UNIVERSITY OF POTSDAM

MASTER'S THESIS

Near-real time detection of lake ice using Sentinel-1 data

Author:
Felix KESSLER

Supervisor:
Dr. Annett Frick

Examiner:
Dr. Bodo Bookhagen

January 16, 2020

Declaration of Authorship

I, Felix Keßler, declare that this thesis my own work. All results that are shown were produced by myself using the given tools. Any work of others that has been used is quoted.

Signed:

Date:

Eidesstattliche Erklärung

Hiermit erkläre, Felix Keßler, dass diese Thesis von mir selbstständig ausgearbeitet und verfasst wurde. Alle Ergebnisse habe ich selbst mit den genannten Hilfsmitteln erarbeitet. Sollte die Arbeit Anderer genutzt worden sein, wurde dies entsprechend vermerkt.

Unterschrift:

Datum:

Abstract

The goal of this work is the development of an algorithm to detect freshwater ice in Scandinavia using Sentinel-1 data. The best result was gained by using time series analysis of *Ground-range detected* data. The lake wide mean of the ratio of the two bands (VV/VH) that get captured in the *Interferometric Wide Swath mode* shows an significant increase compared to the long-term mean once the ice is freezing. Using different programs and environments the process of downloading, processing and analysing the data could be automated.

Keywords— Sentinel-1, Radar, Ice, Remote Sensing, Scandinavia, Time series analysis, IW, GRD

Zusammenfassung

Ziel dieser Arbeit ist die Erstellung eines Algorithmus zur Detektion von Süßwassereis in Skandinavien unter Nutzung und Verarbeitung von Sentinel-1 Daten. Das beste Ergebnis wurde durch eine Zeitreihenanalyse der *Ground-Range detected* Daten erzielt. Das Verhältnis der zwei Bänder (VV/VH), welche im *Interferometric Wide Swath mode* aufgenommen werden, zeigt einen signifikanten Ausschlag im Vergleich zum langzeitlichen Mittel dieses Wertes. Verschiedene Programme und Umgebungen wurden genutzt, um den Ablauf von Download, Verarbeitung und Analyse automatisiert zu gestalten.

Schlüsselwörter— Sentinel-1, Radar, Eis, Fernerkundung, Skandinavien, Zeitreihenanalyse, IW, GRD

Acknowledgements

A lot of people supported me during my thesis and the years of studying before. I can not name them all, but the most important ones deserve to be mentioned here.
First of all: **Annett**. Thank you for giving me the opportunity to write my thesis but also for supporting me all the time. You helped me with subject-specific support when necessary and always conveyed the feeling that what I am doing is good and will produce a good result.
I also have to thank **Dr. Bodo Bookhagen** for being my supervisor and bringing in ideas as well as the introduction into the InSAR methodology.
Thank you **Hanna**, for supporting me throughout the last years of studying. Thank you for helping me with problems both related to the thesis as well as in other parts of life. I want to thank my parents as well, for supporting me the last 6 years throughout all stages of my studies.
Last but not least: My **colleagues at LUP**: Sandy, Randolf, Benni, Annika.... Thank you for nice conversations, both thematically relevant and less relevant. Thank you for countless coffee breaks and a big amount of fun while working.

Contents

List of Figures

List of Tables

List of Abbreviations

ASTER	**A**dvanced **S**paceborne **T**hermal **E**mission and Reflection **R**adiometer (Global Digital Elevation Model)
FLI	**F**riedrich-**L**öffler-**I**nstitute
GEE	**G**oogle **E**arth **E**ngine
GRD	**G**round-**R**ange **d**etected
IGB	Leibnitz **I**nsititut für **G**ewässerökologie und **B**innenfischerei (English: Leibnitz-institute for water ecology and freshwater fishery)
IPF	**I**nstrument **P**rocessing **F**acility
ISCE	**I**nSAR **S**cientific **C**omputation **E**nvironment
LTM	**L**ong **t**erm **m**ean
LUT	**L**ook **u**p **t**able
NASeR	**N**ear-real time **A**nalysis of **S**atellite data for **e**pidemiological **R**isk assesment
SAR	**S**ynthetic **A**perture **R**adar
SLC	**S**ingle **L**ook **C**omplex
SNAP	**Sen**tinel **A**pplication **P**latform
SNR	**S**ignal-to-**N**oise**Ratio**

1 Introduction

1.1 Background information

This work is done as a part of the NASeR-project (Near-real time Analysis of Satellite data for epidemiological Risk assessment) of the Friedrich-Löffler-Institute (federal research institute for animal health). The insitute is interested in the migration of birds from Scandinavia (Figure 1.1) heading south via Germany. Once lakes in Scandinavia are freezing in the winter the local birds are moving southwards. Bird ringing data showed that the birds are migrating via Germany. Since these birds are suspected of spreading bird flu it is worthwhile to study their migration patterns. Scientists are interested in an easily accessible way to determine when the birds are starting to migrate. This should be used to inform farmers to take preventative measures and minimize risks for their animals. Optical satellite data has been used in the

FIGURE 1.1: The area of interest that should be investigated. The outline shown is provided by the Friedrich-Löffler-Institute

past to explore lakes and check if they are frozen. Unfortunately Scandinavia is affected by the polar night during winter and before the polar night starts it is often covered by clouds, therefore optical data is often not available, when radar is.
Due to its independence from weather conditions, RADAR observation is the most reliable way to generate data in Scandinavia in winter [1]. Starting in 2014 (Sentinel-1 A) and 2016 (Sentinel-1B) ESAs satellite pair of Sentinel-1 is providing radar data for the whole Earth. The high temporal resolution of maximum 6 days between two observations increases the usefulness for this study since the freezing of the lakes can happen on smaller timescale for most lakes [2, 3].The data is available for free and can be downloaded from the Copernicus-Webservice
(`https://scihub.copernicus.eu/`).
The goals of this work are (1) to create an algorithm to detect the date of freeze-up of lakes in Scandinavia in near-real time and (2) get an idea of the ice extent of the lakes.

1.2 State of the art

Freeze-up dates are highly influenced by thermal conditions, the air temperature above a lake is determining if a freeze-up is possible [4]. Bigger lakes are able to store more heat and thus will freeze later [2]. Lake area is also important since smaller lakes tend to freeze over when the bulk temperature is higher (2-3° C) whereas big lakes need to have a temperature of 1°C. Lake depth and volume play an important role as well for heat transfer and exchange [5]. Tracking the actual freeze-up is really difficult since lakes can freeze completely within hours once the bulk temperature of the lake is reaching 4° C [3]. Calm, clear and cold nights in Siberia resulted in freezing in late August which used to dissipate once the sun was risen [6]. Small bays often freeze first but are disturbed by waves and wind that can lead to cracks [3].
Different approaches using different radar satellites were developed in the past to detect lake ice. Radar antennas are designed to transmit electromagnetic waves of a chosen polarisation. Most radars are using horizontally (H) or vertically (V) polarised waves, some use circular polarisations [7]. When the wave gets scattered on the ground, the polarisation can change, which leads to differently polarised backscatter. This backscatter is again horizontally or vertically polarised. The naming convention for this systems is for example: HH is transmitted and received horizontally, VH is a wave that is transmitted vertically polarised and received horizontally. Different satellites are using different polarisations. Both polarisations of Sentinel-1 (VV and VH) are interacting differently with the surface of the lake if it is frozen or not, since the dielectric properties of water and ice differ [8]. Not only do ice and water have different effects on the backscattering, waves tend to influence the signal in both VV and VH. VH values are less sensitive to waves and thus wind [2, 9].
The reflection of the wave on ice is also influenced by other factors such as ice

type [10, 11], ice thickness [4, 5, 11], the crystallographic type of the ice [3, 10], the wetness of a potential snow cover of the ice [1, 6, 10, 12, 13], lake depth [4], the frequency of the incoming signal (which is at least more or less fixed for the same satellite), the incidence angle [14, 15] and whether the satellite is in ascending or descending orbit [10].
In higher latitudes dry snow is more common than wet snow [13]. Not only the thickness of a potential snow coverage is important. If the snow is wet, it interacts with the radar signal in a way that prevents it from reaching the ice/water interface and thus the reflection is not influenced by the lake surface but absorbed by the snow. Dry snow is penetrated by SAR and thus the returned waves will display the lake surface [12]. The two polarisations of Sentinel-1 are also interacting differently with dry and wet snow [16]: Dry snow is not really influencing VV values since the snow is penetrated and the interaction of the water/ice boundary with the wave is consistent. VH values are influenced by the accumulation of snow. An increased depolarization through scattering is leading to an rise of the intensity of VH [6, 10, 13, 17].
Investigations at *Lake El'gygytgyn* in Siberia showed that the underlying topography and related to this bioactivity is influencing the backscatter [6]. Higher bioactivity is leading to more bubbles of CO_2, O_2 and CH_4 [18] in the ice and thus higher reflectances [5, 6, 18, 19, 20]. Rough surface conditions of ice, like in cracks in the ice, increase the reflected magnitude since the backscatter is a combination of surface backscatter and volume backscatter. Thus the signal that is reaching the antenna has a higher magnitude [4, 17, 21]. Once the ice is freezing to the ground of a lake, a decrease in backscatter is observed. This is due to the absorbtion of the signal by the underlying ground [5, 6, 22, 23]. Data shows an increase in sigma naught of the HH values for RADARSAT-2 at the start of ice formation [10], VV values of the Quicksat satellite tend to increase at ice-on events as well [2]. But there are some complications occurring when using radar data (HH). There is a low difference between sigma naught values between open water and newly formed ice [10]. A known problem of fresh ice detection using radar is the very little interaction of newly formed ice with radar waves [21], which results in low backscatter in both VV and VH for thin, flat surfaced ice [9]. This is due to the limited scattering at the smooth water-ice boundary [21]. Sigma naught of co-polarized waves was found oscillating during ice formation and increasing once ice is melting due to the roughing of the surface [2, 10]. After "evolving" to sheet ice the ice has a smooth underside but with variable degrees of dielectrical discontinuities. Since the waves are back-scattered by surface scattering, volume scattering or a combination of both the backscatter levels are varying a lot [21].
Previous studies have shown the influence of meteorological data on the backscatter, even though Sentinel-1 is supposedly be weather independent. Higher wind speed shows higher backscatter [4, 10] and the SAR look direction relative to the wind direction is also influencing the backscatter. This was shown for RADARSAT-2 data [4], but could also have influence on Sentinel-1 data. The air temperature is also showing a strong negative correlation to the backscatter/sigma naught values for RADARSAT-2 data [10].

Threshold values for the VV-band of ERS 1 data [5] were used to determine the freeze-up of lakes. But they are adjusted for every scene due to different orbits and incidence angles [15, 24]. The ice conditions have minor effects on the threshold [24]. Another used approach is a threshold defined by the mean sigma naught of the first 90 days of a year plus a value X for HH values of RADARSAT-2 data [10]. Wet snow or liquid water above a potential ice cover affect the usability of the thresholding. Floating ice looks like ground-fast ice, if it is covered by water [17].
Different scientists have different opinions on the viability of cross-polarized data for freezing-periods. Some say it should not be used [25], others use it to distinguish between ice and open waters due to lower sensitivity to wind effects [2] but the best approach should be the usage of VV of Sentinel-1 [4].
An unsupervised classification algorithm (K-means classifier) was used on radar data for lakes that were known to be partly frozen. Three classes were differentiated due to their backscatter properties in co- and cross-polarized channels [24]. The three classes that were assigned are water, ice and wavy water due to the higher sensitivity of the co-polarized data to wind that is not present in cross-polarized data [11].
The basic idea of this approach is to get three classes (see Table 1.1) that have the following properties [25]:

Band	Water	Wavy water	Ice
VV	Low	High	High
VH	Low	Low	High

TABLE 1.1: Relative VV and VH values for different surface states

Once the scenes are classified the two classes having comparable low VH-backscatter values are combined for a water class [25]. Using additional filtering can help to decrease the misclassification due to melting pods but would also decrease the sharpness on the transition from ice/water [24]. This approach showed, unfortunately, that it is not that good for different ice states, snow coverage and problematic output once less than 3 classes are present, e.g. a full snow cover on a lake [25].
Using the SLC data of Sentinel-1 offers the possibility to use SAR interferometry if the surface is nearly uniform and thus not much scattering due to cracks is present [22]. Since the data has to be taken from the same orbit to perform interferometry the temporal resolution is below the GRD-using approaches and the amount of data that has to be used is higher. (One GRD-scene 1 GB, one SLC-scene 4 GB) The coherence of a pixel over two time steps with the same orbit was used as a threshold and if the coherence was below 0.3 the pixels got assigned “ice”. If wavy water is present the coherence is nearly the same as for sheet ice or deformed ice, but a distinction between sheet ice and water is possible using additional wind data [21]. The incidence angle of the incoming wave is irrelevant since the orbits are the

same for each coherence-estimation [21]. Once the lakes are ice covered in both scenes the coherence rises, but there are environmental changes that influence the coherence. Snowfall [26, 27], changing water levels, cracks in the ice and a changing ice dynamic have shown to influence the coherence for Sentinel-1 data [26].

2 Data and Methods

2.1 Sentinel-1

Sentinel-1 is scanning the surface of the earth using a single, right-looking C-band synthetic aperture radar (SAR) instrument (5.405 GHz centre frequency). The SAR instrument can work in dual-polarisation (HH + HV and VV + VH) but is only using single polarisations most of the time, depending on the acquisition mode [28].
The raw data is processed by the Instrument Processing Facility (IPF) to produce level 1 products which are most common. The processing consists of pre-processing, doppler centroid estimation and single-look complex focusing [29]. All products provide a calibration vector and information about the time of the acquisition (center of the image).
Sentinel-1 has 4 different acquisition modes [30]: (1) Stripmap (SM), (2) Interferometric Wide swath (IW), (3) Extra-Wide swath (EW) and (4) Wave mode (WV).

- Stripmap mode
  The stripmap mode is illuminating an 80 km wide swath by a continous sequence of pulses. The results of the stripmap mode have a spatial resolution of approximately 5m x 5m (single look) and contain dual (HH + HV and VV + VH) or single polarisations (VV + VH). It is only activated for imaging small island or in emergency cases [31].

- Interferometric wide swath
  The interferometric wide swath mode is the main acquisition mode used over land producing swaths with 250 km width, the spatial resolution of single looked scenes is approximately 5m x 20m. Images are captured in three sub-swaths using TOPSAR [32]. Using this technique results in constant image quality for all swaths but the azimuth resolution is reduced in comparison to stripmap, since the illumination time is decreased [33].

- Extra-Wide swath
  The Extra-Wide swath mode is used mainly to cover polar regions and sea ice. The EW mode is using the TOPSAR technique to produce 5 sub-swaths covering a total width of 400km at 20m x 40m resolution [34].

- Wave mode
  In Wave mode every 100 km a vignette of 20km x 20km at 5m by 5m spatial resolution is captured. Two consecutive scenes are taken at near-range (23°) and far-range (36°) containing either VV oder HH polarisation data. Over open ocean the main acquisition mode is the wave mode in VV polarisation [35].

Since it is the most used acquisition mode and covers the lakes in Scandinavia, the IW mode is used in this work. There are two different level 1 products for the interferometric wide swath mode: Single-look complex (SLC) and Ground range detected (GRD).

2.1.1 SLC products

Even though SLC data is captured in all acquisition modes, parts of the following information is only true for the used IW SLC products.
Sentinel-1 Single-look complex data can be downloaded as images which have been preprocessed by the ESA. Range processing and azimuth preprocessing, processing and post-processing are performed [29].
The big advantage of SLC data in comparison to GRD data is that the phase data is stored in an imaginary band, thus having both amplitude and phase data for both VV and VH waves is increasing the amount of data. A downside is that more disk space is required (4 to 5 GB per scene in comparison to 1 GB per scene for GRD).
The products contain one image per polarisation channel and sub-swath resulting in 3 to 6 images per scene. Each sub-swath contains the data of different bursts, that were processed as single SLC images beforehand and are restricted in the sub-swaths by black demarcation zones.
IW SLC images are resampled to a single pixel-spacing in both azimuth and range thus no resampling or interpolating has to be done in later stages of processing [36].

2.1.2 GRD products

GRD products are processed further in comparison to SLC data: focusing, detection, multi-looking and projection to an ellipsoid model of the earth have been done. Due to the multi-looking the speckle in the product is reduced. The pixel values of the approximate square pixels contain the magnitude information whereas the phase information present in the SLC data is lost.
Due to the presence of a noise vector annotation data set that is provided with the images the thermal noise can be removed. All bursts within a scene are merged together to create a single GRD product for each polarisation channel [29].
GRD data is produced in different resolutions: full resolution, high resolution and medium resolution. For the IW mode only high and medium resolution are available. High resolution imagery has a spatial resolution of slightly

above 20 m x 20 m and a pixel spacing of 10 m by 10 m [37].
Sentinel-1 is a dual-polarization system that is capturing VV (co-polarized) and VH (cross-polarized) information in IW mode.

2.2 Processing environments

2.2.1 SNAP

The Sentinel Application Platform (SNAP) is an architecture developed by Brockmann Consult combining the different Sentinel Toolboxes. It is designed to process and display big data (Giga-pixel images) from satellites and offers a lot of different functions to process and analyse satellite data [38].

2.2.2 Google Earth Engine

The Google Earth Engine (GEE) is both an archive for an incredibly big amount of data and gives the possibility to apply algorithms to the data to create fast results due to the big computational power [39]. GEE is not able to process complex data yet, thus it is only usable for GRD data analysis and calculations.

2.2.3 ISCE

The Interferometric synthetic aperture radar Scientific Computing Environment (ISCE) is an open source software which was founded by the NASA. It is currently able to process data gathered by eleven different platforms, including Sentinel-1. [40]
ISCE can be installed using Anaconda under Linux. It is possible to use a Windows Subsystem for Linux (WSL) to install ISCE. The WSL allows only command-line processing but this is sufficient for my purpose, as the results can be opened in any GIS.

