetry filtering. In the overview (appendix C.1), those areas can be seen as no data gaps in the grid. The primary cause of this lies in the origin of those points. When the MBES carrying vessel traverses large and/or steep bathymetric features, such as the two seamounts in subset 1 and 2 or the edgy slopes in subset 6, the tilted beams of the swath hit the front while the perspective backside remains acoustically shaded. Beams streaking the feature's backside most probably fail the MBES internal bottom detection algorithm and are therefore invalidated. As stated earlier (section 3.4), all of the 6.23 % invalid soundings were either inter- or extrapolated to provide a basis for the backscatter information. While interpolation is an appropriate means for the compensation of missing data, those soundings are not guaranteed to represent the actual bathymetry. Owing to their artificial origin, it seems as they tend to be rapidly detected by statistical outlier methods. While the data gaps caused by this effect are not nice, it is actually justified to filter those points since it is not certain whether they actually represent the seafloor.

In general, the bathymetry filters used, performed satisfactorily with respect to the comparison with manually cleaned data. However, it was noticed that all the filters adapted, work statistically on a point cloud basis, meaning that effects intrinsic to the MBES swath measurement configuration are not specifically considered. For example the across-track resolution in the RV Sonne data was by a factor two higher than the along-track resolution, the outer beams have larger errors, etc. Since all of the provided filters work on a neighborhood basis, data sets with a highly varying depth would have to be subdivided for filter adjustment. This approach is insofar suboptimal as it would introduce inhomogeneities to the filtering and hence the different error sources would not be addressed equally. Also, large groups of blunders partly fulfill the neighborhood criterion and can therefore not be filtered out. To compensate such effects, both surface-based filters as well as filters that take into account the swath geometry could help.

The backscatter results exceeded by far the expectations, especially since no radiometric corrections were applied. This is also the reason why, strictly speaking, the two data processing results are not comparable, even though an attempt is made to evaluate the differences at least qualitatively. It should be kept in mind that this is partly a comparison of the Kongsberg real-time radiometric corrections and the default FMGT corrections, since they were not specifically adjusted and optimized during the cruise. Notwithstanding the above, the georeferencing looks promising. As can be seen in the results, the grids processed by the implemented program look sharp while the FMGT grids look rather smeared. Schimel et al. (2015) describe that, when using beam time series instead of single value per beam raw data, FMGT uses an approach similar to the one outlined in Beaudoin et al. (2002). As briefly explained earlier, this approach forms half-swath time series which are used to populate a TWTT[1] indexed array. Therefore, the final across-track resolution is essentially affected by the chosen array interval. While this explains the apparent fuzziness of the reference grid, an insufficient array interval would appear as across-track smearing and not along track. This effect is therefore rather assumed to be caused by the applied filter and/or gridding algorithm(s).

In contrast, the solution used in the implemented program concatenates and interpolates all samples provided in the beam time series (except the samples external to the two respective outermost beam bottom detection.) It was thereby noticed that for example at 5,500 m water depth the processed grid is on both sides constantly 120 m wider than the reference grid. The same constant offset was also measured at a water depth of 2,250 m. This is insofar untypical as it excludes any potential origins associated with the water depth. After the raw point cloud data was overlaid, it was found that the boundary of the point cloud corresponded to the extent of the reference grid. Thus, it is suspected that the offset is caused by the windowing applied during the grid creation and does not actually yield additional (and independent) information.

The drawback known in the backscatter processing procedure was the lack of a comprehensive radiometric correction. This was founded on the awareness that Kongsberg already implements a complex dynamic gain that is meant to produce “backscatter data that are reduced to backscattering strength” (Schimel et al., 2015). In order to enable real-time processing, the dynamic gain is based on some simplified models. If the model assumptions fail to depict the actual conditions, it will become visible as artifacts. In the difference grid (appendix C.4), it can clearly be seen that the differences unambiguously show an angular dependence. Therefore it is assumed that the angular dependence was treated differently in FMGT. Kongsberg itself uses a generic model for the correction of angular dependence that does not consider the actual functional interrelationship of the seafloor type. As can be seen in the processed backscatter grids (appendix C.3), the real-time correction however visually seems to be reasonable as there are no obvious across-track dependencies visible. On the other hand, FMGT, after removing Kongsberg’s angular dependence correction, implements a correction that works on a line-by-line (which corresponds to a file-by-file) basis. An

[1]Two Way Travel Time

AVG[2] is thereby applied using corrected backscatter data (Schimel et al., 2015). This difference of the angular dependence correction is undoubtedly what dominates the across-track pattern of the difference grid. As the correction of FMGT works on a file-by-file basis, the abrupt file transitions, which were observed in figure 4-3, probably arise from the default postprocessing AVG. Figure 5-2 shows a section of the subset 2 reference grid that was processed with FMGT. For visualization purposes the contrast was enhanced. The light blue boxes indicate the boundaries of two adjacent files.

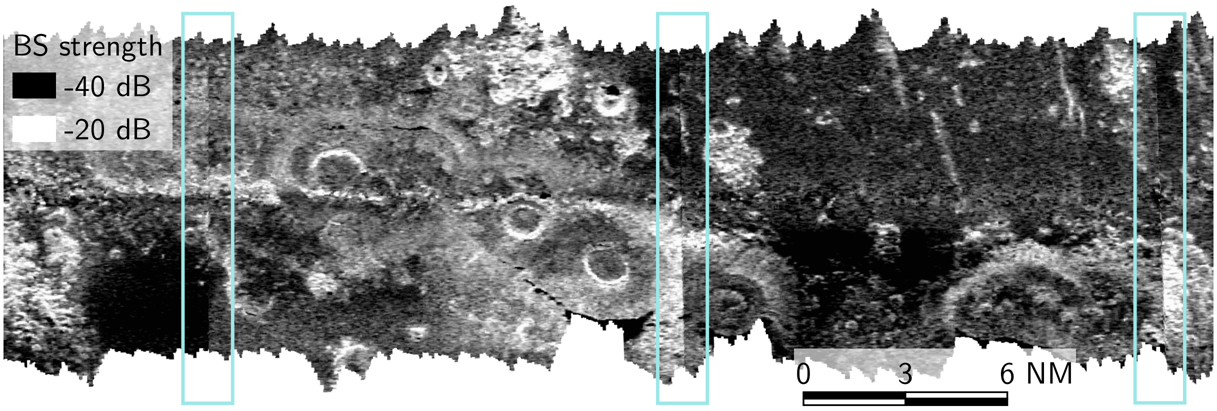


Figure 5-2: Angular dependence correction artifacts in the reference (FMGT) grid of subset 2.

Especially in the center box, the backscatter level of the left and right file has an abrupt transition with a change of level. This actually enhances the assumption, that the along-track backscatter level jumps are artifacts from the file based AVG approach in FMGT, which caused differences in the processing applied to the individual files.

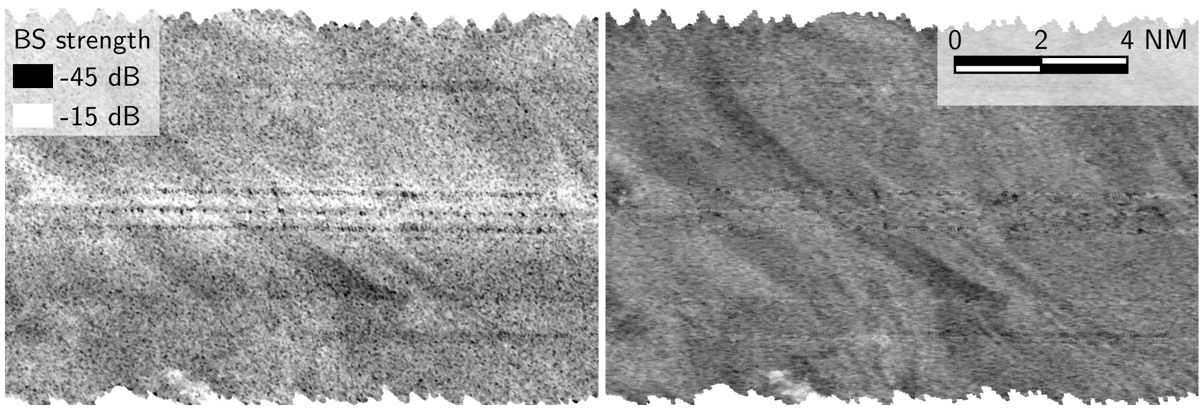
In addition, it was observed that in some of the subsets, namely 2, 5 and 6, the angular dependence differences have strong deviations from one side of the swath to the other. When comparing this observation to the processed bathymetry (appendix C.1), it can be seen that it correlates with the subsets which had the rougher and more diverse bathymetry. Typically, rough bathymetry tends to also be harder since usually soft sediments such as sand or mud would level out and form a rather smooth surface. Both factors have a major effect on the angular acoustic response of the seafloor. It is noticeable that, for example, when looking at subset 2, the seamount chain has a kind of waveform that is once on the port side and then again on the starboard side of the ship. In contrast the surroundings look rather flat and presumably muddy. When determining the angular dependence correction on a subset of data (no matter the length of the subset), which is on one side rough and rocky and on the other side flat and muddy, the determined AVG will reflect this pattern. For the smoother subsets such as 1 and 3, the radiometric correction seems to be very smooth and plausible.

However, besides the above stated relation, no essential bathymetry dependence could be detected in the difference grids. Only weak patterns in subset 2, 5 and 6 seem to slightly correlate with the bathymetry. This is insofar noteworthy as Kongsberg assumes a flat seafloor for the correction of the ensonified seafloor (Hammerstad, 2000). While this does not necessarily mean that the

[2]Angle Varying Gain

ensonified seafloor is perfectly corrected, it does on the other hand show that the reference grids do not have a significantly different correction. Schimel et al. (2015) briefly describe the correction for ensonified seafloor area which is applied in FMGT. For Kongsberg systems it generally uses the nominal pulse width and the transmit specific pulse length (at least for those systems which distinguish transmit sectors). For each beam the algorithm determines whether the ensonified area is pulse or beam width limited. The latter mainly occurs around the nadir beam where the beam ensonifies the footprint instantaneously. If a bathymetric terrain model is provided, it is additionally used to estimate the true incident angle in along- and across-track direction. Since FMGT does only consider the bathymetry if it is explicitly provided, it is assumed that this was not done for the backscatter data processed during the RV Sonne cruise (meanwhile informally confirmed) rather than that the algorithm would not have provided an improved result.