2.3 Ground Truth

Different approaches were used to get a ground truth of the observed lakes to determine both where and at what point in time freezing took place on the lakes. The first approach was to use the Sentinel-Playground with Sentinel-2 data because of the high temporal resolution and the free availability [41]. Different lakes were selected based on the availability of images in the freezing period where freeze-on of the lakes was indicated.
Obviously for Scandinavian winters the amount of images where lakes are free of clouds are few, but even if the lakes are visible it is not 100% accurate because ice is not that easy to see on satellite images. Cracks or other ice features increase the visibility of ice. Figure 2.1 shows a comparison of two images from the "Müggelsee" where this problem is shown.
Both mentioned problems lead to imprecision of the exact date of freezing

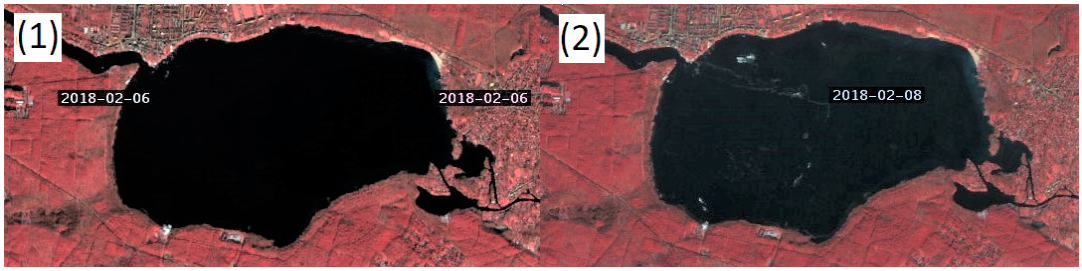

FIGURE 2.1: Comparison of two Sentinel-2 False-Color-Images (B8, B4, B3) the "Müggelsee" near Berlin in February 2018. The images were edited the same way to increase the visibility of the cracks. (1) On 6th of February the lake is covered by 30-50% ice. (2) On 8th of February the lake has an ice cover covering 50% of the lakes, but with cracks in it that make the ice easily recognizable. (Image source: https://services.sentinel-hub.com, Ice-Data: IGB, see Appendix C)

but made it possible to get a rough time span of when the lake froze.
The freezing period was also compared to temperature data that was found on the website `www.timeanddate.com` to get an estimate of when the lake could have frozen in this period. In case the air temperature was significantly higher than the freezing point of water the time span could be narrowed further. Since there is no information about the source of the temperature data, there is on guarantee that the data is correct. [42]
Ground truth data was also collected by the Leibnitz-institute for water ecology and freshwater fishery in Berlin. The institute collected data near-weekly during the freezing period of "Müggelsee" in February-April 2018 and they provided near-weekly ice coverage data of "Großer Stechlinsee" and "Nehmitzsee" from January 2009 onward. Unfortunately there was no comment where the e.g. 50% of ice on the lakes occurred. The data can be found in Appendix C. Thus I used the data in combination with satellite data if possible to get an eligible estimation of the ice extent. Figure 2.2 is showing the location of the different lakes that were used to test or validate results. Lakes in Germany, Finland, Sweden and Norway were investigated.

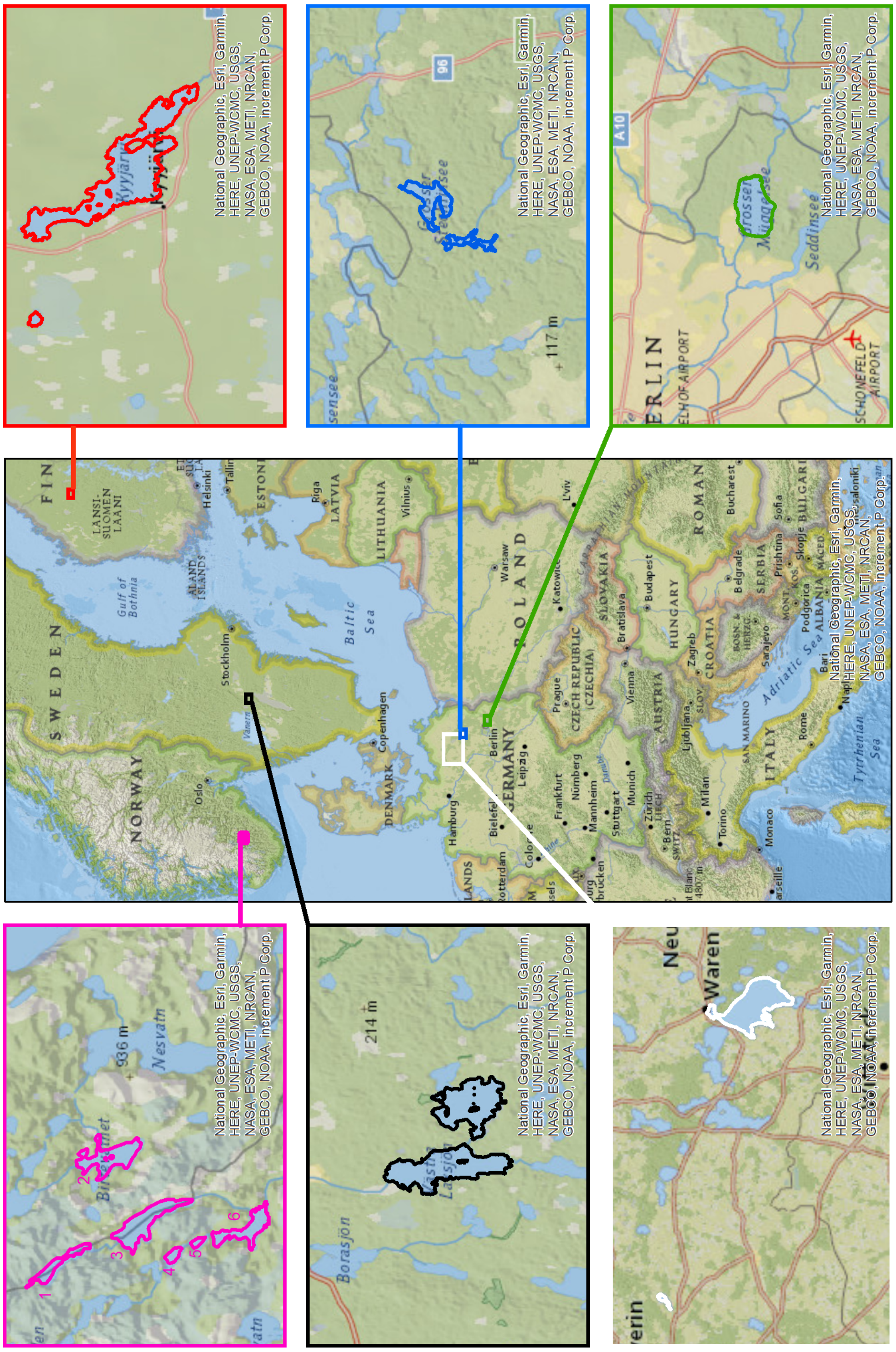

FIGURE 2.2: The lakes that were investigated. The pink part is located in Norway and showing the lakes (1)"Grossæ" (2)"Birtevatn" (3)"Grøssæ" (4)"Varosen" (5)Kjetebuvatn" and "(6)Topsæ". Black is showing "Västra Laxsjön" (West) and "Östra Laxsjön" (East). The white outlines are showing "Müritz" (East) and "Barniner See" (West). In green you can see "Müggelsee" close to Berlin. The blue color is showing "Nehmitzsee" and "Stechlinsee" and in red "Lake Kyyjärvi" and "Peuralampi" in Finland are shown.

2.4 GRD products

Preprocessing in SNAP

Sentinel-1 Ground Range Detected (GRD) data is already preprocessed to a certain degree by the ESA, but needs further processing to make different products comparable. One approach for a standard workflow for the correction of the data gained by Sentinel-1 is shown in figure 2.3 and used here [17, 43].

FIGURE 2.3: Standard workflow for processing of Sentinel-1 GRD data.

Apply-Orbit-File
With each Sentinel-1 scene orbit data is provided that is calculated before by using the theoretical orbit of the satellite. In reality this data is not perfect and therefore the European Space Agency (ESA) is providing precise orbit files days to weeks after the image has been taken. Using this data is improving the quality of the product. Unfortunately this data is provided delayed and is not usable in this project since we need a near real-time result and can not wait for the precise orbit files.
SNAP provides automatic downloads for the precise orbit files which I used but it is necessary to check the box "Do not fail if new orbit file is not found" in case there is no new orbit file to be downloaded.
Without a new orbit file applying an orbit file might be unnecessary, but should be mentioned as a processing step since it is part of a standard processing chain for Sentinel-1 GRD products [43, 44].

Thermal Noise Removal
Sentinel-1 data can be disturbed by thermal noise that has to be removed. Especially the intensity of the cross-polarized channel is susceptible for the impact of thermal noise [45]. The operator is reducing the discontinuities between sub-swaths by normalisation of the whole Sentinel-1 scene [43].
Each Level-1 product comes with a noise look-up table (LUT) for all measurements which can be used to produce noise profiles. Pixels that are between different points in the LUT are calculated using bilinear interpolation.
The thermal noise removal operator can be used for both the removal and the reintroduction of thermal noise to Sentinel-1 scenes [44], but was only used to remove the initial noise in my work.

Remove GRD Border Noise

The preprocessing of the raw data of Sentinel-1 products is – as stated earlier – done by different IPFs. The IPF processing chain leads to artefacts at the image borders which should be removed to get accurate results. Removing these no-value pixels is done by using a threshold for every product processed by an IPF version 2.9 or higher [46]. All images I downloaded from 2018 and later were processed by an IPF higher than 2.9 so the thresholding method is used here.

Calibration

SAR calibration is performed to provide pixel values that are directly related to the backscatter of the scene. This is necessary for quantitative use of SAR data, whereas uncalibrated data can be sufficient for qualitative use [44]. Production of level 1 images does not apply radiometric corrections which leads to radiometric bias in the products which has to be removed. Radiometric correction is necessary to produce comparable data of different times, different sensors or same sensors in different modes [43].
Sentinel-1 data can be calibrated using information that can be found in the product: a calibration vector is provided as a part of the product and the simple conversion can be done. Look-up tables apply range-dependant variables adding to the calibration constants, there are four different LUT that can be used for different results, namely $\beta^0, \gamma^0, \sigma^0$ or digital numbers (DN) [44]. I used σ^0 in my work.

Speckle Filter

The quality of SAR images is lowered by speckle occurring on the images that lead to increased difficulties in interpretation of the results. The speckle is produced by interferences of waves reflected by elementary scatterers [47]. Snap offers a variety of speckle filtering methods (Boxcar, median, Frost, Lee, Refined Lee, Gamma-MAP, Lee Sigma, IDAN) in the speckle filter operator [44].
I am using the Refined Lee filter (Refined Local Statistics Filter) which is applying a local statistics filter on edge directed windows [47]. The filter is accepted as the best filter for single images, since it preserves edges and texture information [43].

Terrain-Correction

SAR data is produced with different incidence angles thus it is necessary to correct for the distorted distances in the images. Terrain correction is used to adjust the image to get rid of the distortion and make an as good as possible representation of the real world [43].
Applying terrain correction is adjusting the image by using both the orbit files and a digital elevation model. In my case I used the ASTER 1sec GDEM of the NASA Jet Propulsion Laboratory, because the SRTM, which would be preferred due to better spatial resolution, is not covering the whole area of interest [44].

The used algorithm for terrain correction is the Range-Doppler terrain correction.
I set the output pixel spacing to 10 m and it is important to uncheck the "Mask out areas without elevation" box, since lakes have no elevation in some DEMs which would lead to errors in the result. A possible problem in the terrain correction are hilly regions where micro-topography is complicating the terrain correction leading to geolocation errors. [48]. .

Preprocessing in the GEE

The preprocessing of the GRD data is done by Google before the data is usable in the GEE. The processing chain is comparable to the one I performed using SNAP. The following steps are performed: Apply orbit file, GRD border noise removal, thermal noise removal, radiometric calibration and terrain correction [49]. The Lee speckle filter is applied as well by using a code-snippet written by Guido Lemoine (see Appendix A).

2.4.1 Time series

To survey if there is a threshold value or ratio that changes when a lake is freezing I created time series for different combinations of bands. I investigated the VV-values and VH-values themselves, the ratio $\frac{VV}{VH}$, a normalized index $\frac{VV-VH}{VV+VH}$ and the product VV*VH. Time series were created for single points in the lakes and a lake-wide mean of the ratio was calculated.
I tested the approaches on different lakes both in the Google Earth Engine and in SNAP. For the lake-wide mean different results were observed for both processing parts.

2.4.2 Thresholds

To check if there are thresholds in any of the specified values/bands I used the GEE to get values for frozen/not frozen pixels in lakes and define a threshold by hand (see figure 2.4). This was previously done to differentiate between ground-fast ice and floating ice, but could also be useful to determine between water and ice [17].

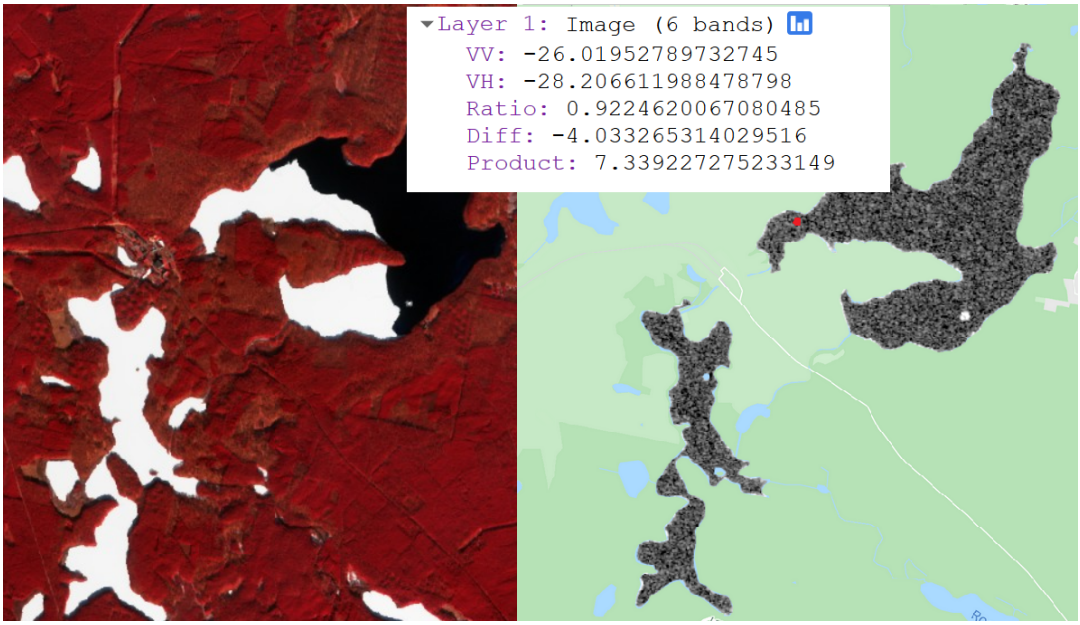

FIGURE 2.4: Example for defining the threshold: (1) False color composite (B8-B4-B3) of the "Großer Stechlinsee" and "Nehmitzsee" in Brandenburg (source: https://services.sentinel-hub.com). (2) Screenshot of the GEE interface. The red dot is showing the location that was picked, in this case it is obviously a frozen part of the lake. The small box shows the values for different bands (the 6th band is the removed angle band).

This was done for several lakes, trying to transfer the found thresholds to other lakes or get a good fit for more than one lake.

2.4.3 Unsupervised classification

According to the initial idea of different responses of the three different types of lake surface, namely calm water, wavy water and ice, different unsupervised classification algorithms were used in the GEE. I tried to run it in SNAP on my local computer but it took way too much time to be usable. Classification was done on the lakes only, so the land pixels will not be taken into consideration and create their own classes.
The GEE is offering different methods for the unsupervised classification that were tested. My results showed that the best of the different algorithms that is usable in the GEE was the K-Means-Classification based on the weka algorithms [50]. This clustering algorithm is one of the most used techniques in related research, which is the reason I focused on it. Additionally, the comparatively fast calculations and the good fit for large scale data strengthened my decision [24]. Due to the usage of relative comparison the absolute values of VV and VH are not important, which reduces the influence of the orbit, incidence angle and lake size [24, 25].
The clustering is based on creating K groups inside the data set, each consisting of one single point to start with. All other points get assigned due to the smallest distance of the point to the center of any group. The center of the group is the mean, thus the term K-Means. After each added point the mean of the group is recalculated. Different iterations through all points make sure that no point is only assigned because it was calculated early[51].
This algorithm is further improved by using the kmeans++ method, which is using a randomized seeding of the initial centroids. This is expected to lead to better results in the end [50].

2.5 SLC products

2.5.1 Interferometry

Interferometric synthetic aperture radar (InSAR) uses the phase difference between two images with the same orbit over the same area at different times [22]. SLC data consists of amplitude and phase bands, whereas the amplitude is an indicator for the strength of the signal while the phase is an indirect indicator for the distance between the satellite and the earth's surface [52]. The phase is a value between 0 and 2 Pi since it is a fraction of a full wave cycle [53]. The phase difference is mainly based on differences in topography, thus InSAR can be used to create high resolution digital elevation models.
An important step in processing interferograms is the perfect alignment of the scenes to ensure the calculations are done correctly [9]. Interferogram formation is done by cross multiplying the master image with the complex conjugate of the slave image, which leads to an output with amplitude and phase information. Amplitude data is the result of the multiplication of the two initial scenes, while the phase band contains the phase difference between both scenes [53]. Another outcome of processing interferograms is the coherence which is an indicator for the level of similarity between two images ranging from 0 to 1. If the coherence is high there is little change between the two images, if there was a lot of change, e.g. during vegetation growth, the coherence should be small.
There are five main influences on the coherence or decorrelation in general [54]: (1) SNR of the satellite system, (2) processing of the raw data to SLC data. (3) coregistration errors, (4) the across-track distance and (5) changing surface conditions. The signal-to-noise ratio is usually high for dry snow covering ground ice, thus the effect of noise should be neglectable [27]. Decorrelation due to processing of the raw data should be minimized during the processing of the data by the IPFs [26]. The coregistration is really good since Enhanced Spectral Diversity (ESD, [44]) got used to improve the results. The across-track distance errors and the temporal change in the surface both have influences on the coherence [26, 54, 55], but the decorrelation due to the perpendicular baselines should be small for Sentinel-1 [26]. Thus a changing coherence should represent a temporal change in the data used here.
Because of the different dielectric properties of ice and water [8] and thus their different responses to incoming waves I decided to calculate interferograms for some lakes and scenes to see if there is a usable change in the results when the lakes froze. Not only should the interferograms show differences, but also the coherence should change during the process. Since the GEE is not yet able to compute complex data the processing of the interferograms was mainly done in SNAP and using the InSAR Scientific Computational Environment (ISCE) for further investigation.

SNAP

The processing was done according to a tutorial published by the ESA (see figure 2.5)[53].
It is necessary that the images are taken in the same orbit since phase differences will not be indicating changes in topography when different orbits are used.

Coregistering the Data

The first step is coregistering the data to make sure that the scenes overlap on a sub-pixel size level. It consists of different operators: Read, TOPSAR-Split, Apply-Orbit-File, Back-Geocoding, Write.
The two Read-Operators are reading the images and define a master and a slave image. TOPSAR-Split is splitting the scenes in selected subswath, bursts and polarisation(s). Polarisations, bursts and subswath have to be the same for both images.
The Apply-Orbit-File-Operator is doing what is already described in the GRD preprocessing. Back-Geocoding is performing an interpolation of the slave image to fit to the master using the adjusted orbits and the DEM [44]. After the coregistration is done, the workflow shown in figure 2.5 can be used.

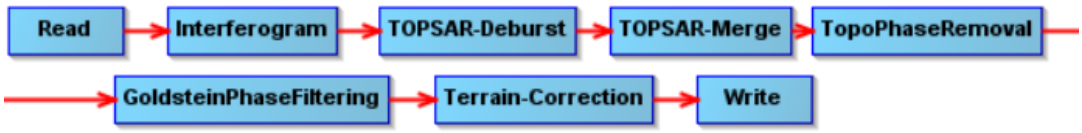

FIGURE 2.5: Standard workflow for processing SLC data in SNAP

Interferogram Formation and Coherence Estimation

This operator is calculating the interferogram. The user can choose to subtract the flat-earth phase (reference) which would be done by calculating a 2D-polynomial. If the orbit and metadata is known the reference is calculated using the influence of the known topography on the phase information [44]. In the dialogue the user can check a box "Include coherence estimation". This will produce an additional output band containing the coherence. If the user chooses to subtract the flat-earth phase before it will be done using the same polynomial as before.

TOPS Deburst and TOPS Merge

Sentinel-1 scenes consist of different bursts within the different swaths. The different bursts are separated in the initial scene. The TOPS-Deburst operator is combining the different bursts in range direction and resulting in a complete image, without the black stripes between each swath. The different swaths are merged together to one full image for the whole scene by the

TOPS-Merge operator which is also using the range overlap between the different swaths [44, 53].

Topographic Phase Removal
To remove the influence of topography on the phase data stored in the bands it should be removed using the topographic phase removal operator. This operator will simulate an interferogram based on a chosen/auto-downloaded interferogram and subtract it from the calculated one. This will result in a flattened interferogram [56].

Phase Filtering
The last step in producing interferograms is filtering the results. The phase information can be distorted by temporal or geometric decorrelation, volume scattering or by processing errors. To decrease the influence of this noise on the phase the signal-to-noise ratio should be increased by using the Goldstein Phase Filter [57]. This filter is mainly necessary if the phase should be unwrapped later since it is decreasing the chance of having unwrapping errors due to noisy regions.

Geocoding
As a last step the result should be geocoded using Range-Doppler-Terrain-Correction (see GRD-Data) to make it usable on maps and ease further interpretation of the results [56, 58].

ISCE

The script to calculate the interferogram for TOPSAR-imagery is called topsApp.py. This script has to be executed in the directory where the topsinsar.xml (see Source Code 1) file is stored and – in my case – the image because the path is not defined further.

```
<topsinsar>
  <component name="topsinsar">
    <component name="master">
      <property name="output directory">master</property>
      <property name="safe">S1B_IW_SLC__1SDV_20180310T165118_
      ↪ 20180310T165145_009972_01212E_0F81.SAFE</property>
          <property name="polarization">vh</property>
    </component>
    <component name="slave">
      <property name="output directory">slave</property>
      <property name="safe">S1A_IW_SLC__1SDV_20180304T165159_
      ↪ 20180304T165226_020868_023CAC_C462.SAFE</property>
          <property name="polarization">vh</property>
          </component>
          <property name="swaths">[2]</property>
          <property name="usegpu">True</property>
          <property name="do Unwrap">True</property>
          <property name="do ESD">True</property>
          <property name="azimuthlooks">3</property>
          <property name="rangelooks">7</property>
          <property name="region of interest">[52.41 52.46 13.5
           ↪ 13.8]</property>
  </component>
</topsinsar>
```

SOURCE CODE 1: Example source-code for interferogram calculation for the VH bands of swath 2 of two consecutive Sentinel-1 scenes.

ISCE is running different steps throughout the process. Each step can be done separately and the result of each step is saved in a PICKLE directory. If the processing is stopped once, it can be restarted after the last completed step [40]. The steps that are performed are shown and described below. Everything after the Topo-step and before the RangeCoreg-Step is performed only for the overlapping areas of the bursts [59].

Startup	The PICKLE directory is created and the .xml-file is checked for typos or wrong property names.
Preprocess	Orbits, bursts, antenna patterns and the IPF are extracted.

ComputeBaseline	Parallel and perpendicular baselines of the scenes are calculated using the orbit information.
verifyDEM	If a DEM is set in the .xml-file the DEM will be checked, if no DEM is set, it will download the DEM for the specified area (if the earthdata settings are valid, there has to be a .netrc-file in the working directory that contains the used information for the download hub and the username/password combination).
Topo	The DEM is mapped into radar coordinates of the master image.
SubsetOverlaps	For the master geometry: top and bottom burst overlap are computed.
CoarseOffset	Offsets between bursts are extracted using the orbit files, since they are not very accurate they are called coarse offsets.
CoarseResamp	The slave image is resampled using the coarse offset information.
OverlapIfg	Interferograms of the overlap regions are calculated.
PrepESD	Double difference interferograms are calculated for each pair of bursts. Double difference interferograms capture displacement in azimuth direction.
ESD	The azimuth offset in pixel size is estimated using all pixels that have a coherence bigger than 0.85 in both bursts.
RangeCoreg	Range offsets are estimated by cross-correlating the amplitude of the master and slave burst overlaps.
FineOffset	Both the azimuth and range offsets are applied to the full bursts.
FineResamp	The fine offsets are used to resample the slave image to the masters geometry.
BurstIfg	Interferograms are calculated for each burst since master and slave are in the same geometry now.
MergeBursts	The different interferograms are merged together
Filter	Additional filtering is performed based on a threshold defined in the .xml-file or if the value is not set using a standard value
Unwrap	Phase unwrapping is performed on the results.

Unwrap2stage — A two-stage unwrapper: the unwrapped interferograms get adjusted using the connected component file that gets created while unwrapping.

Geocode — Geocoding the results in the property "geocode list" in the .xml-file. If no list is specified, the results *filt_topophase.flat*, *filt_topophase.unw*, *los.rdr*, *phsig.cor*, *topophase.cor* and *topophase.flat* will be georeferenced.

The results were used to see if there is a significant change in the interferograms or the coherence when ice forms on the lakes for the first time. The results that were used are called: *topophase.cor.geo*, a indication of the topographic change and *phsig.cor.geo*, the coherence between the two scenes. The *.geo*-ending is indicating that the result is georeferenced, which eases the usage in any GIS.

2.5.2 X-Polarised interferograms

Due to the different interaction of ice with co-polarized and cross-polarized waves an approach introduced by Dr. Bodo Bookhagenwas to calculate X-Polarised interferograms. Those are interferograms between the co- and the cross-polarized backscatter of the same image. Those interferograms should change when ice forms on the lake since the different wave types get scattered differently.
Only the ISCE was used to calculate these interferograms since SNAP is not able to co-register differently polarized waves. The setup is the same as shown in Source Code 1, but the master and the slave images are the same. Additionally, the polarisations of the two scenes should differ. ESD must not be performed since there is no overlap error and it will return an error otherwise.

3 Results

3.1 GRD products

3.1.1 Time series

The code to produce the shown time series with the GEE are stored in Appendix A. Figures 3.1 to 3.5 show the different bands for "Nehmitzsee" in winter 2017/2018.

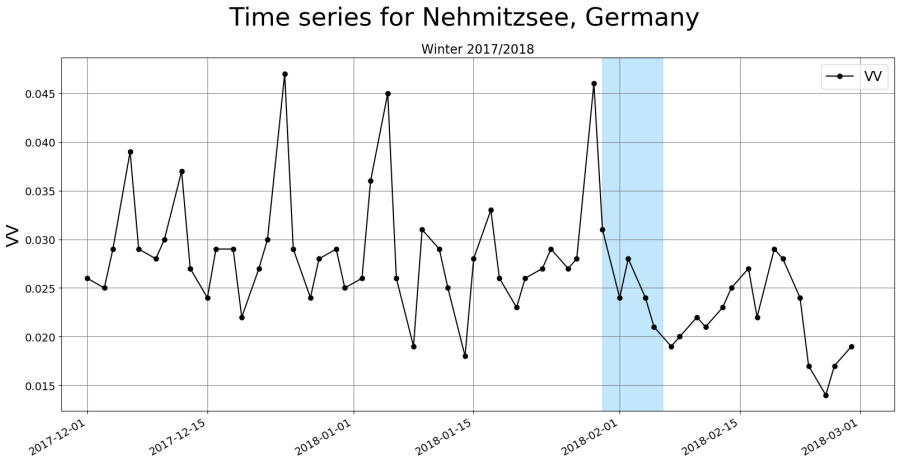

FIGURE 3.1: Time series showing the lake-wide mean value for the VV band during winter 2017/2018. The blue box is the freezing period according to the IGB data (see Appendix C). The lake was shifting from being ice-free to being completely frozen between 30th January 2018 and 6th February 2018

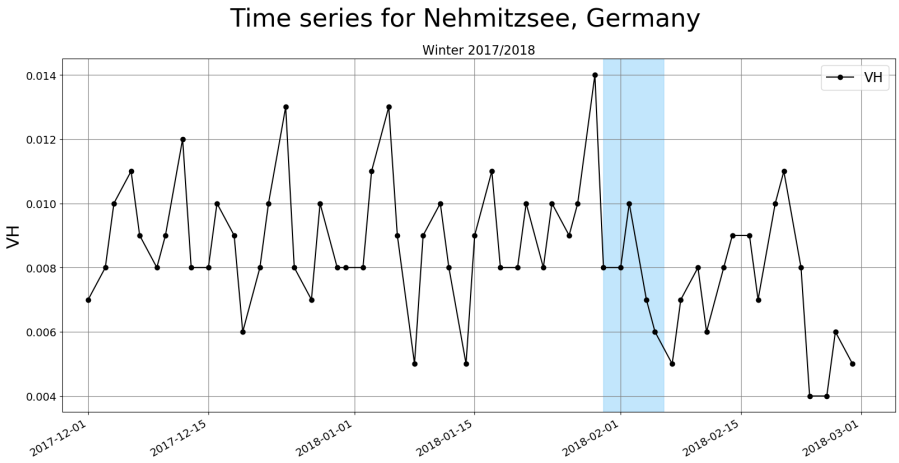

FIGURE 3.2: Time series showing the lake-wide mean value for the VH band during winter 2017/2018. The blue box is the freezing period according to the IGB data(see Appendix C). The lake was shifting from being ice-free to being completely frozen between 30th January 2018 and 6th February 2018

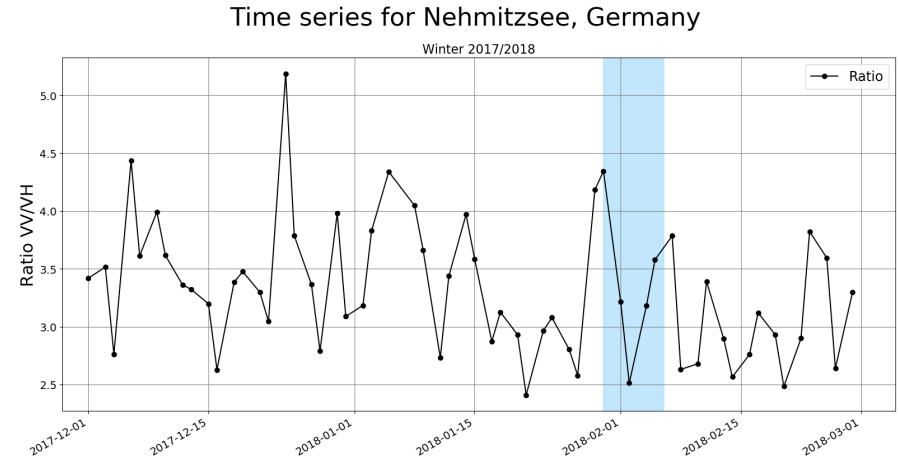

FIGURE 3.3: Time series showing the lake-wide mean value for the ratio of the two bands during winter 2017/2018. The blue box is the freezing period according to the IGB data(see Appendix C). The lake was shifting from being ice-free to being completely frozen between 30th January 2018 and 6th February 2018

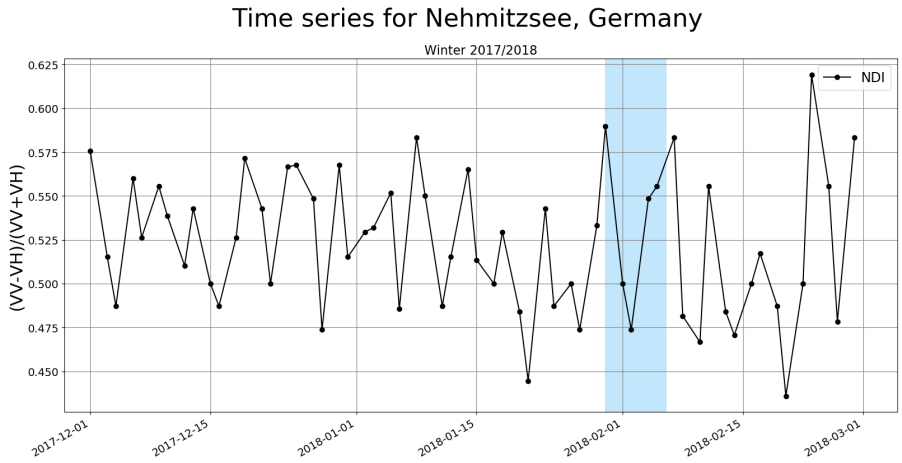

FIGURE 3.4: Time series showing the lake-wide mean value for the normalised difference index of the two bands during winter 2017/2018. The blue box is the freezing period according to the IGB data (see Appendix C). The lake was shifting from being ice-free to being completely frozen between 30th January 2018 and 6th February 2018

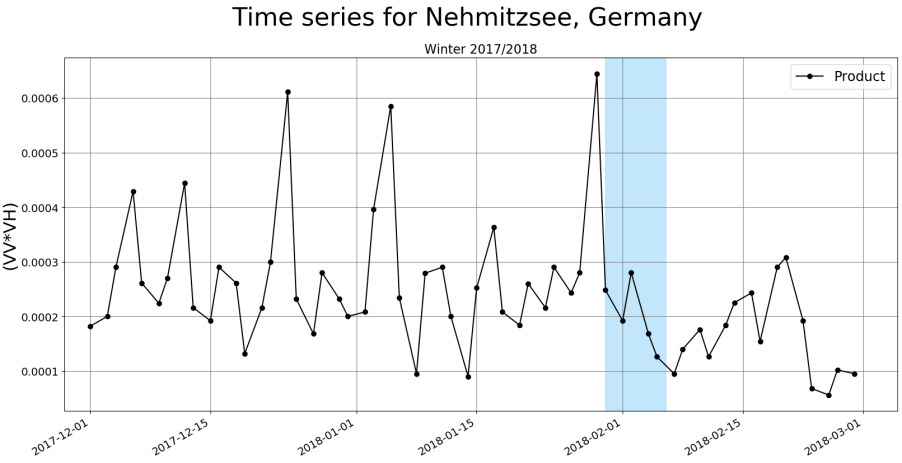

FIGURE 3.5: Time series showing the lake-wide mean value for the product of the two bands during winter 2017/2018. The blue box is the freezing period according to the IGB data (see Appendix C). The lake was shifting from being ice-free to being completely frozen between 30th January 2018 and 6th February 2018

The graphs show that there is no real use of the *NDI* values since they change a lot during the whole time series and show no significant or reliable behaviour during the freezing period.
The graph of the product seems to be better, but the peak before the freezing

started and the non-existence of significant behaviour throughout the freezing period lead to rejection of the product as an indicator of freezing.
The charts for the single bands and their ratio appear to be better suited and were therefore investigated further. A problem with observing the changes of VV or VH values over time is the difference in the incidence angle, since the angle has a high influence on the backscatter. Higher angles tend to have lower backscatter values for both VH and VV values [10].
Thus I decided to investigate mainly the **ratio** of the two bands under the assumption of a comparable influence of the incidence angle on both VV and VH backscatter [16]. Using the ratio could also lead to a decrease of the influence on values that is caused by changing snow conditions if both values are influenced the same way.
Note that the data shown before is extracted of the GEE, the following figures are produced using SNAP preprocessing on my local PC.
Since there was no big discrepancy between looking at small areas (30m x 30m, 50m x 50m) or the mean over whole lakes, we decided to be computing time efficient and use the mean values of the whole lake polygon buffered with -10 m to get rid of potential mixed pixels from lake shores.