Finally, the along-track banding pattern which was observed especially in subset 1, most probably occurs due to different configurations of the transmit sectors. Among others they are individually adjusted in pulse length and transmit frequency. The consequential source level and (modeled) beam pattern variations of the different sectors are directly compensated and should not appear in the raw backscatter data. Also, Kongsberg is the only MBES manufacturer who completely considers the frequency dependent variations when estimating the absorption in the water column (Schimel et al., 2018). Therefore the only effect, which remains as potential transmit sector caused artifact, is the variation of pulse and beam width. The latter is indirectly varied by the change of frequency. As stated in Schimel et al. (2015), it is not known whether Kongsberg itself considers changes of beam width which would otherwise be visible in the near nadir area. On the other hand, the pulse width is the critical parameter in the off nadir areas. Thereby issues can be specifically caused by the FM chirp waveform which is typically used to extend the range. However, the signal waveform variations were found to be most probably corrected by Kongsberg (Schimel et al., 2018). When directly comparing a section of subset 1 (Fig. 5-3) in the processed grid a and the reference grid b, the reference grid is much more appealing and seems to better work out the local variations rather than sectorial variations. Thereby the nadir stripe in figure a is assumed to be an artifact of the high intensity specular reflection of the nadir beams rather than an artifact of the beam width.



(a) Grid processed in implemented program. (b) Grid processed in FMGT.

Figure 5-3: Section of subset 1 backscatter grids showing along track banding artifacts.

5.1. Comparison to the QPS software suites

Often when working with hydrographic data, commercial software packages are so complex and extensive that it is easy to forget how many components need to be considered in the background to deliver the final hydrographic products. When comparing the implemented program, the aim is not to compete, but rather to assess how many of the required components have been achieved and what important steps are still missing. Emphasis is placed on the aspects identified in the discussion of the results. The software comparison is based on the two QPS suites that were used to process the RV Sonne reference data: Bathymetry processing was done in QPS Qimera, and backscatter processing was done in FMGT, a module of QPS Fledermaus.

Bathymetry processing

The general workflow in Qimera is actually similar to the workflow implemented in the program. At the beginning the raw data is decoded. Based on the extracted datagrams, the bathymetry point cloud is calculated and then cleaned for blunders.

When the raw Kongsberg ALL files are imported, Qimera attempts to decode as much of the given information as possible. In contrast to the implemented program, it also decodes all the data required to process the raw bathymetry data stored in the raw range and angle datagram. That includes among others the vessel configuration stored in the installation datagram, the vessel motion reduced from any Kongsberg real-time corrections, reduced positions, heights, any applied SVPs and the surface sound speed used for beam forming, as well as the sonar settings collected from the various sources (*How-to Qimera: Migrate a Qimera Project to FMGT* 2020). When using the raw bathymetry, all the IHO (2008) required sensor corrections can be applied and/or individual sensor data can be substituted by more suitable data (i.e., the vessel configuration could be changed and/or other motion data could be used). As the implemented program is based on the XYZ bathymetry, all these corrections cannot be applied. While this is a drawback, the amount of data Qimera needs to collect to process the raw bathymetry shows how complex it really is. Since Qimera removes all corrections which were applied by Kongsberg to the sensor data in real-time, it complies with the assumptions that the shift of the bathymetry observed during the comparison is caused by a preprocessing difference.

On the other hand, Qimera also offers to use the processed (XYZ) bathymetry instead of the raw data. In this case, the post-processing options are limited, but it still provides a better basis for comparison. When using the XYZ datagram, Qimera still offers the correction and substitution of the navigation data and the tide correction. For the latter, it should be noted that Kongsberg ALL is one of the formats that does not provide tidal data. Therefore, an external tide file is always required when applying tidal corrections (*Qimera: Sonar Source Data Import Capabilities* 2020). Especially the tidal corrections would have been useful for processing of the RV Heincke data set, since it clearly suffered from tidal effects which could have been resolved by a tide correction. The corrections which are not possible anymore in Qimera when using the XYZ bathymetry, include vessel configuration, sound speed reprocessing and motion correction.

Qimera offers both manual and (semi-)automatic approaches for bathymetry cleaning. Manual data cleaning can be either area-based or swath-based. In general, both tools allow the operator to manually discard suspicious soundings. In the case of the area-based approach, the surface of a selected data subset is isolated for manual cleaning. On the other hand, the swath editor is survey line based and allows the operator to individually clean consecutive swathes of that specific line. Experience has shown that manual cleaning functionalities can come in very handy. Especially the grouped blunders can be difficult to detect by filters, although they are very obvious to the human eye. However, manual data cleaning is also a lengthy task and to some degree dependent on the hydrographer. While a good hydrographer probably achieves the better results, it still does not guarantee consistency when combining data sets processed by different hydrographers. That is why it should be focused on improving the filtering methods rather than recreating manual cleaning functions with open source tools.

The (semi-)automatic approaches in Qimera include spline and CUBE[3] filtering. The spline filter is based on the idea that a spline surface is fitted through the noisy bathymetry data and all soundings, which lie too far from the surface are rejected. Similar to the statistical outlier filter, it requires two passes trough the data set. In a first pass the (statistically) large blunders are removed until a certain quality criterion is met. In the second pass a refined spline surface is created to flag the noisy soundings (*How-to Qimera: Spline Filtering details* 2020). A handy feature of the spline filter implementation in Qimera is that it provides spline filter configurations according to the IHO specifications for hydrographic surveys as defined in IHO (2020). The filters used in the implemented program partially also use statistical methods to detect errors, but work more at the local neighborhood level. This has the disadvantage that the MBES resolution depends on the water depth and thus the neighborhood parameters change over a data set.

According to Le Deunf et al. (2020), the CUBE filter is one of the most popular filters in bathymetric data processing. Several commercial software suites implement the CUBE filter which also partly biases its popularity. The algorithm is described in Calder and Rice (2011) (as cited in Le Deunf et al., 2020). It consists of three stages: In a first stage, each sounding is associated with its TPU[4]. Then, for each node of a grid, a hypothesis is established where the seafloor is most likely located. In the third stage, the operator dissolves any ambiguities by conforming the best fitting hypothesis. This approach may be classified as semi-automatic, as it requires the hydrographer's interaction. This has the advantage that the blunder rejection does not solely depend on the subjective perception of a human while simultaneously complicated blunder agglomerations can still be manually addressed.

Backscatter processing

FMGT requires both the georeferenced bathymetry and the backscatter data. Kongsberg ALL is the only manufacturer-specific format that can be used directly to provide FMGT with both pieces of information in one file. However, since the ALL format does not provide the ability to store accept/reject flags along with the bathymetric information, it may be necessary to first clean up the bathymetry and then export/import the cleaned bathymetry along with the raw backscatter (*FMGT Quick Start Guide* 2020). In the implemented program, only the uncleaned bathymetry can be used to georeference the backscatter beam time series. Since the beams are concatenated between bottom detections, the algorithm relies on bathymetric information being available for each beam. On the other hand, it

[3]Combined Uncertainty and Bathymetry Estimator
[4]Total Propagated Uncertainty

may be questionable how representative beam time series are if they are associated with a blunder.

FMGT offers three possible outputs for backscatter image processing: Mosaic, statistics and ARA[5]. Since the implemented program computes only the backscatter mosaic, the comparison will focus solely on that. Already in the data comparison between the implemented program and FMGT, some differences were noticed. As deduced, the basic georeferencing algorithm probably differs slightly. However, the major difference is imposed by the radiometric correction which is applied in FMGT but not in the implemented program. Following the description of the initial Geocoder implementation by Fonseca and Calder (2005), each sample is first corrected for the variable acquisition gains, power levels and pulse widths. Subsequently, a range of individual radiometric corrections is applied, whereby the flat seafloor assumption used to enable real-time processing is replaced by the actual bathymetry measured by the MBES. Thereby, the actual ensonified area of the seafloor and also the effective incident angle are computed and the latter is used to correct the Lambert's law correction applied during acquisition. Afterwards the beam pattern is corrected on a ping by ping basis under the consideration of the angular responses around each ping.

In addition to the radiometric corrections, Geocoder also removes the speckle by using a morphological median filter with a percentile threshold. The mapping of the backscatter samples in mosaic space is then conducted by using the so called homography. In an attempt to summarize the more complex algorithm, one may say that a "rectangle in the track coordinate system is transformed to a quadrilateral in the projection coordinate system" (Fonseca and Calder, 2005). The pixel inside the quadrilateral is then assigned a backscatter value interpolated from the initial four samples. Afterwards, an inverse-mapping is applied to the sample set in order to avoid aliasing effects. Both despeckling and antialiasing attempt to remove the noise in the data while preserving the large structures. This might have partially led to the effect perceived as smeared, but it is partly desired. In the final mosaic algorithm, again a rather complex method called feathering is applied, which is meant to produce seamless transitions between overlapping survey lines by inter alia considering the quality factor of a sample (Fonseca and Calder, 2005).

The outlined workflow, which is at least similarly built into FMGT, is so complex that an actual comparison makes little sense. However, it does give an impression of how big the difference is between creating a backscatter mosaic that is visually appealing and one that is physically meaningful. Even if one does not aim for a similar sophisticated level of radiometric corrections, there is definitely room for improvement.

5.2. Potential improvements and outlook

Since several components are involved in the program implemented in this thesis, it makes the most sense from a purely structural point of view to derive possible improvements along the processing flow. When creating the bathymetry point cloud, the identified deficits were the missing sensor corrections. Although it is an option to start with the raw bathymetry data, it is not considered the most important improvement at this point, mainly because of the high effort it requires. Of course, some correction possibilities are not feasible when using the processed XYZ instead of the raw range and angle bathymetry, but the achievable results are comparably sound if good acquisition practices are ensured. Corrections that can and certainly should still be implemented are primarily

[5]Angle Range Analysis

the tidal correction, but also navigation processing. In fact, these two improvements are considered the most important for bathymetry processing. Since there is no tidal data in the Kongsberg ALL raw files, additional tidal data are needed. They would have to be provided before the soundings are located. Then, similar to the navigation, they could be interpolated at the MBES ping time and the appropriate depth correction added to the measured beam depth, simultaneously with the transducer depth. This sounds simple, but it actually requires user interface customization as well. Additionally, consideration must be given to how the tidal data should be formatted, or whether the tidal data reader can be adapted to different formats. What might also be worth considering in this context is the ‘height’ datagram, which provides height information to a predefined geoid model. It was not found in the available Kongsberg data (EM 122 and EM 710), but in general it might be interesting to evaluate if it could be used to obtain the depth of the seafloor with respect to the desired vertical reference.