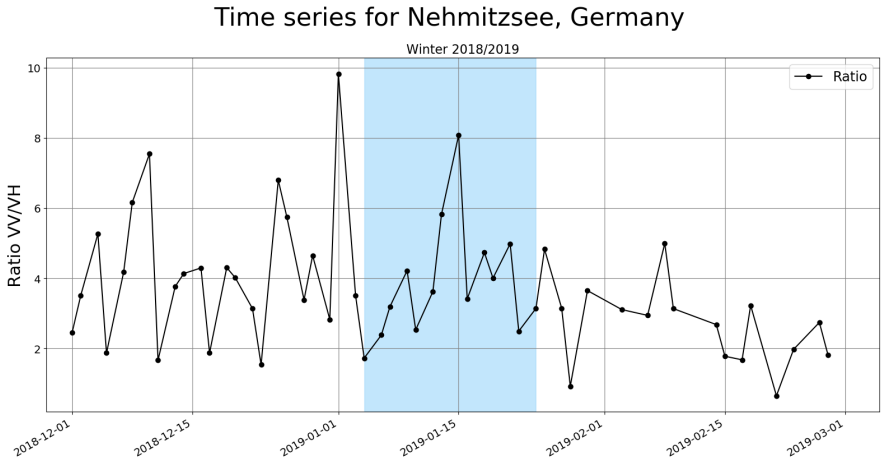

FIGURE 3.6: Time series showing the lake-wide mean value for the ratio during winter 2018/2019 for the Nehmitzsee. The blue box is the freezing period according to the IGB data (see Appendix C). The lake froze between 4th January 2019 and 30th January 2019

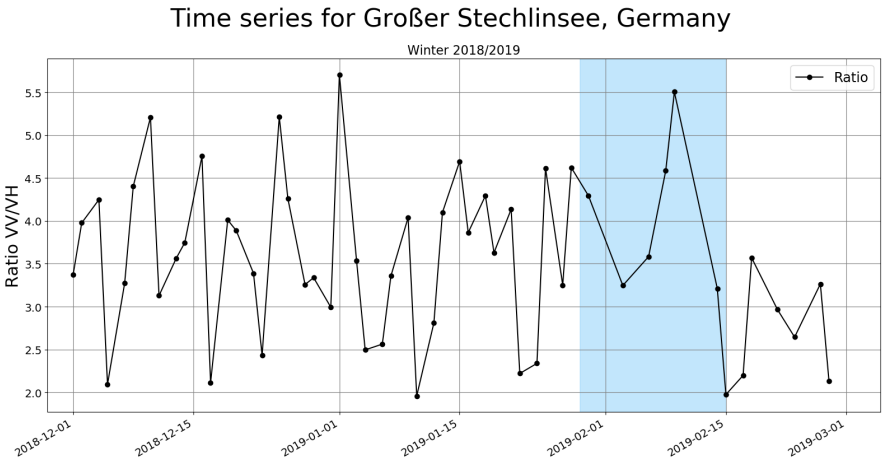

FIGURE 3.7: Time series showing the lake-wide mean value for the ratio during winter 2018/2019 for the "Großer Stechlinsee". The blue box is the freezing period according to the IGB data (see Appendix C). The lake was freezing between 29th January 2019 (1% surface ice) and 15th February 2019 (50 % surface ice)

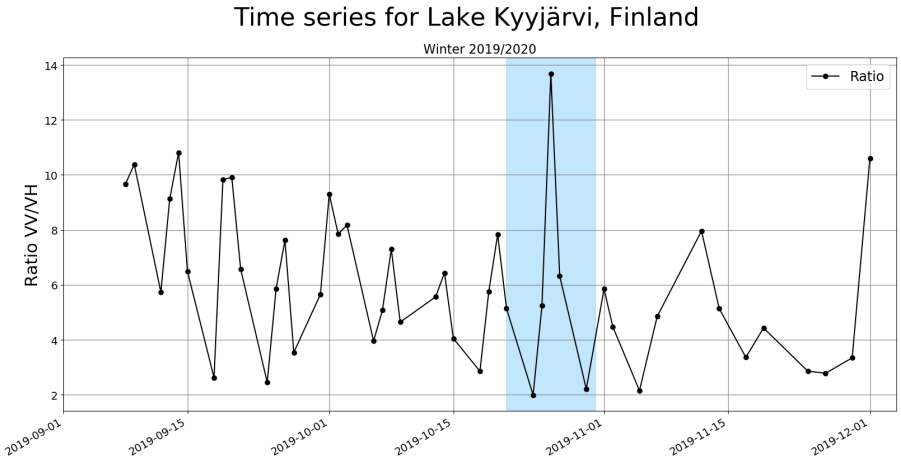

FIGURE 3.8: Time series showing the lake-wide mean value for the ratio during winter 2019/2020 for "Lake Kyyjärvi". The blue box is the freezing period according to satellite data (Sentinel-2), the lake was freezing between 10th October 2019 and 30th October 2019

Looking at figures 3.6 to 3.8 a change is occurring during the freezing period. The three figures show a significant peak during the freezing period. "Lake Kyyjärvi" is not showing behaviour like this before the freezing period.

3.1.2 Tresholds

Defining thresholds for the different bands worked fine for single images but it was not transferable to other dates or other lakes on the same date. To see the used code see Appendix A.

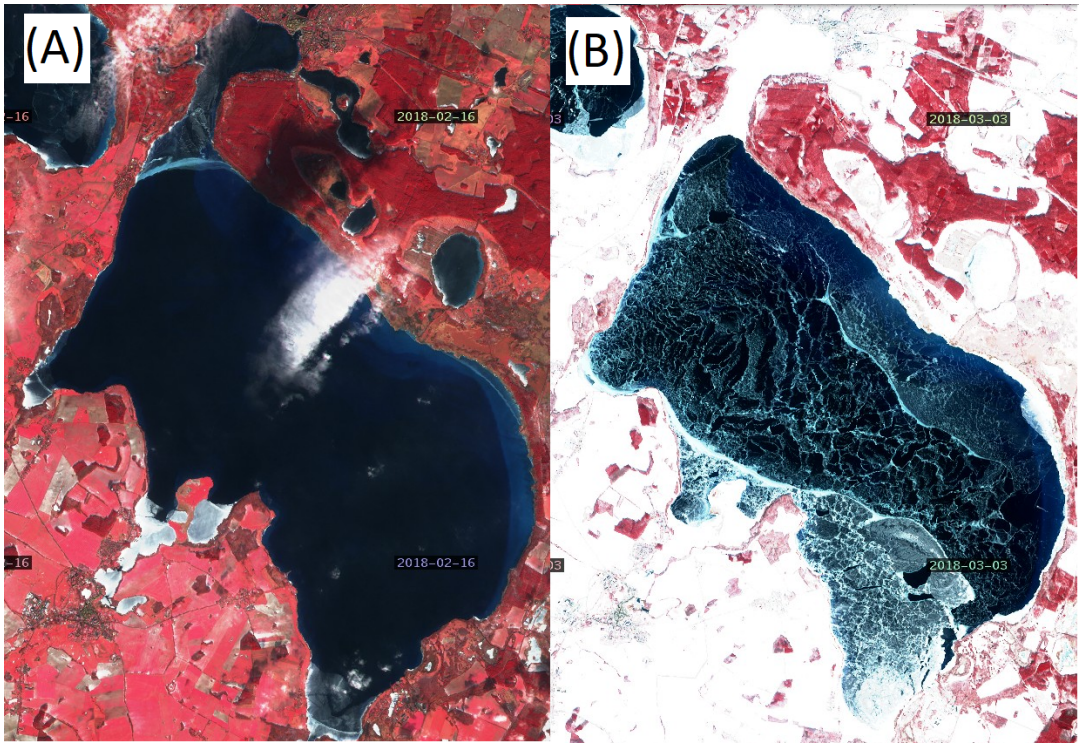

FIGURE 3.9: Images taken for **validation** of the results. (A) False color composite (B8-B4-B3) of "Müritz" on 16th February 2018 showing ice in the northern part ("Binnenmüritz") and in the smaller bays in the south. (B) False colour composite (B8-B4-B3) of "Müritz" on 3rd March 2018. The whole lake is covered with ice, the "Binnenmüritz" is covered by snow. (sources: https://sentinel-hub.com)

The thresholds were picked by hand and adjusted using the results to increase or decrease the values. In the end the best results were produced using the following thresholds:

Band	Threshold	Ice below or above threshold
VV	0.01	below
VH	0.0018	above
Ratio	6.6	below

TABLE 3.1: Threshold values for different bands. Note the values for VV and VH are not recalculated in dB but are given in floating point numbers. The numbers result in a good fit for the first images.

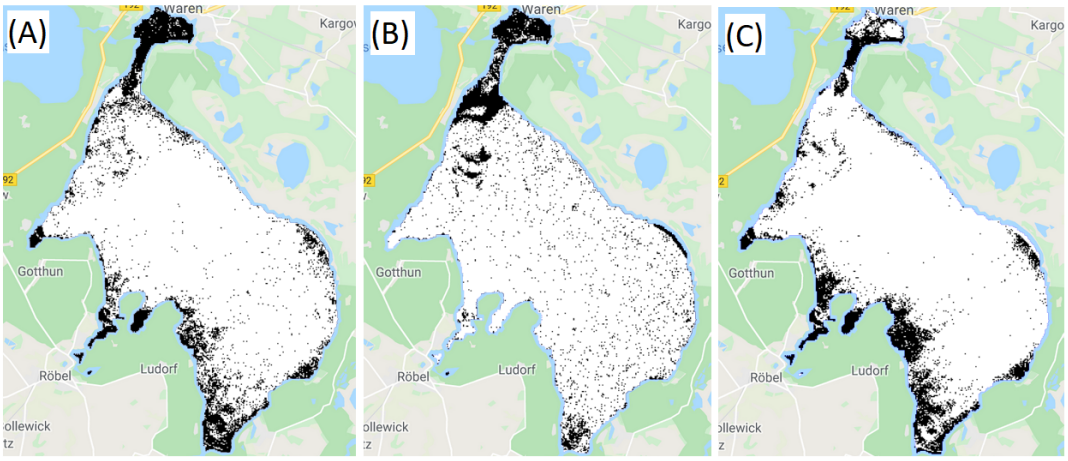

FIGURE 3.10: The results using the thresholds in table 3.1 on 16th February 2018 at "Müritz". Black areas represent ice, white areas represent water.(A) Ratio-threshold. (B) VH-threshold. (C) VV-threshold. All images are screenshots taken of the GEE.

Comparing the different results shown in figure 3.10 to the validation image (figure 3.9, (A)) shows that both the ratio-threshold and the VH-threshold show promising results. The completely frozen state of the "Binnenmüritz" is reflected in the results, as well as some minor frozen areas in the smaller bays at the south of the lake. The VV-threshold is working quite well, even though some parts are misclassified as ice but are water in reality.

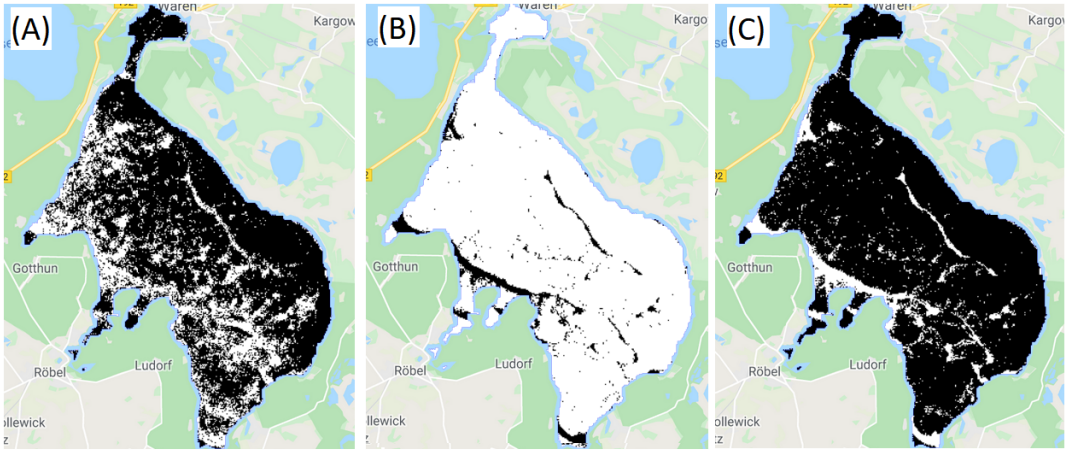

FIGURE 3.11: Results using the thresholds in table 3.1 for 2018-03-03 at the "Müritz". Black areas represent ice, white areas represent water.(A) Ratio threshold. (B) VH threshold. (C) VV threshold. All images are screenshots taken of the GEE.

The results for the 3rd March 2018 are showing that the threshold for the VH values that seemed to fit quite good for the 16th February (figure 3.11) is completely off. The whole lake is characterized as not frozen even though it is nearly completely frozen (figure 3.9, (B)). Figure 3.11 (A) is showing that the ratio-threshold is not that good as well, since there are a lot of areas classified

as not frozen. The best representation at this stage of the process is coming of the threshold for the VV-values. Not only visual interpretation was performed, I also checked how good each threshold is representing the classes ice and water for both dates. The data is supporting the visual interpretation.

2018-02-16				
Reality	Classified as	Ratio-Threshold	VH-Threshold	VV-Threshold
Ice	Ice	13.00 %	13.35 %	9.07 %
	Water	3.82 %	4.47 %	7.75 %
Water	Ice	18.26 %	7.79 %	19.09 %
	Water	64.92 %	75.39 %	64.09 %
Correct Classification		77.92 %	88.74 %	73.16 %

TABLE 3.2: Percentage share of the classification using the different thresholds in table 3.1 for 2018-02-16.

2018-03-03				
Reality	Classified as	Ratio-Threshold	VH-Threshold	VV-Threshold
Ice	Ice	69.52 %	8.71 %	84.77 %
	Water	30.48 %	91.29 %	15.23 %

TABLE 3.3: Percentage share of the classification using the different thresholds in table 3.1 for 2018-03-03. The whole lake is covered by ice, thus the correctly classified values are the same as the ice-values.

To see if the defined thresholds are usable on the same date but for another lake they were used to classify the "Barniner See" in Mecklenburg-Vorpommern. Figure 3.12 is showing a satellite data image that was used for validation of the results. The lake is almost completely frozen, besides a small area in the north-eastern part.

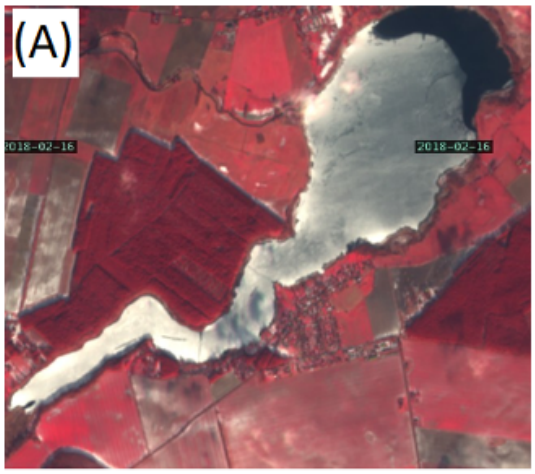

FIGURE 3.12: Validation false color composite (B8-B3-B3) of "Barniner See" on 16th February 2018. (source: https://sentinel-hub.com)

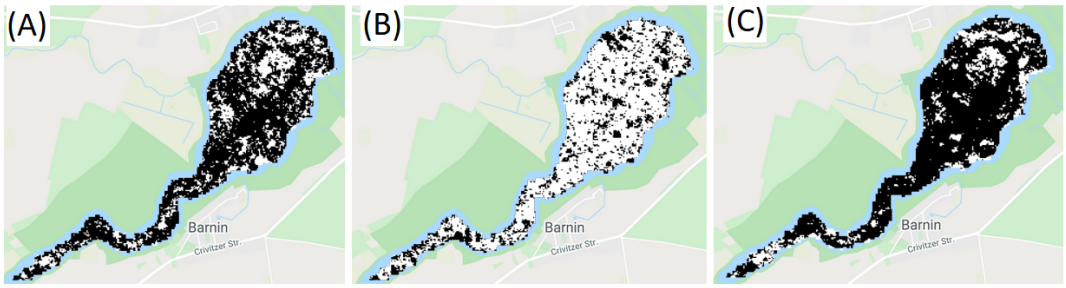

FIGURE 3.13: The results using the thresholds in table 3.1 for 2018-02-16 at "Barniner See". Black areas represent ice, white areas represent water. (A) Ratio-threshold. (B) VH-threshold. (C) VV-threshold. The images are screenshots from the GEE.

Comparing figure 3.12 to figure 3.13 shows there is no real use of the thresholds working for the "Müritz" at the "Barniner See" on the same day. The results show that they have problems to represent the surface state of the lake, with the VV-Values being the best of three bad options, based on visual interpretation.