The navigation processing is also considered an important improvement. Again, assuring good survey practice should avoid systematic navigation errors. However, in contrast to for example the motion sensor, a GNSS antenna depends on its surroundings. Just as the MBES, GNSS is an extremely complex measurement set up with various potential sources of error. Without focusing too much on errors which might occur in the atmosphere and at the satellites, there are still a couple of sources left, such as receiver noise, multipath, signal shadowing (by bridges in harbors). For both test data sets (from RV Sonne and RV Heincke), the surroundings did not really impose a source of error. However, there was a problem with the position data streaming that caused the required positions of one file to be partially written to another file. Actually, this shows quite well one of the decisions which have to be faced when attempting the navigation processing: Should the entire track of a data set be processed coherently or rather on a file by file basis. The latter is easier to implement but would not coherently resolve issues that effect several files. This could be avoided by collecting the navigation data from all the files of the data set to process them together. This approach, however, requires solutions on how to efficiently handle the navigation data and the associated bathymetric information. Irrespective of how to handle the data, it should also be considered what processing tools would actually be required such as the visualization (probably 2D would be enough), possible filters versus manual options, etc.

As for bathymetry filtering, the main structure is quite adaptable, but also depends on what is provided in PDAL. While not acute, it would always be worthwhile to update the components of the filter pipeline as PDAL evolves. In particular, it is assumed that a surface-based and/or a cluster-oriented filter would greatly improve the bathymetry filter pipeline. While at this point not tested, especially the PDAL planefit and the dbscan filters for clusters should be considered for adaptation, since they are assumed to address some of the deficits found. Another thing that should at least be mentioned is the mbio reader stage of PDAL, which is intended for bathymetric data and relies on the MB-Systems library. At this point, the mbio reader can only be run as a dynamic plugin under Linux/OS. This would generally be difficult to implement under Windows and would therefore exclude many potential users. In addition, it offers configurations for various MBES manufacturers, but not for the processing itself. All in all, it shows that PDAL is aware of the hydrographic community and may develop other tools intended for hydrographic data. In this case, it would greatly improve the overall pipeline.

The major drawback of the backscatter processing is the lack of a radiometric correction. Depending on whether the focus lies on bathymetry or backscatter processing, this can also be considered one of the important improvements. The quality of the RV Sonne data processing results heavily relied on the complex corrections which were applied by Kongsberg during acquisition. The result was a backscatter mosaic that was at least visually appealing. However, it is assumed that it is physically not reliable/meaningful and can not be quantitatively combined with independent backscatter data. In the comparison with QPS Fledermaus's FMGT, it was shown how complex a proper radiometric correction can be. While these corrections are certainly not easy to achieve, at least simpler approaches should be implemented. Hammerstad (2000) describes some of the 'things' that could be done.

In terms of visualization, Potree has proven to be a very valuable tool. At the moment, the default configuration is used, which has led to some impractical side effects. Among other things, the classifications still correspond to the default LAS classes and not to the bathymetry filters. Additionally, no backscatter can be displayed until now. Either an additional visualization for the backscatter mosaics would have to be offered or alternatively and preferably the Potree viewer would have to be configurated accordingly.

Without enumerating in detail all the possibilities of improvement, the most important ones are summarized as follows, according to their estimated relevance:

1. Tide correction of bathymetry
2. Simple approaches of radiometric corrections for the backscatter processing
3. Potree configuration for backscatter data and bathymetry filter classification
4. Navigation processing

The results of the program implemented in this thesis confirm the general feasibility of the chosen approach. When working with commercial or even scientific software, one usually obtains a bathymetric point cloud or backscatter mosaic without having to think too much about what was necessary to get there. While it is not always necessary to focus on the required background processing steps, otherwise, it has proven to be very educational for understanding the relationships inherent in MBES. At the beginning of this work, this was a chance that was rather underestimated. In fact, however, the combination of Python and Jupyter Notebook provides an ideal platform for students to learn the basic principles of MBES in explanatory texts while being able to apply and test what they have learned with interactive widgets and the underlying Python code. It also provides a platform to explicitly illustrate and demonstrate each step of the MBES processing chain. It should be noted that despite all this, a clear structure has to be maintained. For example, bathymetry cannot be filtered before it has been merged.

A possible application in this respect could be the adaptation of a processing workflow for the HCU research vessel DVocean. For such an educational approach, it would be worth considering not tying all Python code to widgets. The added value would be on the one hand for the students to gain experience in programming, on the other hand the workflow would be more flexible and the approach more widely scalable. Widgets would still make sense, but only for frequently invoked functions. To further extend the educational approach, one could focus much more on the display capabilities of Jupyter Notebook and tie it together with the pure documentation.

6. Conclusion

The aim of the thesis was to explore the interactive processing possibilities of Kongsberg MBES bathymetry and backscatter data with Jupyter Notebook and Python. The program which was implemented for this purpose is subdivided into three components: Bathymetry preprocessing, backscatter preprocessing, and filters and corrections.

The bathymetry preprocessing is based on the Kongsberg XYZ beam data. Some of the IHO recommended corrections can therefore not be applied. At the present, sensor corrections have to be applied externally and then imported via an ASCII interface to the processing notebook. When comparing with reference data processed in commercial software (Qimera), it has been found that there is a minimal systematic deviation in the preprocessing. The cause can most likely be limited to a difference in the individual sensor corrections of Kongsberg and Qimera.

The backscatter preprocessing is based on the Kongsberg seabed image data. The implemented georeferencing algorithm works in a convincing manner and allows a high spatial resolution of the beam time series backscatter data. Since no post-processing radiometric corrections were applied, it is not guaranteed that the backscatter of the implemented software is physically meaningful. However, it has been shown that the extensive dynamic gain applied by Kongsberg in real-time gives very satisfactory results. The backscatter mosaics created this way were at least visually very appealing. If the backscatter data is exploited in a self-contained manner, it can presumably provide the basis for a seafloor classification. To validates its absolute utility, more comprehensive comparisons would need to be conducted including data from independent sensors.

The filters and corrections were finally implemented in Jupyter Notebook based on a combination of the three open source point cloud processing tools PDAL, Entwine and Potree. All three tools are optimized for processing point clouds and the associated data volumes. In this way, the processing speed could be improved. The starting point for this is Entwine's octree-based EPT format. For the processing itself, IPython widgets are used to interactively parameterize and execute PDAL pipelines for bathymetry filtering and backscatter corrections, respectively. The bathymetry filtering can be checked via the web-based Potree viewer. This step is not yet possible for the backscatter processing, since Potree cannot display amplitudes by default. Both, bathymetry and backscatter processing, could be extended functionally. The further use of the notebook would lead the way for the extension.

At present Python is one of the programming languages of choice for programmers coming from a data science and application background rather than having an excessive education in computer science. It is easy to implement, provides an almost pseudo-code like readability and it has a vast community which helps to solve almost any problem. Thereby the integration of performance oriented C++ libraries essentially helped to handle the large data amounts during the filters and corrections. The approach of using Jupyter Notebook as a graphical user interface is promising, but some aspects need to be considered. Jupyter Notebook offers an interesting way to combine data processing in the form of code in the background directly with documentation. In this context, the use of widgets offers the possibility to create specific interfaces between the two, facilitating processing. It was found that excessive use of widgets can make a notebook quite cluttered and carries a high risk of undesirable behavior. When using Jupyter Notebook exclusively as a GUI, well-defined workflows are required that are parameterized with the widgets provided. For more complex setups, they have insufficient testing capabilities and can be executed improperly due to their flexibility.

Regarding the initial question, how well Jupyter Notebook and Python can be utilized for interactive processing of bathymetry and backscatter data: The implemented program is by far not as comprehensive as the available commercial or scientific software. However, in relation to the time given within the frame of this thesis, the general feasibility could be demonstrated and the achieved performance and results are promising.

7. Bibliography

Artilheiro, F.M.F. (1998). “Analysis and Procedures of Multibeam Data Cleaning for Bathymetric Charting”. Master’s thesis. New Brunswick, Canada: University of New Brunswick.

ASPRS (2018). *LAS Specification*.

Beaudoin, J. et al. (2002). *Geometric and radiometric correction of multibeam backscatter derived from Reson 8101 systems*.

Bell, A., B. Chambers, and Butler H. (2020). *PDAL Documentation: Version 2.2*. URL: https://pdal.io/ (visited on 12/13/2020).

Brown, C. J. et al. (2015). “Backscatter measurement by bathymetric echo sounders”. In: *Backscatter measurements by seafloor–mapping sonars*. Ed. by X. Lurton and G. Lamarche, pp. 79–106.

Calder, B. R. and G. Rice, eds. (2011). *Design and implementation of an extensible variable resolution bathymetric estimator*.

Carbonnelle, P. (2020). *PYPL: PopularitY of Programming Language index*. URL: https://pypl.github.io/PYPL.html (visited on 12/13/2020).

Chen, Z. et al. (2012). “Upward-fusion urban DTM generating method using airborne Lidar data”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 72, pp. 121–130. ISSN: 09242716.

de Jong, C. D. (2002). *Hydrography*. 1. ed. Vol. 33. Series on mathematical geodesy and positioning. Delft: Delft Univ. Press. ISBN: 90-407-2359-1.

FMGT Quick Start Guide (2020). URL: https://confluence.qps.nl/fledermaus7/reference-manual/fmgt/fmgt-quick-start-guide (visited on 12/13/2020).

Fonseca, L. and B. R. Calder (2005). *Geocoder: An Efficient Backscatter Map Constructor*.

Grządziel, A. and M. Wąż (2018). “The Invention and Developing of Multibeam Echosounder Technology”. In: *Polish Hyperbaric Research* 62.1, pp. 33–42.

Hammerstad, E. (2000). *Backscattering and Seabed Image Reflectivity*.

Han, S. (2018). “Towards Efficient Implementation of an Octree for a Large 3D Point Cloud”. In: *Sensors (Basel, Switzerland)* 18.12.

Hare, R. (1995). “Depth and position error budgets for multibeam echosounding”. In: *International Hydrographic Review* LXXII.2, pp. 37–69.

Hobu, Inc. (2020). *Entwine Documentation: Version 2.1*. URL: https://entwine.io (visited on 12/13/2020).

Hovem, J. M. (2013). “Ray Trace Modeling of Underwater Sound Propagation”. In: *Modeling and Measurement Methods for Acoustic Waves and for Acoustic Microdevices*. Ed. by M. G. Beghi. InTech. ISBN: 978-953-51-1189-4.

How-to Qimera: Migrate a Qimera Project to FMGT (2020). URL: https://confluence.qps.nl/qimera/2.0/en/how-to-qimera-working-with-kongsberg-all-data-127670562.html (visited on 12/13/2020).

How-to Qimera: Spline Filtering details (2020). URL: https://confluence.qps.nl/qimera/latest/en/how-to-qimera-spline-filtering-details-42316820.html (visited on 12/13/2020).

How-to-FMGT: FMGT Supported Data Formats (2020). URL: https://confluence.qps.nl/fledermaus7/reference-manual/fmgt/fmgt-supported-data-formats (visited on 12/13/2020).

IHO (2005). *Manual on Hydrography: M-13*. Monaco: International Hydrographic Bureau.

– (2008). *Standards for Hydrographic Surveys*. Monaco.