2018-02-16				
Reality	Classified as	Ratio-Threshold	VH-Threshold	VV-Threshold
Ice	Ice	60.5 %	62.74%	33.46 %
	Water	23.45 %	21.21 %	50.49 %
Water	Ice	9.61 %	9.19 %	7.79 %
	Water	6.38 %	6.86 %	8.27 %
Correct Classification		66.88 %	69.6 %	41.73 %

TABLE 3.4: Percentage share of the threshold values compared to Sentinel-2 derived real data for "Barniner See".

Table 3.4 is supporting the stated fact that the results are not usable since there are too many misclassified pixels.

3.1.3 Unsupervised classification

The unsupervised classification led to good results for big lakes if the scale was set to 30 m/px. The code for the creation of the results can be found in Appendix A.

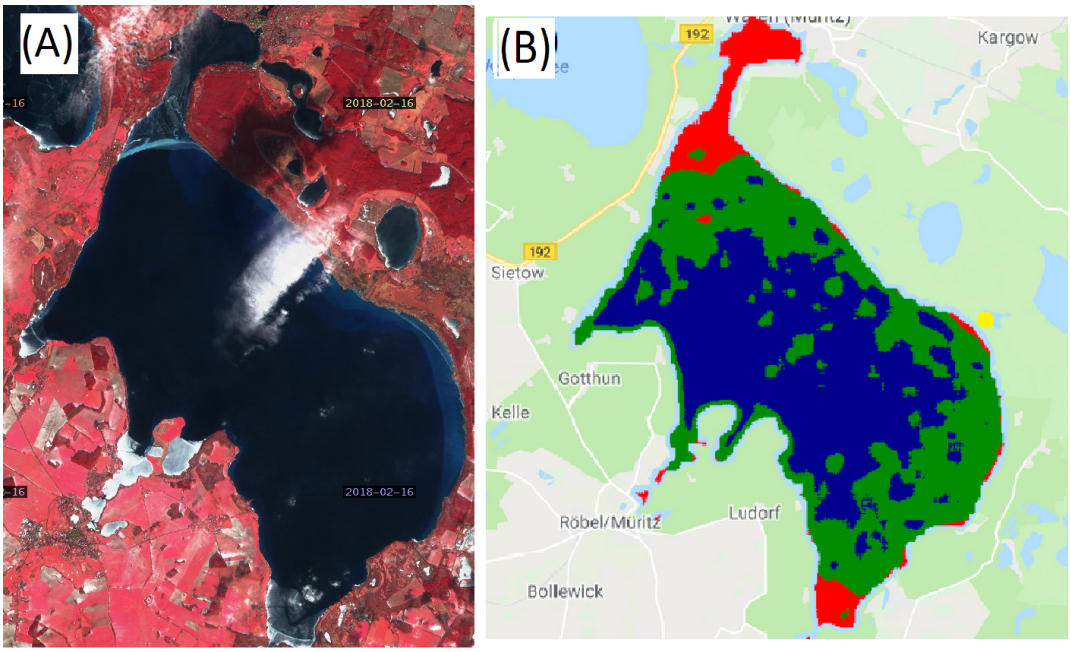

FIGURE 3.14: "Müritz" on 2018-02-16. (A) False color composite (B8-B4-B3). (source: https://sentinel-hub.com)) (B) Result of the unsupervised classification with 3 classes and the scale set to 30m/px. The red colored pixels represent ice, the two other classes represent water. The small yellow dot is a hint where even small ice areas get recognised quite well. (Screenshot of GEE)

Figure 3.14 and table 3.5 show that the unsupervised classification is working to at least some degree. In this case the red class indicates ice, the other two classes indicate water. This is not really representing the three classes of water, wavy water and ice, since the pattern does not look like it is caused by waves. But the detection of ice is working. Especially on the borders of the lake, e.g. in the north-eastern part (yellow dot), even small areas of ice are detected. But there are also some areas that got misclassified as ice like the small red part in the middle of the green class in the north west of the lake.

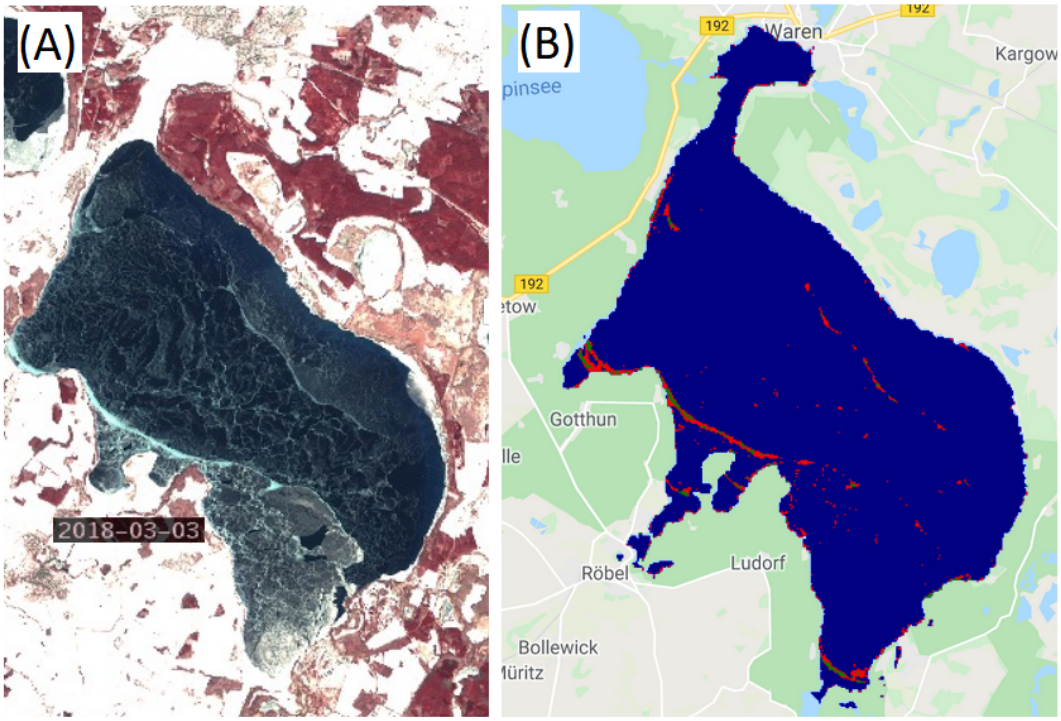

FIGURE 3.15: "Müritz" on 2018-03-03. (A) False color composite (B8-B4-B3) of the Lake. (source: https://sentinel-hub.com) (B) Result of the unsupervised classification scale is set to 30. The blue colored pixels are representing ice. (Screenshot of GEE)

Figure 3.15 shows another example for ice on the lake and the result of a classification. Once again it looks like the result is quite good since the whole lake is covered in ice and nearly the whole area was classified as one class. Due to the random seeding of the class-centroids in the beginning the ice is now in another class as shown in figure 3.14, but this could have been solved by comparing the mean values of the classes if the results were used in further analysis. The percentual share of correct and misclassified pixels in both classes were calculated. The two classes that are not ice were combined as one water class as suggested [25].

Reality	Classified as	2018-02-16	2018-03-03
Ice	Ice	12.39 %	96.70 %
	Water	7.04 %	3.30 %
Water	Ice	8.24 %	0 %
	Water	72.33 %	0 %
Correct Classification		84.72 %	96.70 %

TABLE 3.5: Percentual share of water and ice classified pixels of the total amount of pixels. The overall fit for "Müritz" is quite good on both dates. Note that the lake was completely frozen on 2018-03-03, thus no pixels are water in reality.

Since it worked quite well on big “Müritz”, the classification was also used on the smaller “Barniner See”. Unfortunately figure 3.16 shows that for small lakes – regardless of the scale set to 10 m/px or 30 m/px - the classification is not working at all.

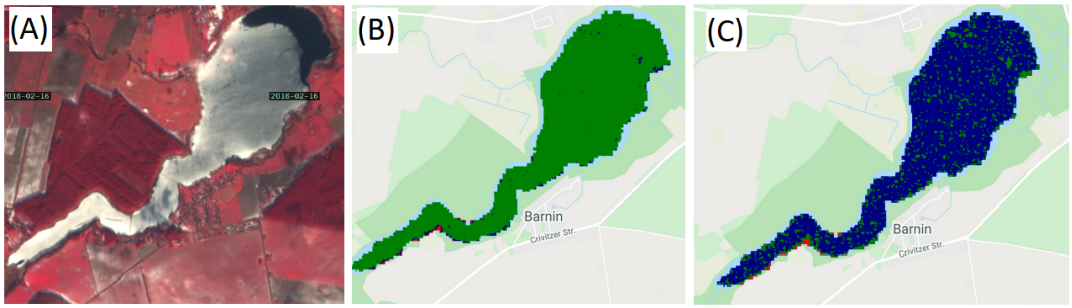

FIGURE 3.16: “Barniner See” on 2018-02-16. (A) A false colour image (B8,B4,B3) of the lake. (source: https://sentinel-hub.com) (B) The result of the unsupervised classification with the scale set to 30m/px.(C) The result of the unsupervised classification with the scale set to 10m/px. The results are screenshots of GEE results.

Due to the bad results it is not even possible to assign different classes to the different colors shown in the figure. If one would assign the class ice to the biggest class (green for (B) and blue for (C) on figure 3.16) the goodness of fit can be calculated(see 3.6).

2018-02-16			
Reality	Classified as	Scale 10 m/px	Scale 30 m/px
Ice	Ice	79.80 %	77.74 %
	Water	5.00 %	7.06 %
Water	Ice	14.36 %	13.42 %
	Water	1.70 %	1.78 %
Correct Classification		81.5 %	79.62 %

TABLE 3.6: Percentual share of water and ice classified pixels of the total amount of pixels for the "Barniner See" on 16th February 2018.

Even though table 3.6 indicates good results for "Barniner See" due to a relatively high accuracy in total correctly classified pixels, this is mostly due to the low amount of water pixels in the lake at this day. Nearly 0% of the water pixels get classified as water, thus this method should not be taken into account to detect water on small lakes.

The number of clusters was switched between 3 and 6, since doubling the number of clusters and recombining them afterwards can be used to increase the accuracy of the result. Unfortunately this did not change anything in the results or improve the classification, it even lead to worse results for the big lakes.

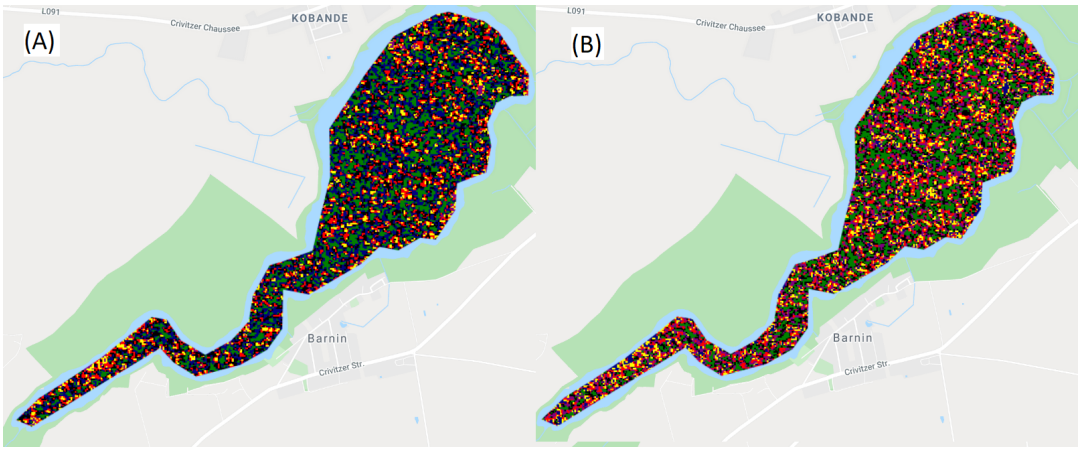

FIGURE 3.17: Examples for unsupervised classification using 6 classes on "Barniner See" on 2018-02-16. (A) Scale set to 10m/px. (B) Scale set to 30m/px. The results are screenshots of GEE results.

Figure 3.17 shows that the result is not improving if six classes were used on a small lake.
Since the results for small lakes – which are common in Scandinavia – were that bad the approach was neglected even though it might be useful if one was interested in big lakes.

3.2 SLC products

Both satellite data (figure 3.18) and the data of the IGB (see Appendix C) show that "Nehmitzsee" was frozen 6th February 2018 (100% ice cover) and 23rd March 2018 (80% ice cover). "Großer Stechlinsee" only froze up to 45% on 8 March 2018. The ice on "Großer Stechlinsee" was mainly present on the two sidearms in the south-western and the western part.

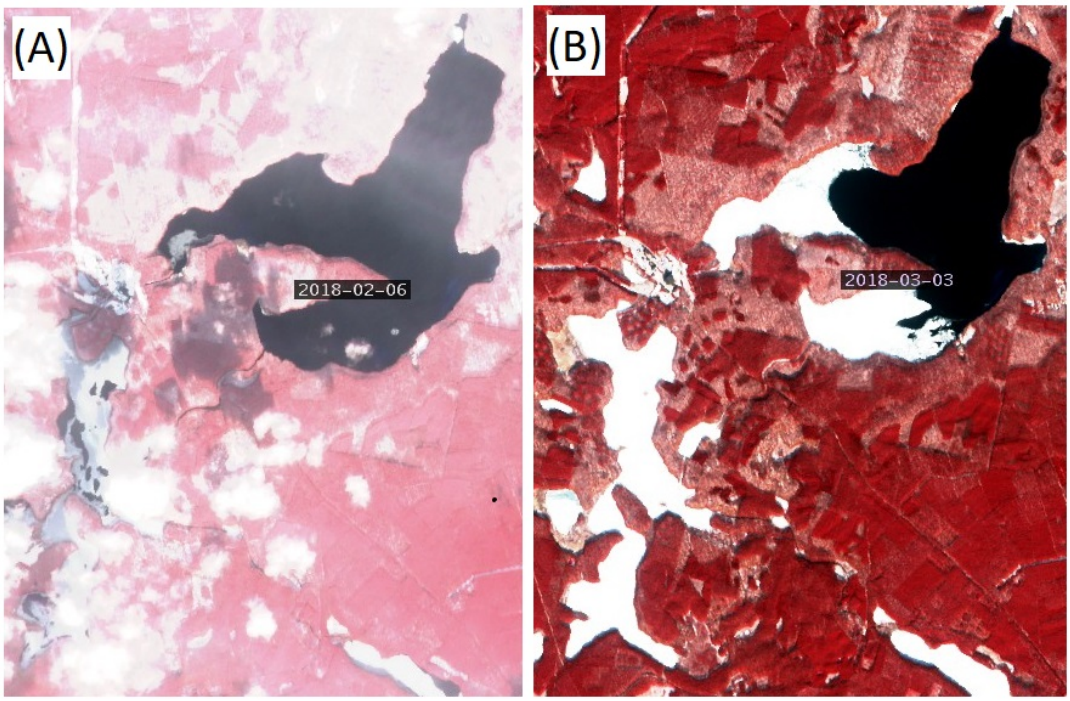

FIGURE 3.18: False color composites (B8-B4-B3) of "Nehmitzsee" and "Großer Stechlinsee" in Brandenburg showing different ice extents on different dates. (A) A cloudy scene on 6th February 2018. (B) A scene taken on 3rd March 2018. (source: https://sentinel-hub.com)

3.2.1 Topophase

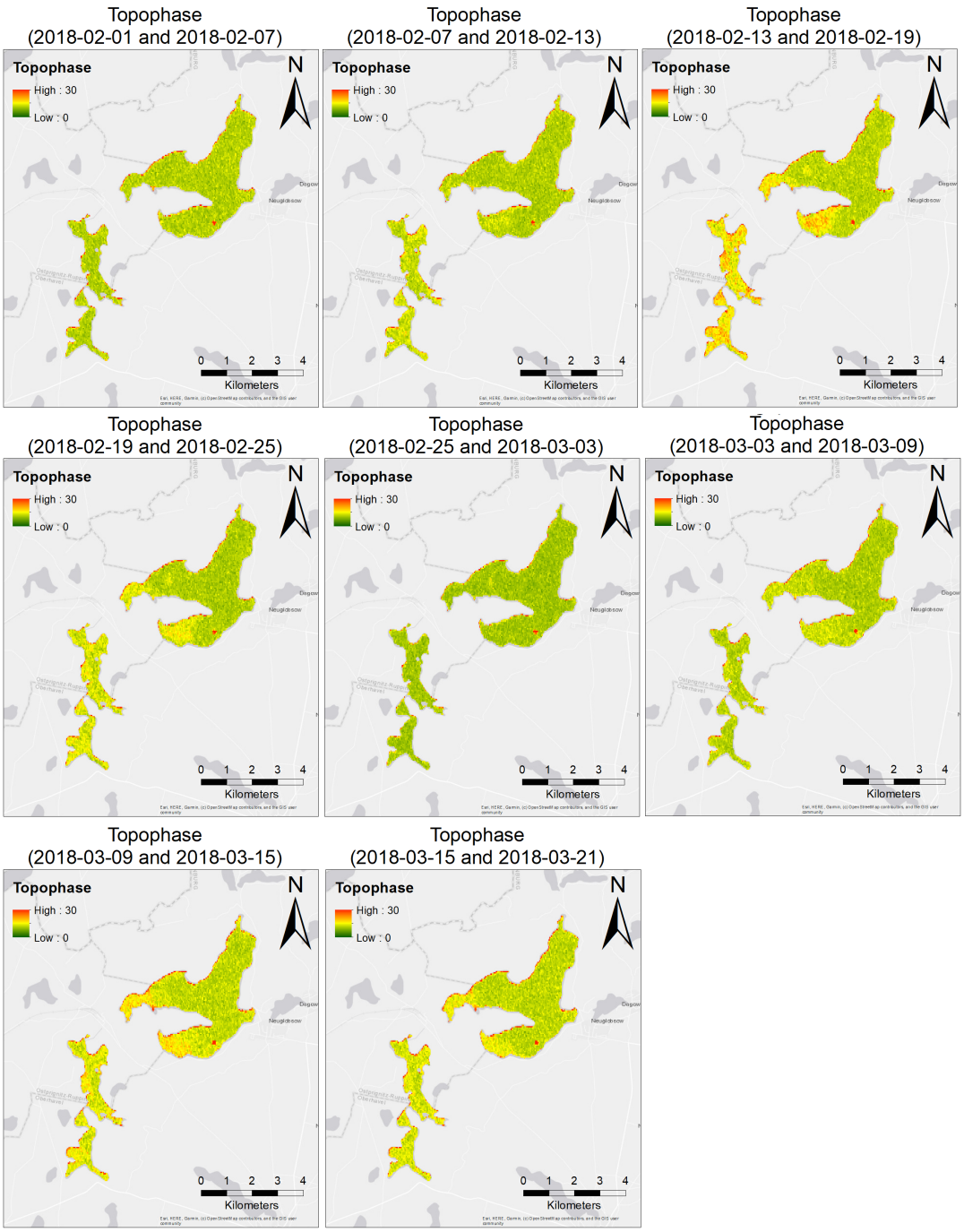

FIGURE 3.19: A time series showing the *topophase.cor.geo* band 2 results for one orbit of Sentinel-1 data starting on 1st February 2018 up until 21st March 2018 for "Nehmitzsee" and "Großer Stechlinsee".

During the first time step there is no significant signal thus no significant change visible in the topophase band 2 (Figure 3.19), even though "Nehmitzsee" is freezing up from 10% ice cover up to a complete ice cover. A first increase in "Nehmitzsee" and small parts of "Großer Stechlinsee" (south-western part)

between 2018-02-07 and 2018-02-13 are visible, which is increasing even more in the next time-step. The change seems to be more accurate for "Großer Stechlinsee" up to the time-step 2018-02-25 to 2018-03-03, when both lakes show values that are comparable to the initial time-step, even though the ice-extent of "Großer Stechlinsee" is increasing again. The increase at the end of the time series is not explainable by the ice data, since "Nehmitzsee" is not changing its ice-cover up to the end of March.

3.2.2 Coherence

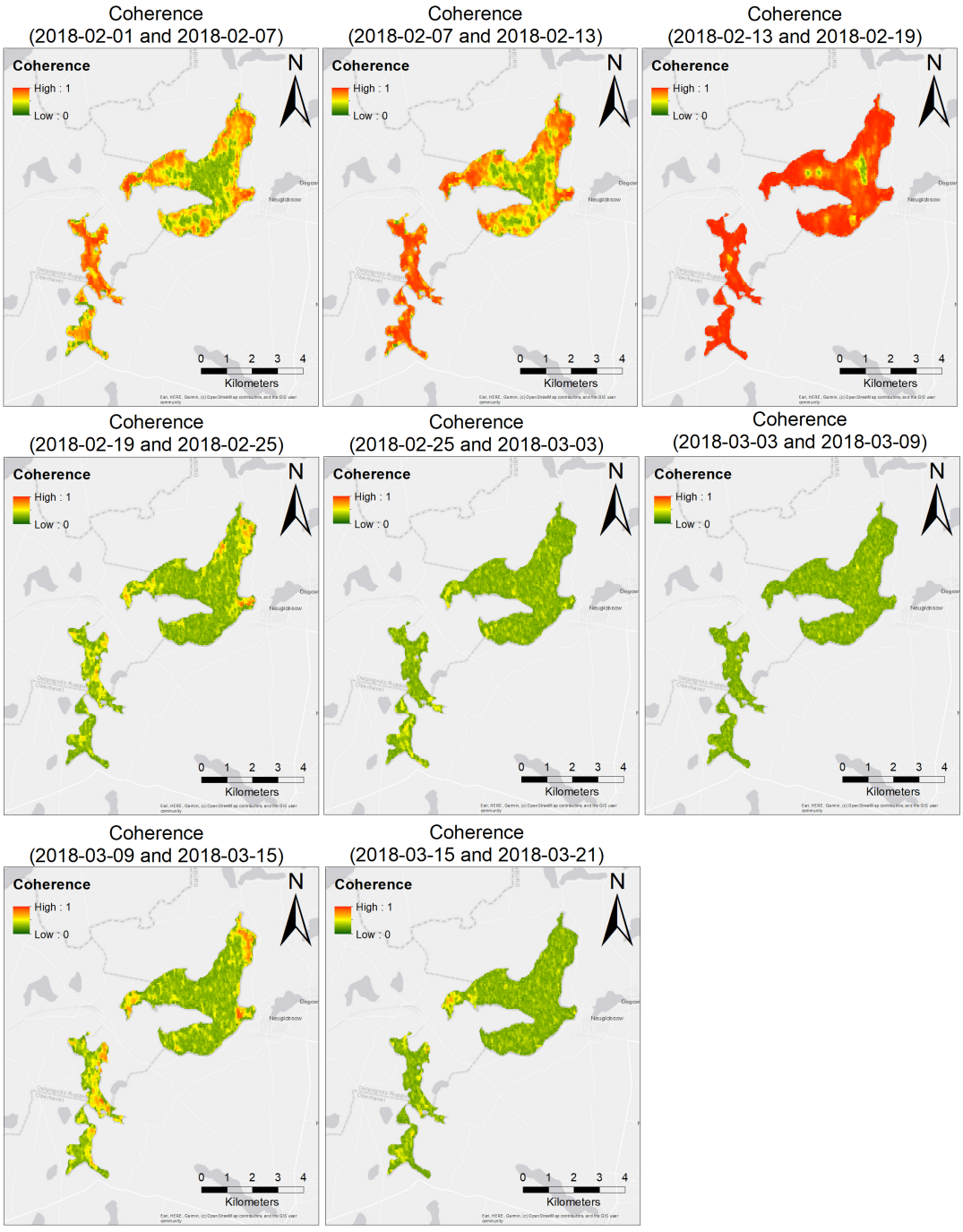

FIGURE 3.20: A time series showing the *phsig.cor.geo* for one orbit of Sentinel-1 data starting on 1st February 2018 up until 21st March 2018.

The coherence time series (figure 3.20) shows – like the *topophase* – an increase up until the 3rd time-step between 2018-02-13 and 2018-02-19. But comparing this increase to the satellite data (figure 3.18) shows that parts of Großer Stechlinsee are not frozen at all, despite showing high coherence values.

3.2.3 X-Polarised interferograms and coherence

X-Polarised interferograms were calculated for "Müggelsee", “Großer Stechlinsee”/”Nehmitzsee” and “Västra Laxsjön” and “Östra Laxsjön” in Sweden. The results that were used were the *phsig.cor.geo* for the coherence between the VV and VH bands in the image and the result called *topophase.cor.geo* as a representation for the different responses of the waves to the surface state. The lakes in Sweden were not frozen on 16th November 2018, but a satellite image recorded on 26th November shows signs of ice on the surface of the lakes (figure 3.21).

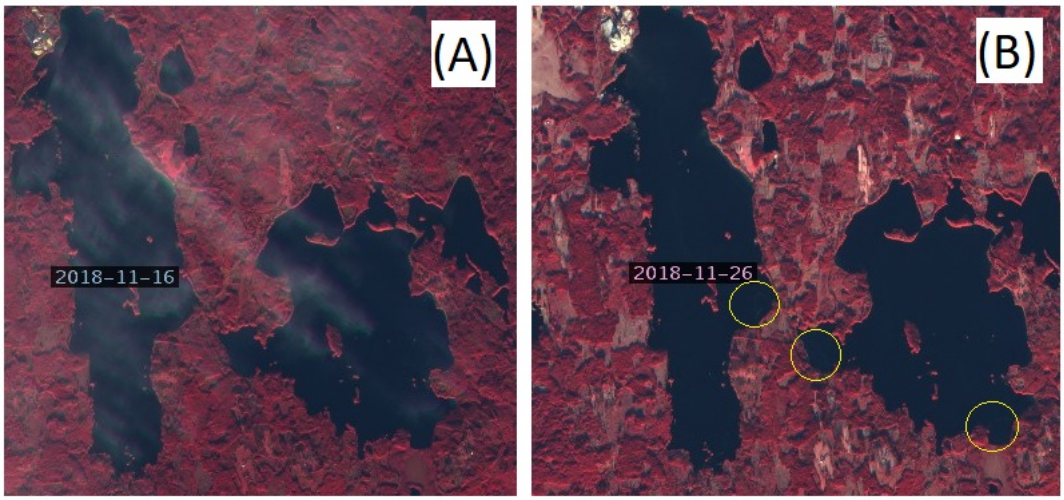

FIGURE 3.21: The freezing time of “Västra Laxsjön” (left) and “Östra Laxsjön” (right) in Sweden.(A) False-color composite (B8-B4-B3) on 16th November 2018 showing no signs of ice. (B)False-color composite (B8-B4-B3) on 26th November 2018 showing a frozen surface. Ice features like small cracks are visible in the yellow circles. (source: https://sentinel-hub.com)

Topophase

The results for “Müggelsee” (see figure 3.22) show that the *topophase* band 2 has higher values on 4th March when the whole lake was covered by clear ice. The values decrease when the ice is getting thinner (2018-03-10 and 2018-03-22) and shows a little response to the snow on top of the ice on 22nd March. Unfortunately on 2018-02-26 there are equally high values present as on 2018-03-10, even though there is no ice on the lake, only on the northern shoreline.

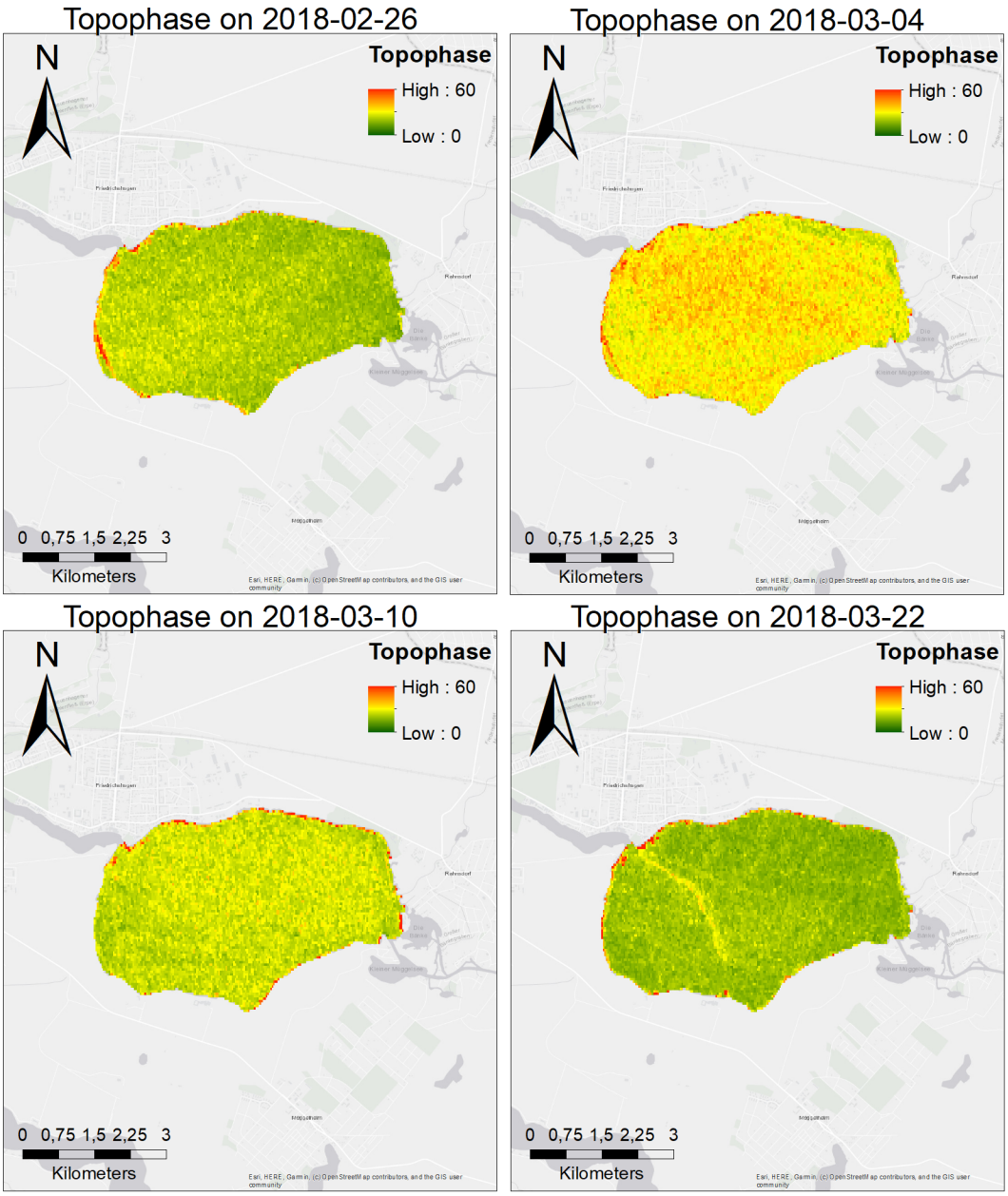

FIGURE 3.22: A time series showing the *topophase.cor.geo* for one orbit of Sentinel-1 data starting on 1st February 2018 up until 21st March 2018 for "Müggelsee".

Both "Nehmitzsee" and "Großer Stechlinsee" were ice-free at the beginning of the short time series shown in figure 3.23. "Nehmitzsee" was completely frozen starting at the 6th February up until the end of March, whereas "Großer Stechlinsee" did not have more than 45% ice cover in this winter, the maximum amount of ice in this time series is 38 – 42% between 2018-02-22 and 2018-03-01.

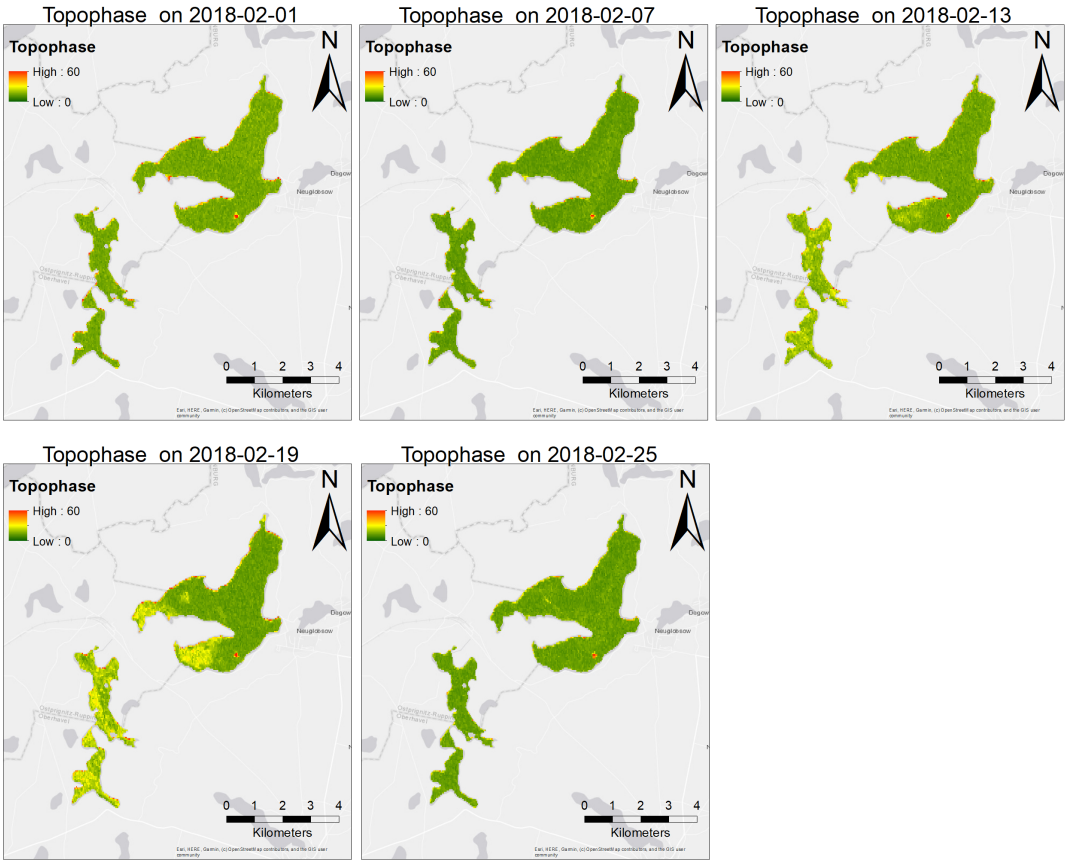

FIGURE 3.23: A time series showing the *topophase.cor.geo* band 2 of the X-Polarized interferograms for the "Nehmitzsee" and the "Großer Stechlinsee".

There are some well-represented features like the frozen part of "Großer Stechlinsee" on 2018-02-19, but the snow cover seen in the satellite images (figure 3.18) is not represented. There is no indication for ice or snow on the lake. Maybe this is caused by snow accumulating on the ice. This might be an even bigger problem for Scandinavia, where snow is occurring quite often when it is cold enough for lakes to freeze.