– (2020). *Standards for Hydrographic Surveys*. Monaco.

inVerita (2020). "Top 10 programming languages gaining momentum in 2020 according to GitHub". In: *Level Up Coding*. URL: https://levelup.gitconnected.com/top-10-programming-languages-gaining-momentum-in-2020-by-github-33d099cef8c0 (visited on 12/13/2020).

Jupyter Team (2015). *Jupyter Notebook Documentation: Version 6.1.5*. URL: https://jupyter.org (visited on 12/13/2020).

Karney, C.F.F. (2015). *geographiclib: Python implementation of the geodesic routines in GeographicLib (C++)*. URL: https://geographiclib.sourceforge.io/html/python/.

Kennedy, P. (2016). *pyall: Based on ALL file version October 2013*. URL: https://github.com/pktrigg/pyall.

Kinne, S. et al. (2019). *SONNE-Berichte: Cruise SO268-3: Vancouver – Singapore, 30.05. – 5.07.19*.

Kongsberg (2007). *Kongsberg EM 710: Multibeam echo sounder: Product Description*.

– (2011). *Kongsberg EM 122: Multibeam echo sounder: Product Description*.

– (2018). *Kongsberg EM Series - Multibeam echo sounders: EM Datagram formats: Instruction manual*.

– (2019). *SIS: Seafloor Information System: Reference Manual*.

Le Deunf, J. et al. (2020). "A Review of Data Cleaning Approaches in a Hydrographic Framework with a Focus on Bathymetric Multibeam Echosounder Datasets". In: *Geosciences* 10.254.

Lenart, A. S. (2011). "Solutions of Direct Geodetic Problem in Navigational Applications". In: *International Journal on Marine Navigation and Safety of Sea Transportation* 5.4, pp. 527–532.

Löwis, M. von and N. Fishbeck (1997). *Das Python-Buch: Referenz der objekorientierten Skriptsprache für GUIs und Netzwerke*. 1. Aufl. Bonn and Reading, Mass. [u.a.]: Addison-Wesley-Longman. ISBN: 3827311101.

Lurton, X. (2002). *An introduction to underwater acoustics: Principles and applications*. New York: Springer. ISBN: 3-540-42967-0.

Manning, C. (2016). *Entwine*.

MDN contributors (2020). *WebGL: 2D and 3D graphics for the web*. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API (visited on 12/13/2020).

Open Source Initiative (2020). *Licenses & Standards*. URL: https://opensource.org/licenses (visited on 12/13/2020).

PDAL (2019).

PDAL (2020). *About: Point Data Abstraction Library*. URL: https://pdal.io/about.html#about (visited on 12/13/2020).

PDAL Contributors (2020). *PDAL Point Data Abstraction Library*.

Perez, F. and B. E. Granger (2007). "IPython: A System for Interactive Scientific Computing". In: *Computing in Science and Engineering* 9.3, pp. 21–29. URL: https://ipython.org.

PSF (2020). *What is Python? Executive Summary*. URL: https://www.python.org/doc/essays/blurb/ (visited on 12/13/2020).

Qimera: Sonar Source Data Import Capabilities (2020). URL: https://confluence.qps.nl/qimera/latest/en/qimera-sonar-source-data-import-capabilities-202312454.html#id-.QimeraSonarSourceDataImportCapabilitiesv2.2-ALL (visited on 12/13/2020).

Rusu, R. B. and S. Cousins, eds. (2011). *3D is here: Point Cloud Library (PCL)*. Shanghai, China.

Rusu, R. B., Z. C. Marton, et al. (2008). "Towards 3D Point cloud based object maps for household environments". In: *Robotics and Autonomous Systems* 56.11, pp. 927–941. ISSN: 09218890.

Schimel, A.C.G., J. Beaudoin, A. Gaillot, et al. (2015). “Processing backscatter data: From datagrams to angular response and mosaics”. In: *Backscatter measurements by seafloor–mapping sonars*. Ed. by X. Lurton and G. Lamarche, pp. 133–164.

Schimel, A.C.G., J. Beaudoin, I. M. Parnum, et al. (2018). “Multibeam sonar backscatter data processing”. In: *Marine Geophysical Research* 39, pp. 121–137.

Schütz, M. (2016). “Potree: Rendering Large Point Clouds in Web Browsers”. Diploma. Vienna: Vienna University of Technology.

SeaBeam (2000). *Multibeam Sonar: Theory of Operation*. East Walpole MA.

Skywell Software (2020). “Github Most Popular Languages: Will Python Overtake Java”. In: *Medium*. URL: https://skywellsoftware.medium.com/github-most-popular-languages-will-python-overtake-java-d609d3d4f303.

van Rossum, G. (1998). *Glue It All Together With Python: Workshop on Compositional Software Architecture*. Monterey CA. URL: https://www.python.org/doc/essays/omg-darpa-mcc-position/ (visited on 12/13/2020).

Vanderplas, J. T. (2017). *Python data science handbook: Essential tools for working with data*. Beijing: O’Reilly. ISBN: 978-1-491-91205-8.

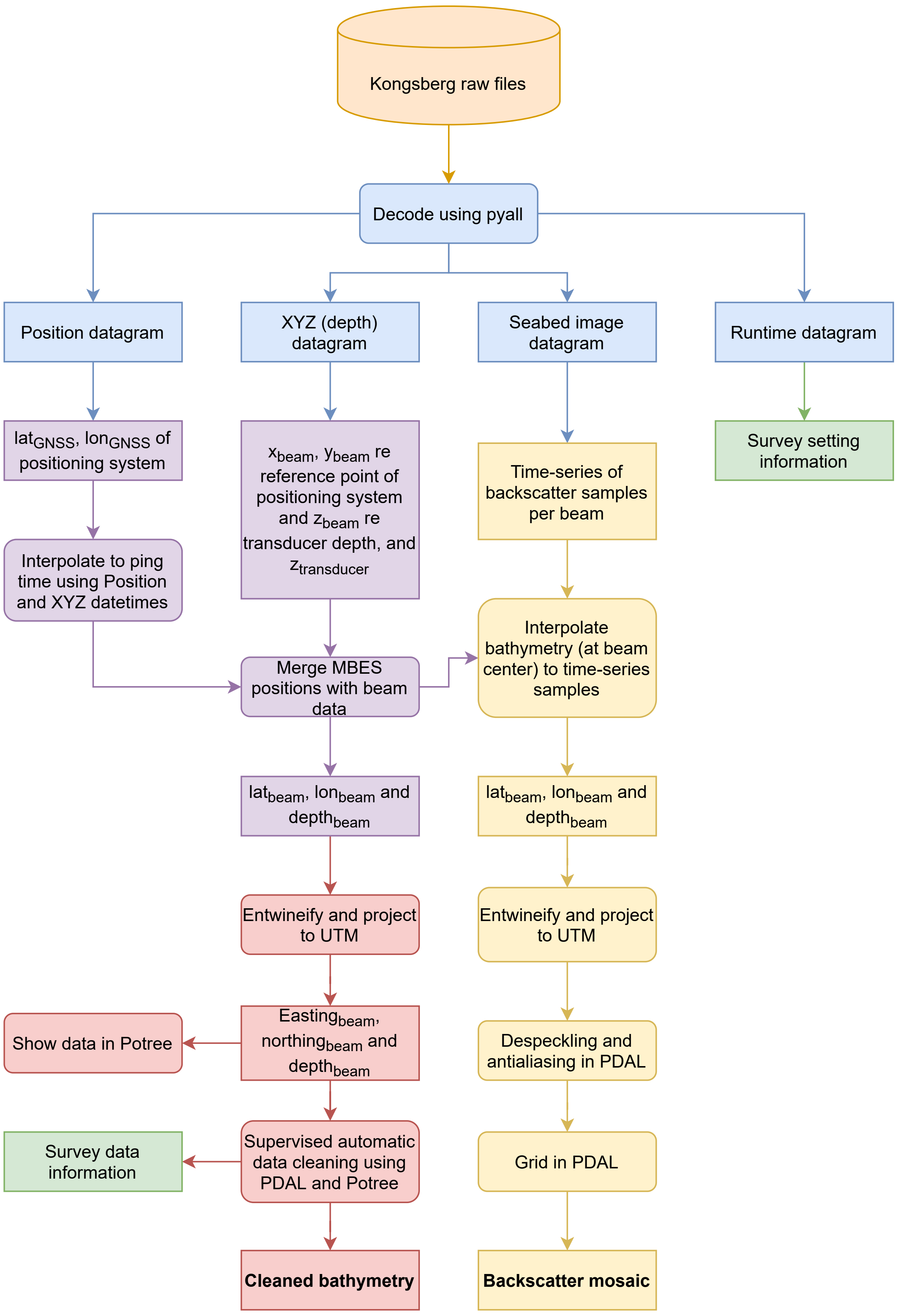
Vincenty, T. (1975). “Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations”. In: *Survey Review* XXIII.176, pp. 88–93.

Weber, T. C. and X. Lurton (2015). “Background and Fundamentals”. In: *Backscatter measurements by seafloor–mapping sonars*. Ed. by X. Lurton and G. Lamarche, pp. 25–52.

What is open source? (2020). URL: https://opensource.com/resources/what-open-source (visited on 12/13/2020).

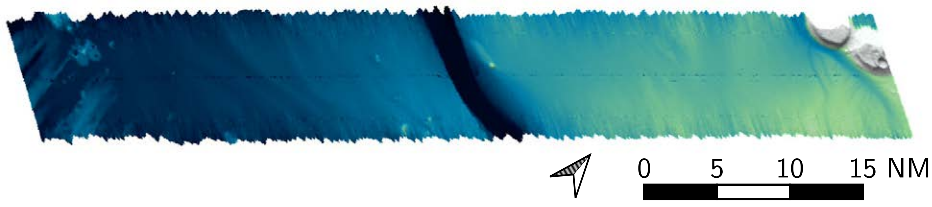
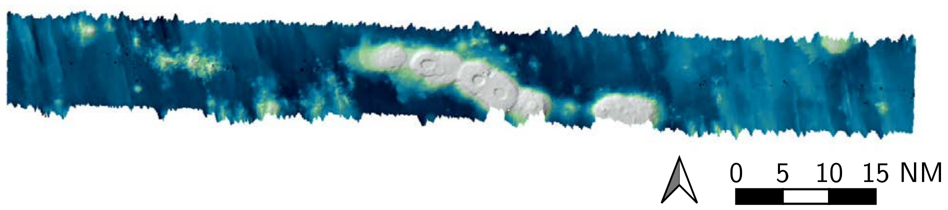
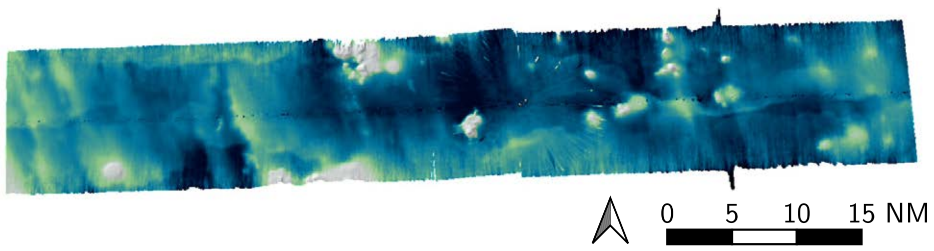
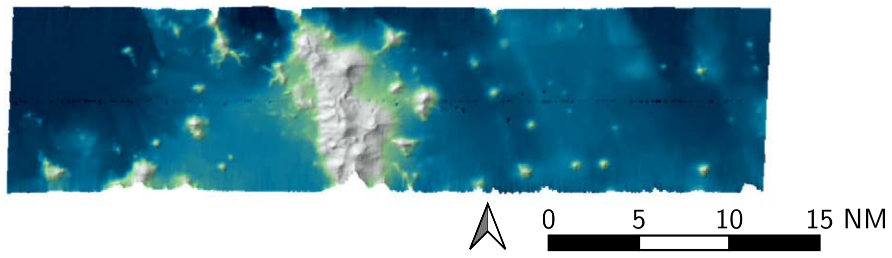
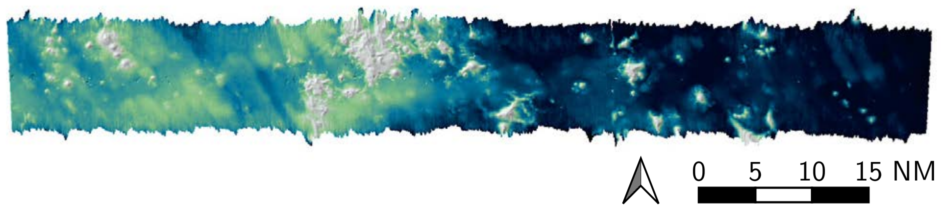
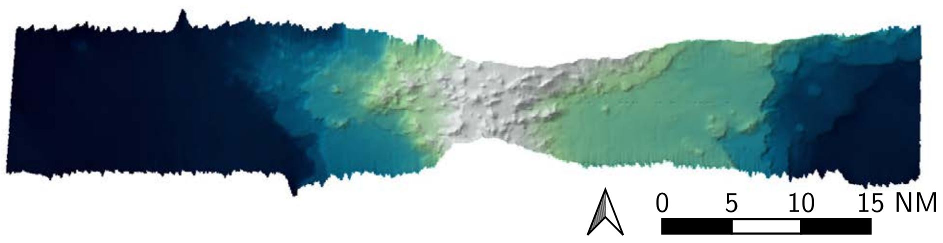
Zhou, Q.-Y., J. Park, and V. Koltun (2018). “Open3D: A Modern Library for 3D Data Processing”. In: *arXiv*.

A. Project structure

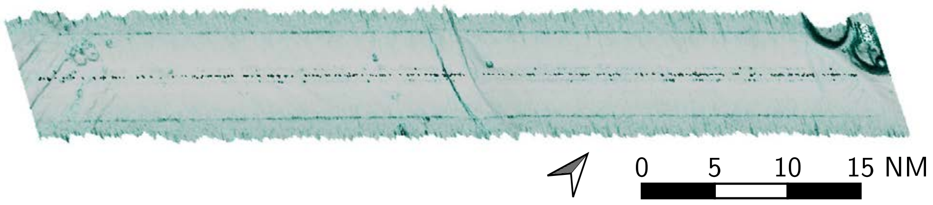
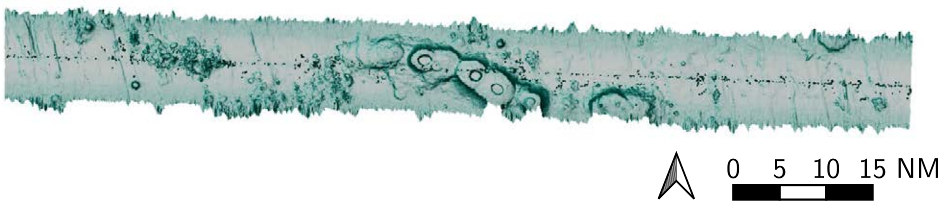
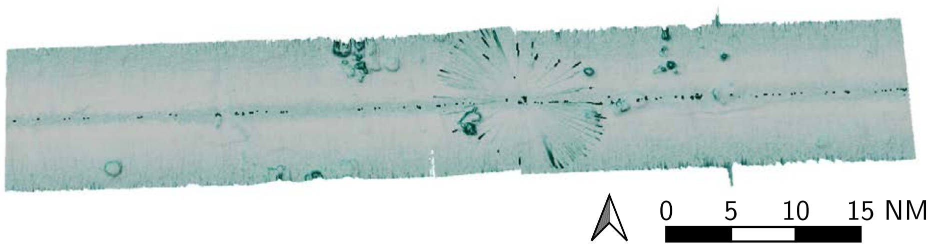
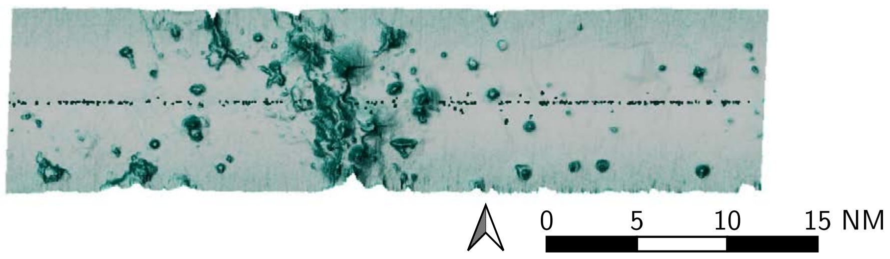
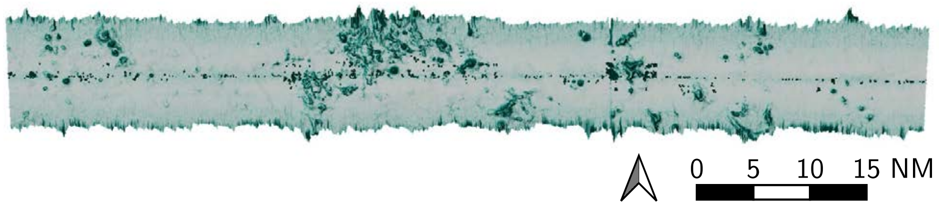
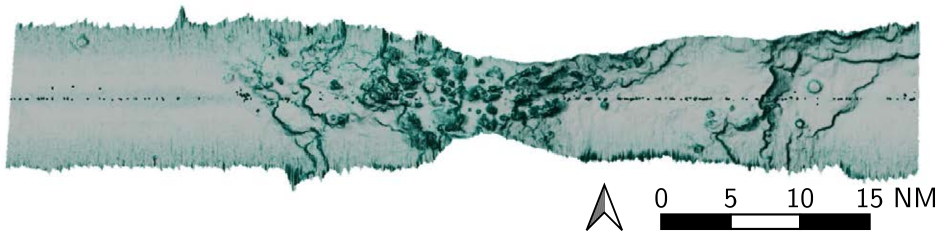

Kongsberg raw files
Decode using pyall
Position datagram
XYZ (depth) datagram
Seabed image datagram
Runtime datagram
lat_{GNSS}, lon_{GNSS} of positioning system
x_{beam}, y_{beam} re reference point of positioning system and z_{beam} re transducer depth, and $z_{transducer}$
Time-series of backscatter samples per beam
Survey setting information
Interpolate to ping time using Position and XYZ datetimes
Merge MBES positions with beam data
Interpolate bathymetry (at beam center) to time-series samples
lat_{beam}, lon_{beam} and $depth_{beam}$
lat_{beam}, lon_{beam} and $depth_{beam}$
Entwineify and project to UTM
Entwineify and project to UTM
Show data in Potree
$Easting_{beam}$, $northing_{beam}$ and $depth_{beam}$
Despeckling and antialiasing in PDAL
Survey data information
Supervised automatic data cleaning using PDAL and Potree
Grid in PDAL
Cleaned bathymetry
Backscatter mosaic

B. Raw SO268-3 data subsets

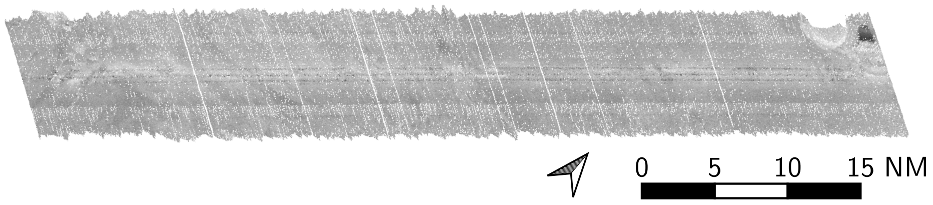
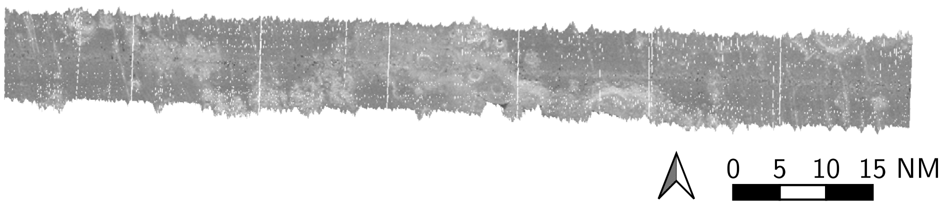
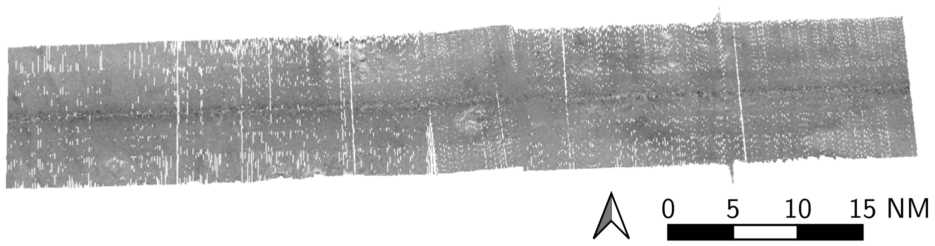
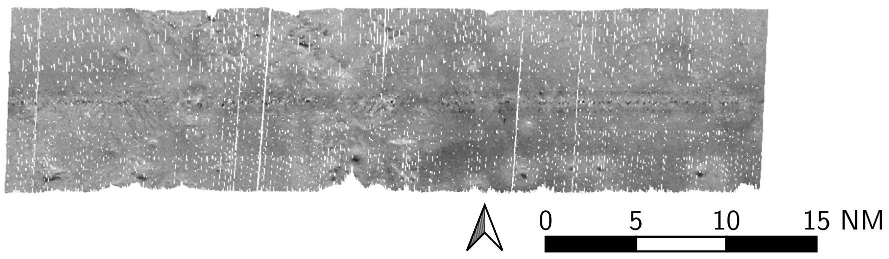
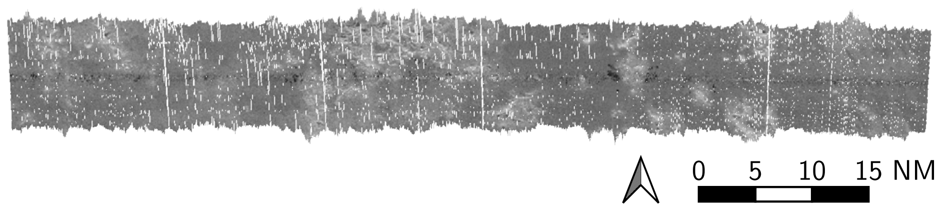
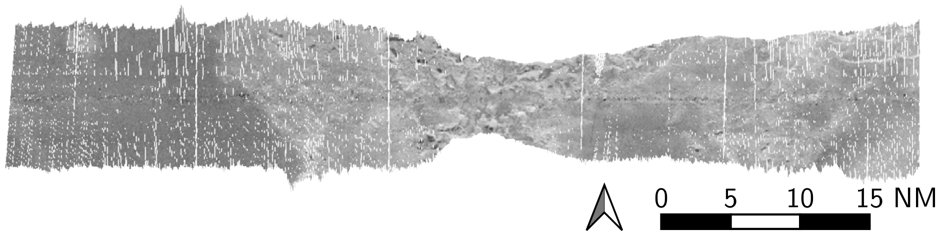
B.1. Bathymetry