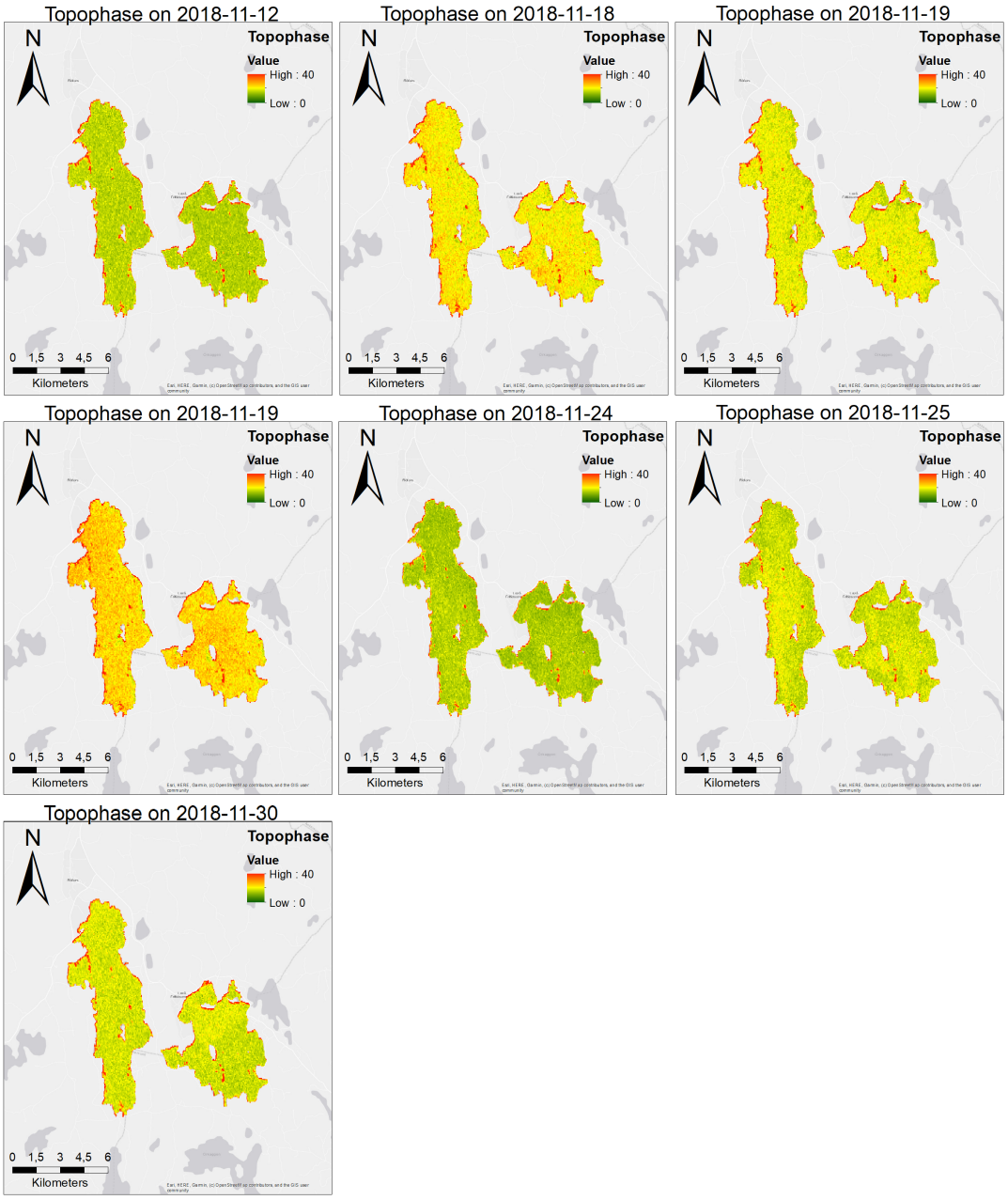

FIGURE 3.24: A time series showing the *topophase.cor.geo* band 2 of the X-Polarized interferograms for "Västra Laxsjön"(left lake) and "Östra Laxsjön" (right lake). Data is not taken from one single orbit. On 2018-11-19 both data from Sentinel-1 A and Sentinel-1 B are available (upper right is Sentinel-1 A data, middle left is Sentinel-1 B data. Even on the same date there are differences in the values.

Figure 3.24 shows a time series for the topophase band 2 for two lakes in Sweden. The time series indicates that something happened between 2018-11-12 and 2018-11-18/2018-11-19, maybe a freezing process but unfortunately the 24th looks quite low in terms of band-value for the topophase and the re-increase of the values after the 24th are not explainable using the given data. There is no indication for a thawing and refreezing process on

satellite data. The different values for the same date are indicating that a change happened throughout the approximately 11 hours between the acquisitions. This might be caused by early morning freezing of the lake which is not present in the afternoon [6]

Coherence

"Müggelsee" shows no significant change in the coherence during the time series (Figure 3.25). Only the north-western part shows high values of coherence which would represent similarity in the reflection of VV and VH waves. But since there is no change throughout the month of investigation there is no hint that this could be used to determine ice-cover.

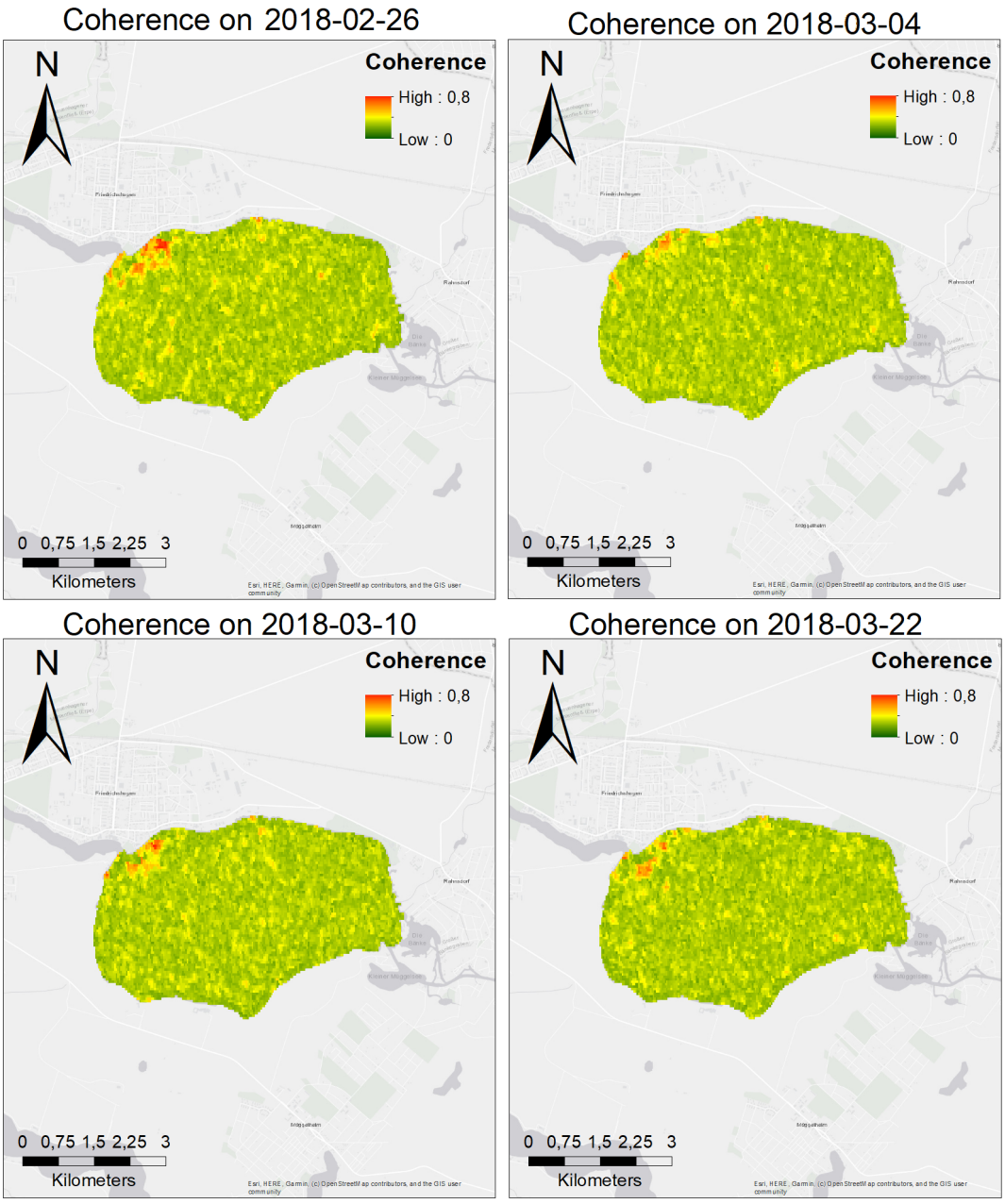

FIGURE 3.25: A time series showing the *phsig.cor.geo* for one orbit of Sentinel-1 data of the "Müggelsee". The coherence is not changing a lot even though the ice cover of the lake is increasing and decreasing again in this time frame according to the IGB data (Appendix C)

The time series for the coherence on "Nehmitzsee" and "Großer Stechlinsee" (figure 3.26) shows – like the time series for the "Müggelsee" – no significant change.

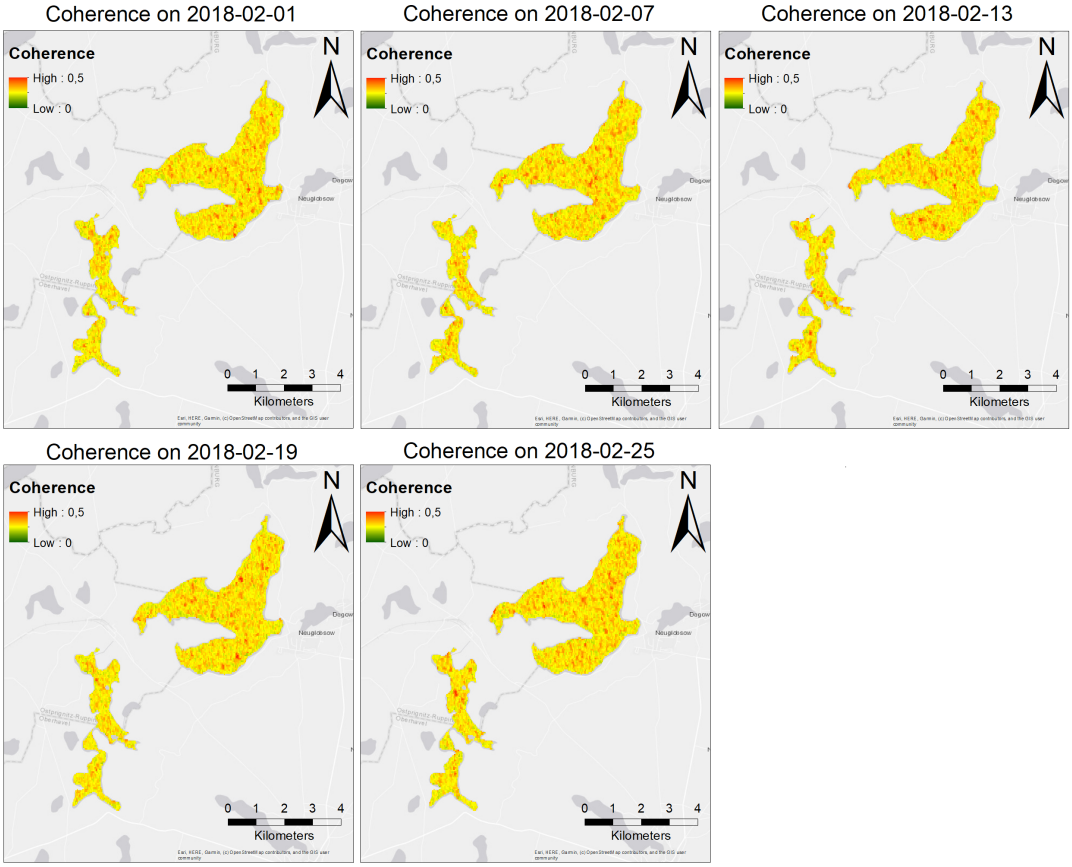

FIGURE 3.26: A time series showing the *phsig.cor.geo* for one orbit of Sentinel-1 data on "Nehmitzsee" and "Großer Stechlinsee". The coherence is not changing significantly during the time covered by the time series even though there is significant change in ice cover (see Appendix C).

The coherence value is not significantly changing on the lakes in Sweden over the time series (figure 3.27). There is no indication that this could help to recognize ice.

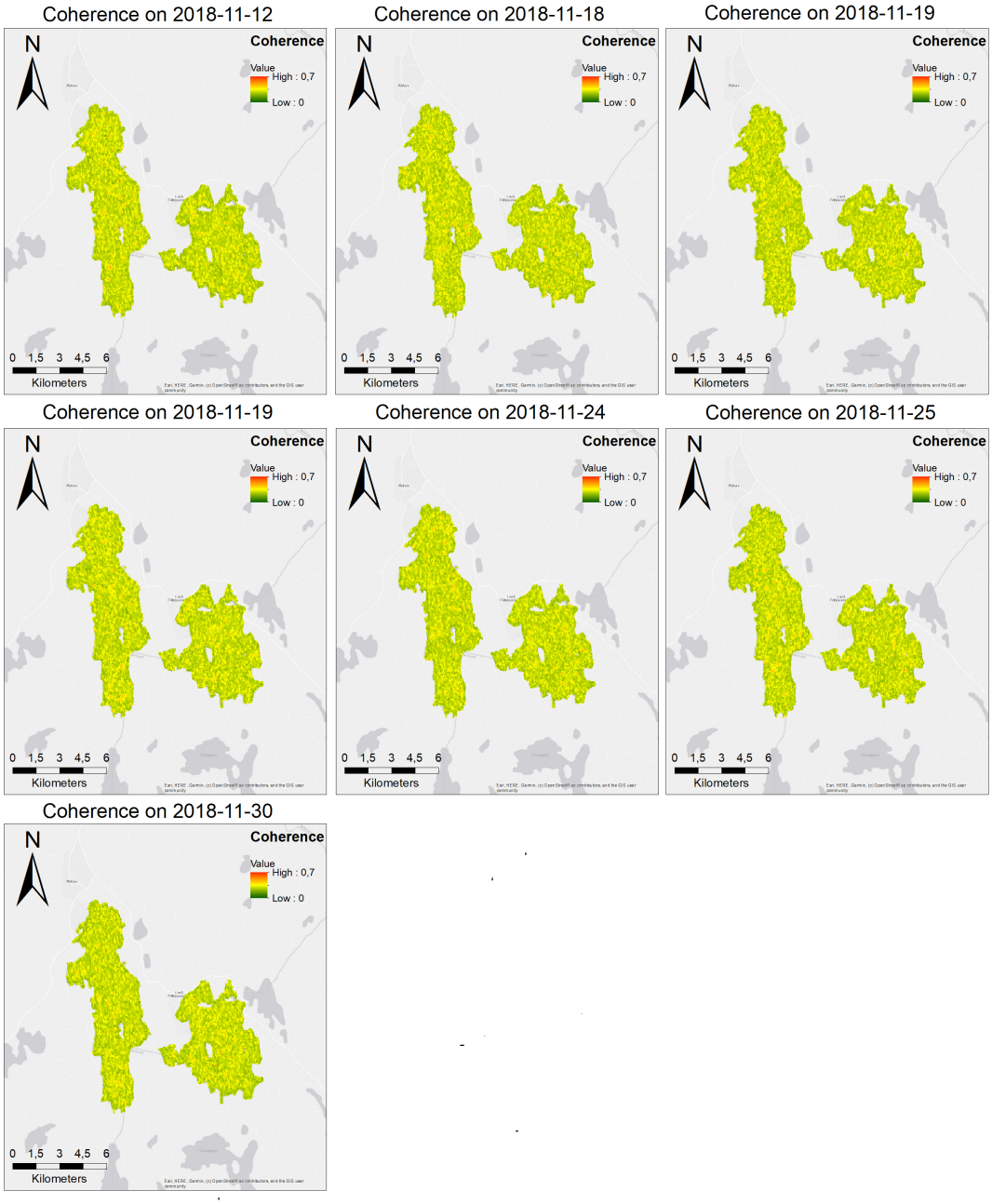

FIGURE 3.27: A time series showing the *phsig.cor.geo* for one orbit of Sentinel-1 data on "Västra Laxsjön"(left lake) and "Östra Laxsjön" (right lake). No change is present in the time series but the lakes are freezing, see figure 3.21.

4 Conclusion

Looking at all the results in the previous chapter shows a big problem: there is no approach that is working perfectly. Single approaches seem to work for different lakes or different times, but they are not applicable to all images or lead to promising results for all lakes.
The best results were accomplished by using a time series of the ratio value for a lake-wide mean. Thus I decided to use this method and try to specify further usage by **comparing the values to a long-term mean (LTM)**.

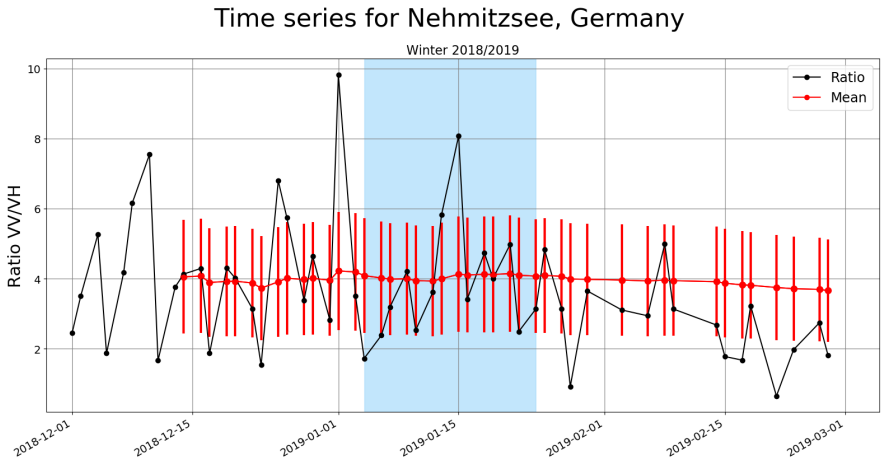

FIGURE 4.1: Resulting time series of the ratio VV/VH for "Nehmitzsee" in winter 2018/2019. The red dots are showing the LTM, the vertical red lines are showing ±40% deviations of the LTM. The blue box is showing the freezing period as indicated by the IGB data.