Subset 1 Depth ⬜ -3345 m 🟦 -3598 m ⬛ -3852 m	
Subset 2 Depth ⬜ -4565 m 🟦 -5050 m ⬛ -5536 m	
Subset 3 Depth ⬜ -5572 m 🟦 -5690 m ⬛ -5809 m	
Subset 4 Depth ⬜ -4895 m 🟦 -5265 m ⬛ -5635 m	
Subset 5 Depth ⬜ -5033 m 🟦 -5235 m ⬛ -5437 m	
Subset 6 Depth ⬜ -2933 m 🟦 -4876 m ⬛ -6431 m	

B.2. Standard deviation

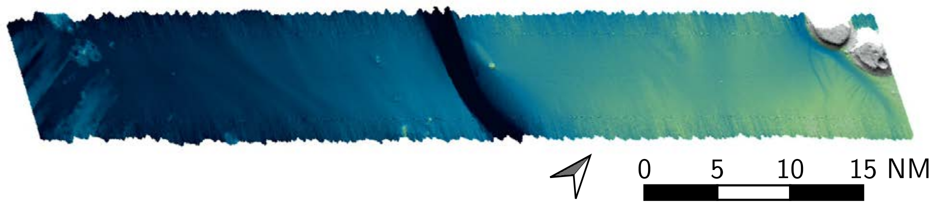
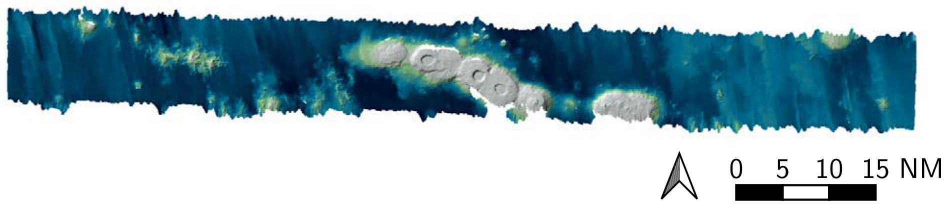
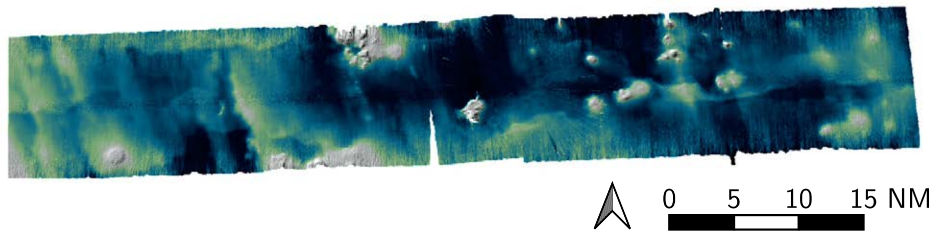
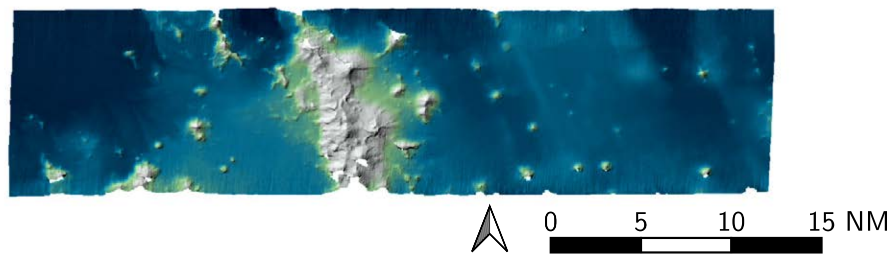
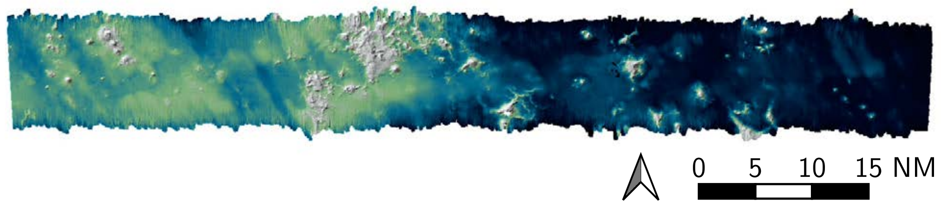
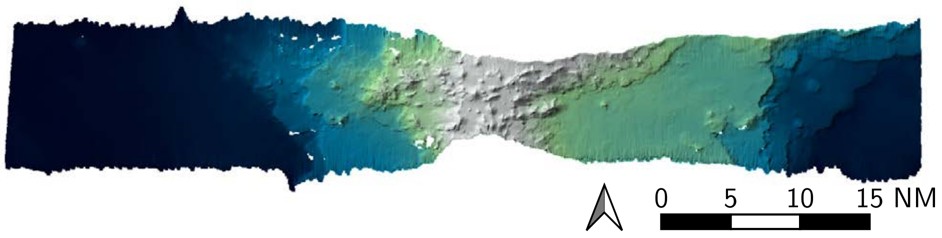
Subset 1 Std Dev 🟫 -45 m ⬜ 0 m 🟩 45 m	
Subset 2 Std Dev 🟫 -45 m ⬜ 0 m 🟩 45 m	
Subset 3 Std Dev 🟫 - 45 m ⬜ 0 m 🟩 45 m	
Subset 4 Std Dev 🟫 -45 m ⬜ 0 m 🟩 45 m	
Subset 5 Std Dev 🟫 -45 m ⬜ 0 m 🟩 45 m	
Subset 6 Std Dev 🟫 -45 m ⬜ 0 m 🟩 45 m	

B.3. Backscatter

Subset 1 BS strength ■ -79.9 dB □ -3.1 dB	
Subset 2 BS strength ■ -80.0 dB □ 3.2 dB	
Subset 3 BS strength ■ -57.0 dB □ -5.3 dB	
Subset 4 BS strength ■ -58.2 dB □ 2.6 dB	
Subset 5 BS strength ■ -60.4 dB □ 1.1 dB	
Subset 6 BS strength ■ -64.1 dB □ -1.0 dB	

C. Processed RV Sonne data subsets

C.1. Processed bathymetry

Subset 1 Depth ⬛ -3847 m 🟦 -3600 m ⬜ -3353 m	
Subset 2 Depth ⬛ -5537 🟦 -5050 m ⬜ -4563 m	
Subset 3 Depth ⬛ -5776 m 🟦 -5692 m ⬜ -5605 m	
Subset 4 Depth ⬛ -5633 m 🟦 -5266 m ⬜ -4899 m	
Subset 5 Depth ⬛ -5440 m 🟦 -5255 m ⬜ -5063 m	
Subset 6 Depth ⬛ -6435 m 🟦 -4683 m ⬜ -2931 m	

C.2. Difference to Qimera Bathymetry

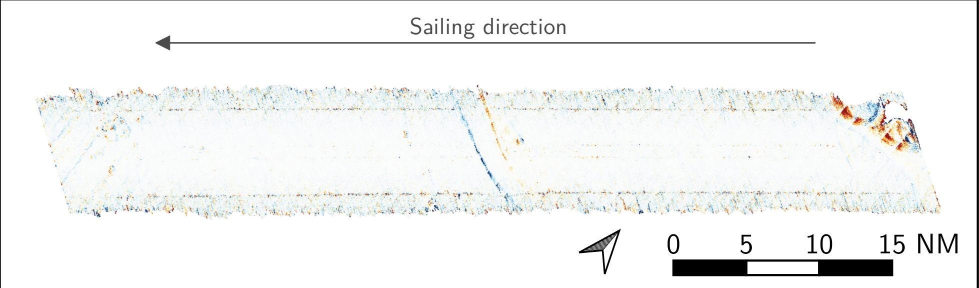
Subset 1

Difference

🟦 -25 m

⬜ 0 m

🟥 25 m



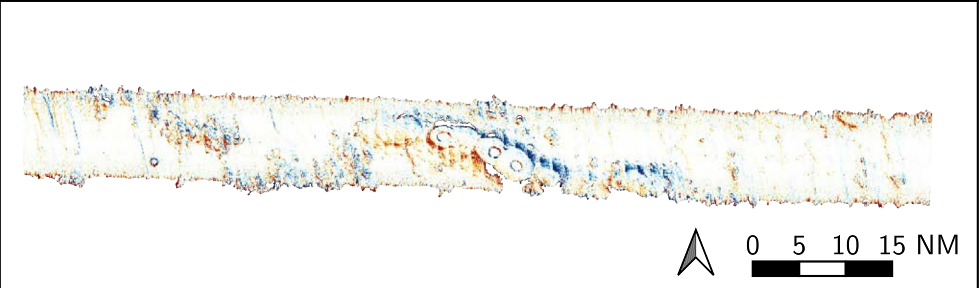
Subset 2

Difference

🟦 -25 m

⬜ 0 m

🟥 25 m



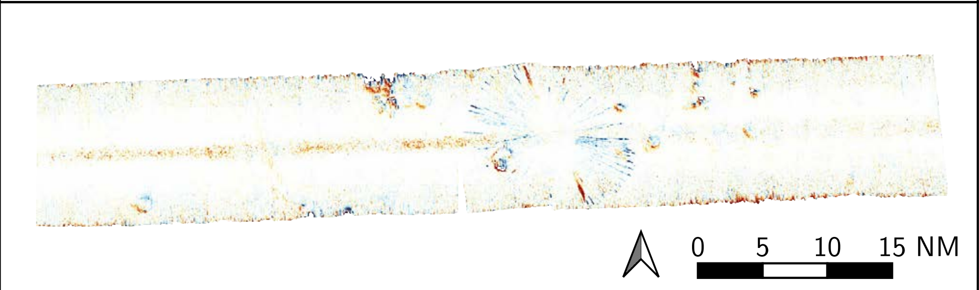
Subset 3

Difference

🟦 -25 m

⬜ 0 m

🟥 25 m



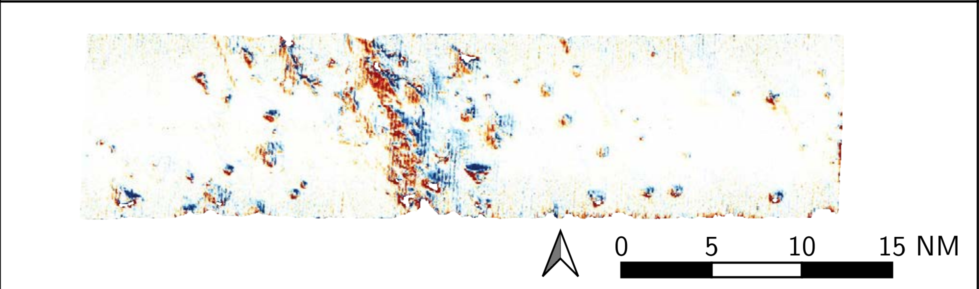
Subset 4

Difference

🟦 -25 m

⬜ 0 m

🟥 25 m



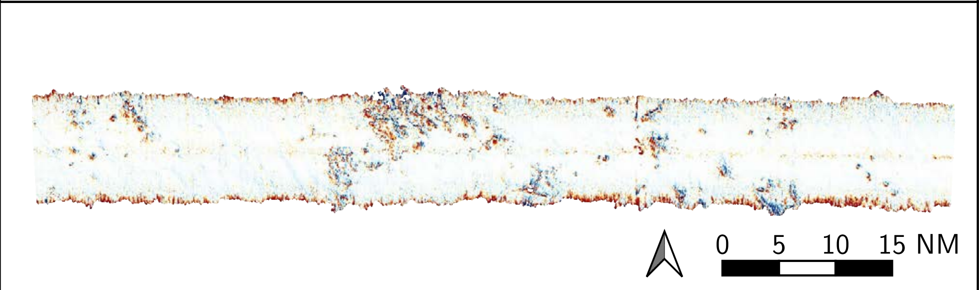
Subset 5

Difference

🟦 -25 m

⬜ 0 m

🟥 25 m



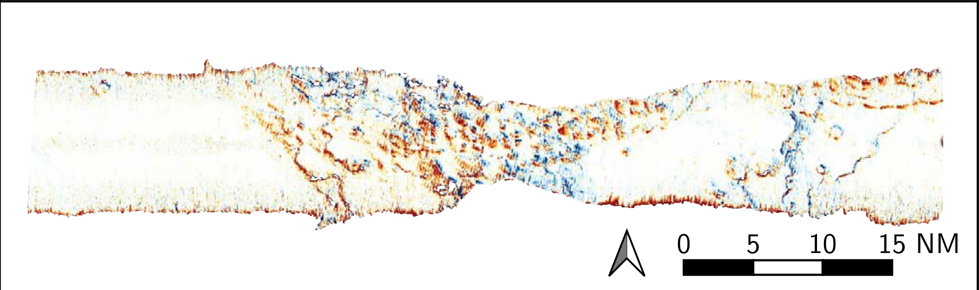
Subset 6

Difference

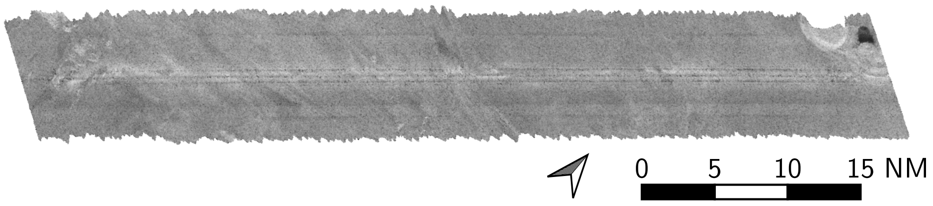
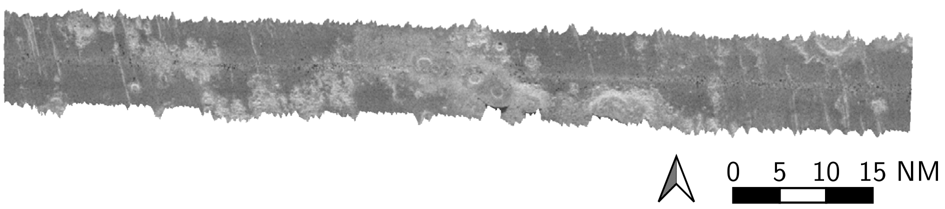
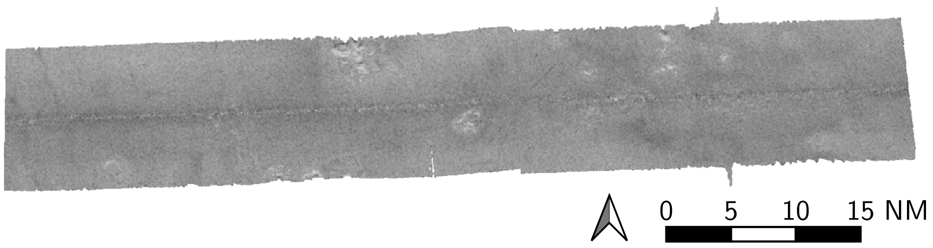
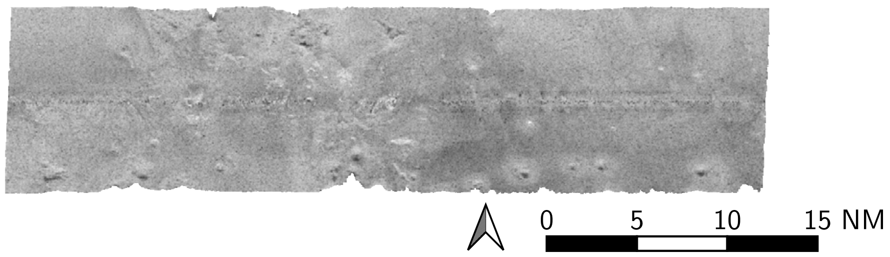
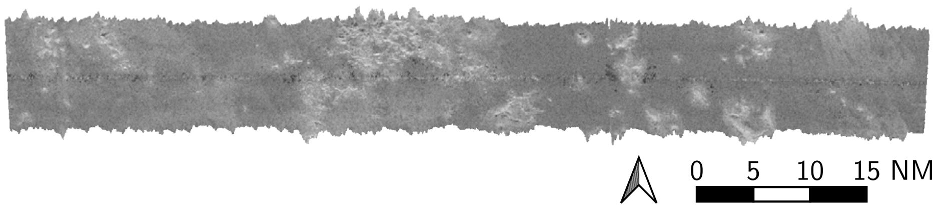
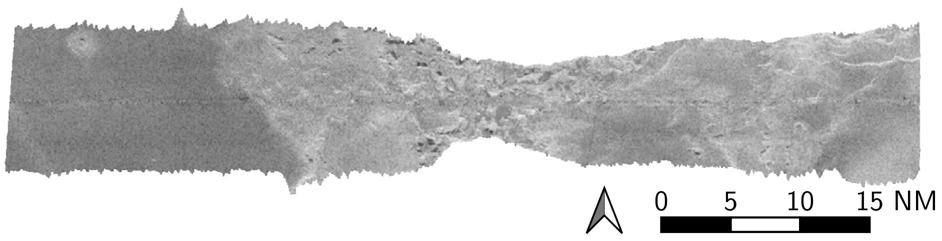
🟦 -25 m

⬜ 0 m

🟥 25 m



C.3. Processed backscatter

Subset 1 BS strength ■ -65 dB □ 0 dB	
Subset 2 BS strength ■ -65 dB □ 0 dB	
Subset 3 BS strength ■ -65 dB □ 0 dB	
Subset 4 BS strength ■ -65 dB □ 0 dB	
Subset 5 BS strength ■ -65 dB □ 0 dB	
Subset 6 BS strength ■ -65 dB □ 0 dB	

C.4. Difference to FM Geocoder backscatter

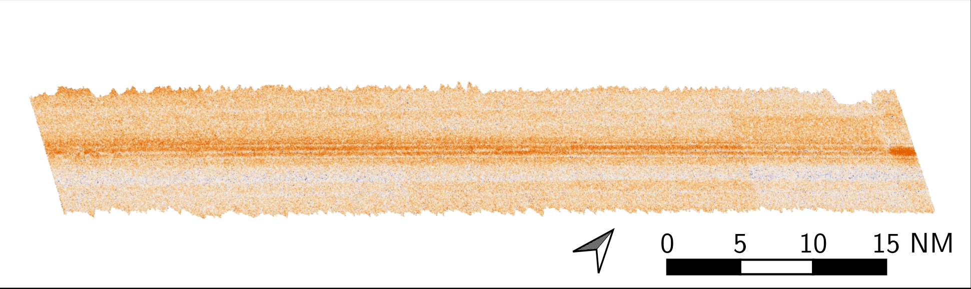
Subset 1

Difference

🟧 -15 dB

⬜ 0 dB

🟪 15 dB



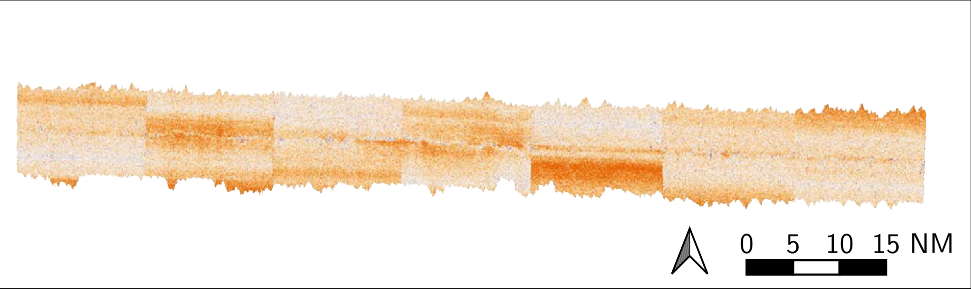
Subset 2

Difference

🟧 -15 dB

⬜ 0 dB

🟪 15 dB



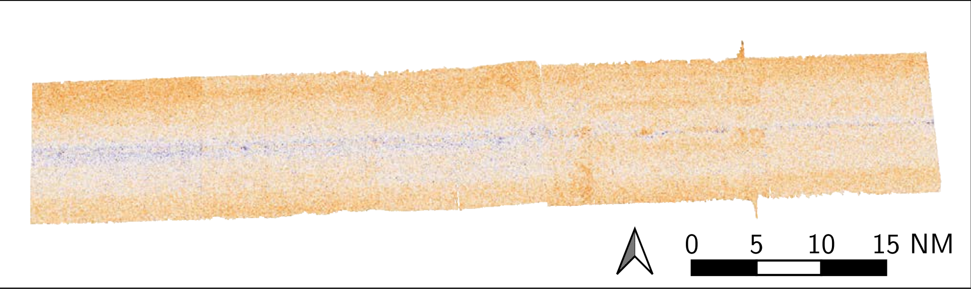
Subset 3

Difference

🟧 -15 dB

⬜ 0 dB

🟪 15 dB



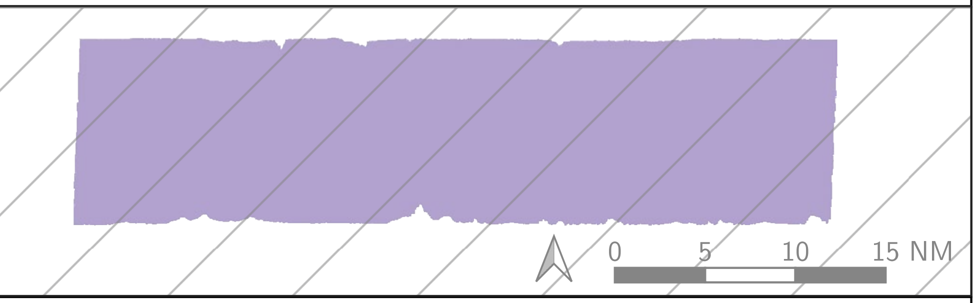
Subset 4

Difference

🟧 -15 dB

⬜ 0 dB

🟪 15 dB



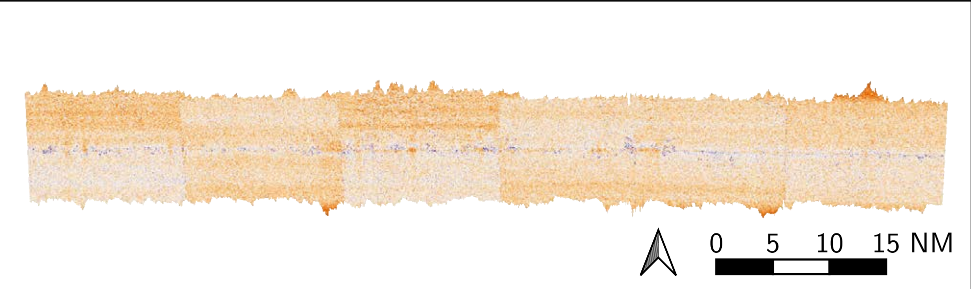
Subset 5

Difference

🟧 -15 dB

⬜ 0 dB

🟪 15 dB



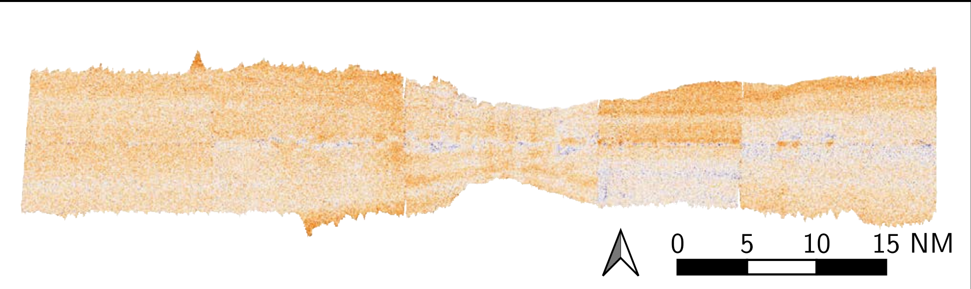
Subset 6

Difference

🟧 -15 dB

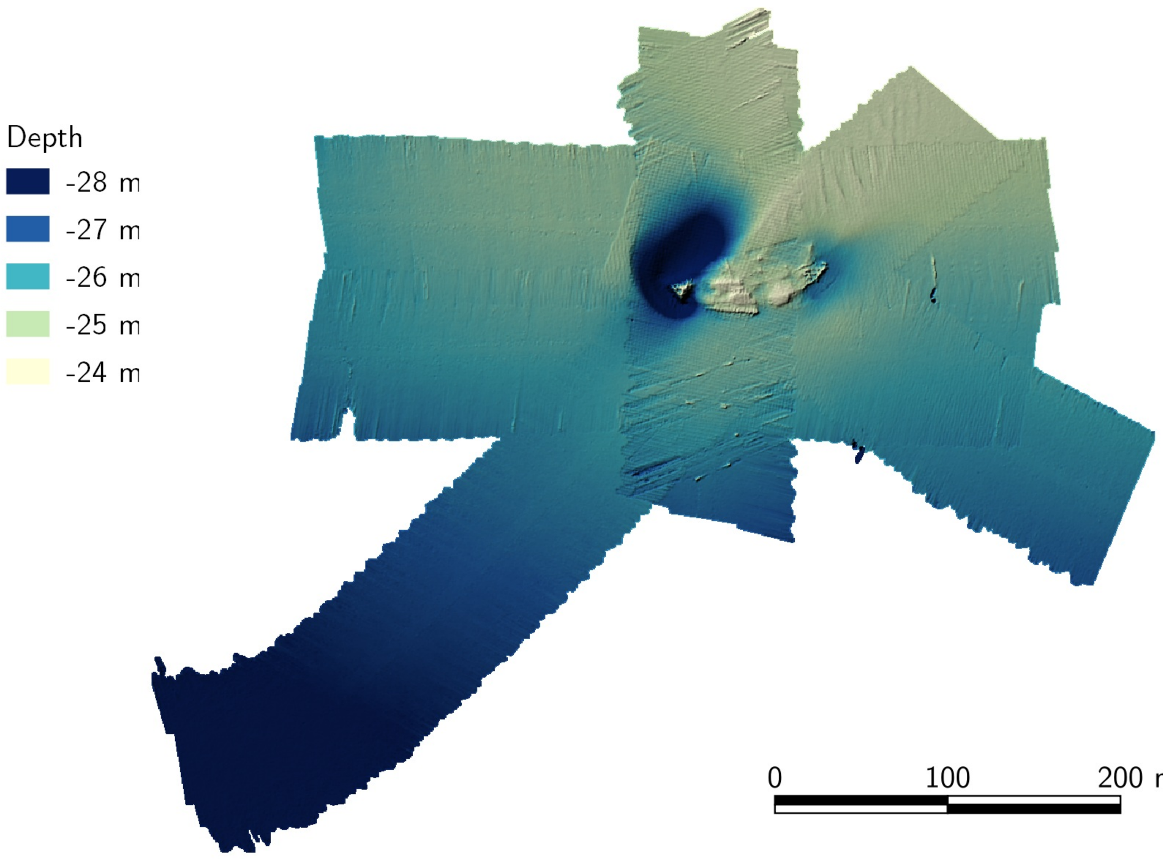
⬜ 0 dB

🟪 15 dB

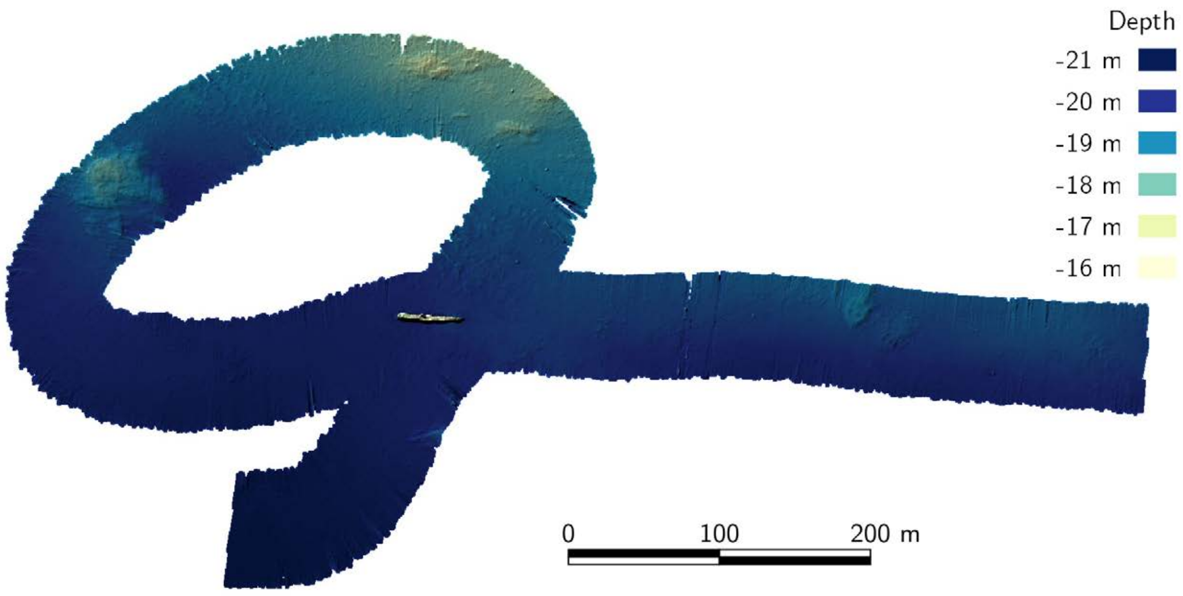


D. Processed RV Heincke data subsets

D.1. Wreck 1



D.2. Wreck 2



Sophie Andree wurde für ihre Masterarbeit, die sie an der HafenCity Universität Hamburg verfasst hat, mit dem DHyG Student Excellence Award 2021 ausgezeichnet.

Die Verarbeitung hydrographischer Daten – Bathymetrie und Rückstreuung – erfordert eine Reihe von Prozessen, um zuverlässige Informationen zu erhalten. Mehrere kommerzielle und wissenschaftliche Softwarelösungen sind für diesen Zweck verfügbar, aber die Nutzung ist nicht immer frei zugänglich. Dem soll durch den Einsatz von Open-Source-Bibliotheken begegnet werden. Die Kombination von Jupyter Notebook und Python hat großes Potenzial für die interaktive Datenverarbeitung.