Figure 4.1 shows a time series of the lake wide mean of the ratio $\frac{VV}{VH}$ of "Nehmitzsee" in Brandenburg in winter 2018/2019. It is visible that the lake wide mean shows a single high peak in the freezing period of the lake compared to long term mean ±40% of the lake wide mean. Unfortunately there are peaks before the freezing period, thus the lake would be classified as frozen to early. There is no data before 4th February 2019, thus I can not be 100% sure that the lake was not frozen at all before.

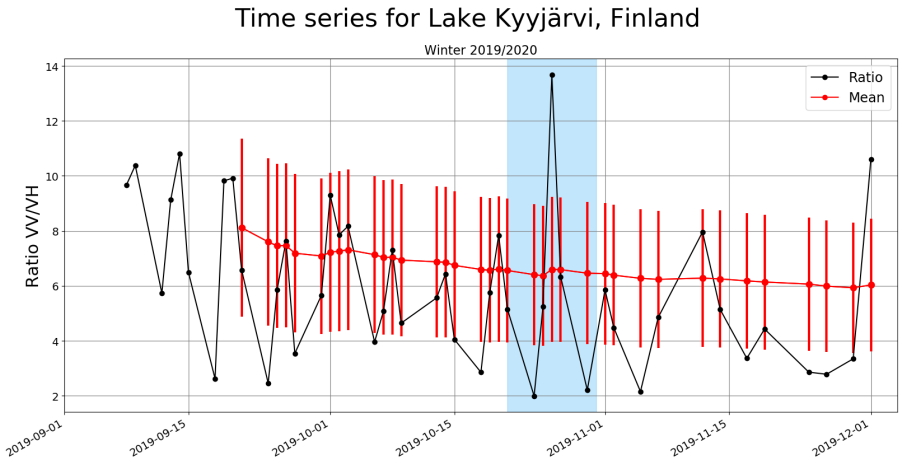

FIGURE 4.2: Time series of the ratio VV/VH for "Lake Kyyjärvi". The red dots are showing the LTM, the vertical red lines are showing ±40% deviations of the LTM. The blue box is showing the freezing periods.

"Lake Kyyjärvi" is showing a different behaviour. The only time the ratio-value is far above the long-term mean is inside the freezing period which was derived from satellite data.
The data for "Lake Kyyjärvi" is supporting the idea that using a comparison of the data with the LTM is usable. It is one of the fastest and most accurate approaches, thus I decided to use it as an indicator, even though I know that it is not perfect.
The result of my work is the following: As soon as a lake-wide mean value is higher than the LTM+40%, the lake is considered being frozen.
I decided to assign one of two states to the lakes. If they are not frozen the state is 0, once they are freezing according to the method the state is changed to 1.

State	Meaning
0	Not Frozen
1	Frozen

TABLE 4.1: The different states and their meanings assigned due to the used algorithm.

5 Automation

The results have shown that the best way to detect the ice in this work was a simple time-series of the ratio $\frac{VV}{VH}$.
Since the ratio was used, it was possible to use the GRD data and thus save a lot of disk space and decrease computational time in comparison to the usage of SLC data.
There are four steps that have to be done:

1. Download all GRD scenes of the area of interest with a given time span (I think the best way is to do this daily, so in the following text I will assume one is doing this daily).

2. Process the downloaded data using the SNAP Graph processing tool.

3. Extract the ratio values for all lakes and check if the values have increased a lot in comparison to the mean over the whole series.

4. Map and export the data automatically.

I created one batch-file for the whole process which contains all of the steps and uses different python environments, different programs and runs on its own once it is started.
The different scripts named here can be found in Appendix B

5.1 Downloading

The data is downloaded using sentinelsat [60] which can be used to search and download Copernicus Sentinel satellite images. It is possible to install sentinelsat through pip and it is necessary to have an account on the Copernicus web service to download the data. Sentinelsat is a command-line tool:

```
1    sentinelsat -u <user> -p <password> -g <search_polygon.geojson> -s
     ↪ 20150101 -e 20151231 -d --producttype GRD --url
     ↪ "https://scihub.copernicus.eu/dhus"
```

The parts user, password and url can be set as environment variables in Windows which is even shortening the command. The following variables have to be set: *DHUS_USER* = *<user>*, *DHUS_PASSWORD* = *<password>* and *DHUS_URL=<https://scihub.copernicus.eu/dhus>* (all without the < >).
The *-g* for geometry is defining the geometry of the search, so it is the outline of my area of interest around Scandinavia that is shown in figure 1.1.
-s and *-e* are defining the start and the end dates of the search, if they are not

defined only data of the last day will be investigated. *-d* is defining that the data should be downloaded and saved on the hard drive.
–producttype is defining the product type that should be downloaded There are plenty of other settings that can be used to specify the search, like queries to define polarisations. Those can be found in the documentation [60].
In the end the command used in this work looks like this (with set environment variables, for daily use):

```
sentinelsat -g outline.geojson -d --producttype GRD
```

The data will be stored as .zip files in the current directory

5.2 Preprocessing with SNAP

To preprocess the data the graph processing tool (GPT) of SNAP was used. It allows you to execute graphs on data outside of SNAP using the command line [44]. To process all of the data in one step I created a for-loop inside of my batch-file.

```
for %%X in (*.zip) do (gpt GRD_Preprocess.xml -PFilename="%%X" -t
↪ "G:\MT\PreprocessedData\%%~nX.dim")
```

This line of code is iterating through all files that end on .zip in the directory, i.e. all downloaded .zip-files of step 1, and processing them using the *GRD_Preprocess.xml* that was described in Section 2.4. Since no precise Orbit-files will be available on the day after the images were taken, I removed that part of the graph to decrease computation time.
The *-PFilename=”%%X”* is setting the input variable to the currently iterated file and *-t (-target)* is defining where the preprocessed data should be stored with the same name as the input file.
(NOTE: If this should be run outside of a batch-file in a command line interface only use a single % sign in front of all the variables.)

5.3 Extracting values and updating tables

To do this step it is necessary to have the following setup:

- an executable **python 2** distribution called **python27.exe** with the following packages and their dependencies installed: arcpy, datetime, numpy (comes with ArcMap 10.1, need to rename python.exe to python27.exe to not mix up with the python3 distribution)
- an executable **python 3** distribution called **python.exe** with the following package and their dependencies installed: numpy, datetime, glob, pandas.
- A *lakes.shp* file containing all lakes inside of the area of interest (minus 10m buffer on each side for mix pixels)
- Three files named: *lakes.csv*, *LTM.csv* and *State.csv*. The *lakes.csv* contains the **FIDs** of the lakes, and the mean values of the different dates. The *LTM.csv* includes the **FIDs** and - starting after 10 mean values are added to the *lakes.csv* - the mean value of the time series up to this date. The *State.csv* consists of two columns: the **FIDs** of the lakes and the **State**.

(If you have a python environment with all of those packages it should be possible to use only one of them, I was not able to install arcpy on my python 3 environment and could not install any packages in the ArcMap-delivered python 2 environment.)

The third step of the processing is happening in two steps :

```
1  for /R  G:\MT\PreprocessedData %%R in (*.img) do python27
   ↪  zonalstatistics.py %%R lakes.shp
2  python TimeSeries.py
```

The first line of this part of the code is iterating recursively through all the preprocessed data and looking for .img files which is the format the data is stored inside of SNAPs BEAM-DIMAP-format.
For each file it is running *python27 zonalstatistics.py* which is extracting the **count** of pixels and the **sum** of the values inside each polygon inside the lakes.shp. For each scene the data is stored as a text file with the corresponding FIDs for the lakes. No scene will cover the whole area, so some FIDs will not be in the text files but this is no problem in further processing.
The second line is using all the text files created in the first step and calculate the mean for every FID/lake if there are several measurements for a day (sometimes both Sentinel-1A and Sentinel-1B are covering lakes at one day). The mean is calculated by adding up all the sums and the counts respectively, and dividing the sum by the count afterwards. This is taking into account that not all lakes are fully covered by a scene sometimes. The mean values will be added to the *Lakes.csv* according to the FID, the lakes that were not covered that day will be assigned a NaN for that day.

The code is further importing the *State.csv* and the *LTM.csv*, calculating a new LTM and adding it to the table, when more then 10 lake-wide means are calculated. Once the lake-wide mean value for a day is higher than 1.4 times the long-term mean, the state of the lake will be changed to 1.

5.4 Mapping and exporting

It is necessary to have a Mapping.mxd map project that contains the lakes.shp with set symbology for different states. The code line for the last part is:

```
1    python27 ReintroduceArcpy.py
```

The code is importing both the *lakes.shp* and the *State.csv* files. The state column of the shapefile is getting deleted, and the updated column of the *State.csv* is merged to the shapefile based on the FID. Since the symbology is set already it will get updated by new values in the state-field of the shapefile. A PDF-file will be saved with a map including a basemap and the lakes with their corresponding states. After this the satellite data and the preprocessed data will get removed.

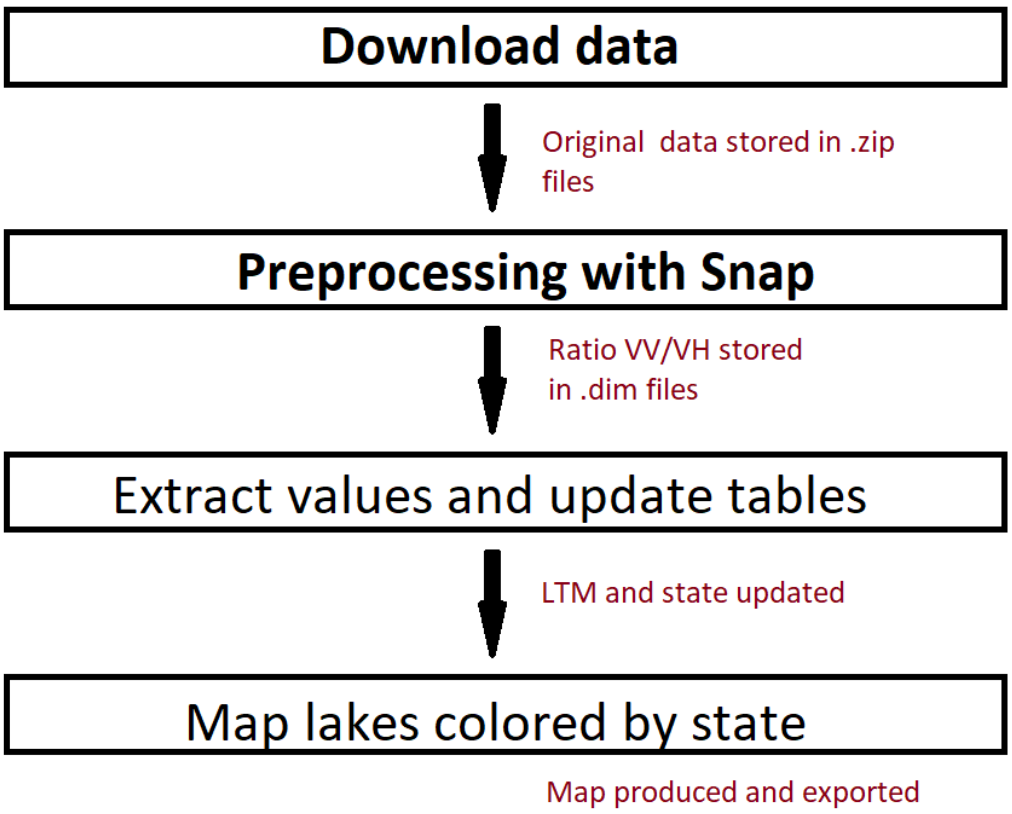

FIGURE 5.1: Visual overview of the different steps happening in the automated process.

5.5 Regarding big lakes

After consultation with Dr. Annett Frick, we decided to cut the polygons of the big lakes into smaller pieces, since the big lakes will not freeze as one body. This was done manually by me. Obviously this will lead to mistakes in classification and might even lead to many patches in different states next to each other.

6 Validation

The described method that worked best was validated at different lakes in winter 2018/2019 to assess the viability of the idea. Overall the fit seems to be quite good in the tested scenarios.

6.1 Germany

6.1.1 Nehmitzsee

Freezing data

Data of the IGB shows that the "Nehmitzsee" was freezing between 4th February 2019 and 24th February 2019, starting at 10% up to a complete ice cover. Unfortunately there is no satellite data to narrow the time span. There is no data before the 4th February 2019 thus I can not be 100% sure that the lake was not frozen at all before.

Time Series

The time series for "Nehmitzsee" for winter 2018/2019 is part of the conclusion (Figure 4.1). It shows that the time series for "Nehmitzsee" has 3 peaks that are above 1.4 times the LTM. One of them is inside the freezing period. The state of the lake would be 1, according to table 4.1, in late December, approximately 2 weeks before the first data point suggests that the lake is partly frozen. The peak in the middle of the freezing period could represent the freeze-up of the whole lake.

6.1.2 Großer Stechlinsee

Freezing data

"Großer Stechlinsee" was not completely frozen, starting at 1% on 29th January 2019 and ending the main freezing period at 50% on 15th February. Again, there is no satellite data to get a more precise time.

Time Series

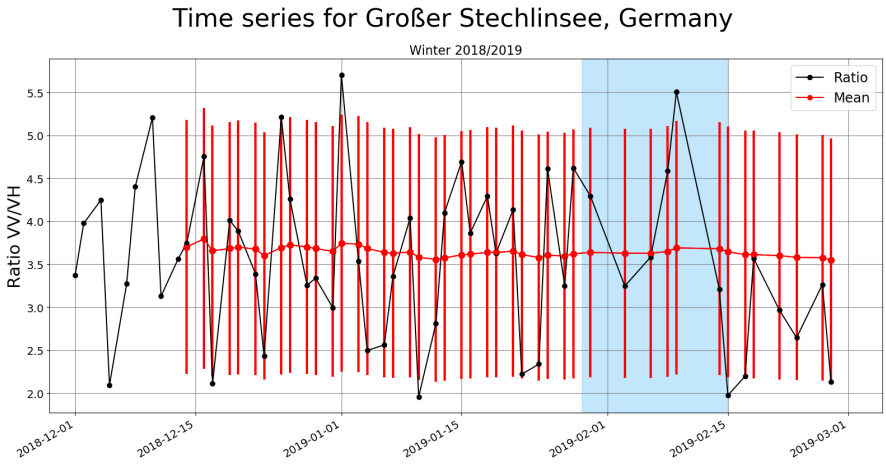

FIGURE 6.1: Validation of the described process on the time series of the ratio VV/VH on "Großer Stechlinsee" in winter 2018/2019. The red dots are showing the LTM, the vertical red lines are showing ±40% deviations of the LTM. The blue box is showing the freezing period.

The time series for "Großer Stechlinsee" shows a similar pattern as the time series for "Nehmitzsee": There is a peak above the threshold value during the freezing period, but unfortunately another peak before the freezing period. Additionally, the lake is not freezing completely, thus the result might be biased since a big part of the lake is not freezing throughout the winter.

6.2 Finland

6.2.1 Lake Kyyjärvi

Freezing data

Sentinel-2 data of 23rd October 2019 is not showing signs of ice on the lake. On 30th October the lake is frozen, small cracks are showing. This is consistent with temperature data: During the described period the temperature changes from up to 10° C to negative temperature values.

Time Series

The time series for "Lake Kyyjärvi" is already shown in figure 4.2. Throughout the freezing period a single, big peak above the LTM is visible. As stated in the conclusion the method is working perfectly fine for this lake.

6.2.2 Peuralampi

Freezing data

Sentinel-2 data of the 21st October 2019 is not showing signs of ice on the lake. On 5th November the lake is frozen, the surface is definitely covered by ice. Since the lake is close to "Lake Kyyjärvi" the temperature data is supporting this period again.

Time Series

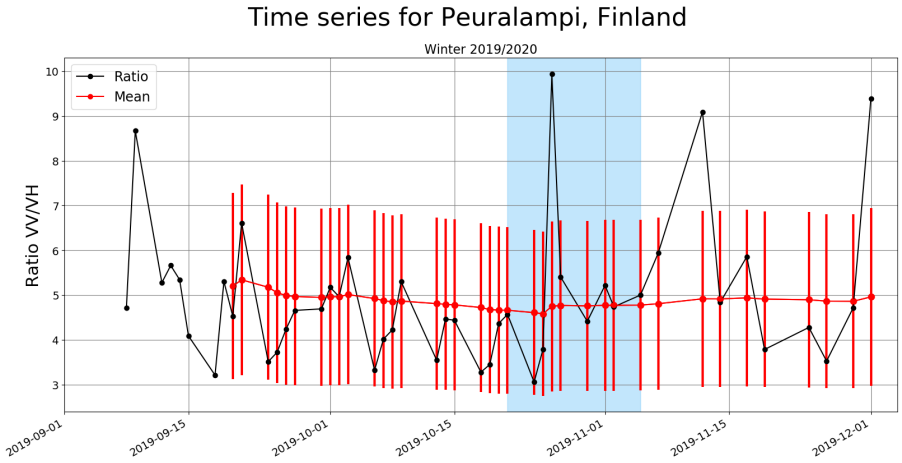

FIGURE 6.2: Validation of the described process on the time series of the ratio VV/VH on "Peuralampi" in winter 2019/2020. The red dots are showing the LTM, the vertical red lines are showing ±40% deviations from the LTM mean. The blue box is showing the freezing periods.

"Peuralampi" is showing the same pattern as "Lake Kyyjärvi": A single high value in the freezing period. This is supporting the used method even for a small lake in Finland.

6.3 Norway

The six lakes that are investigated here are pretty close to each other, see figure 2.2. Thus the temperature data for all lakes is the same and will not be mentioned in the different sections for every single lake. In late October 2019 the temperature decreases and varies around 0° C throughout November. The freezing period was determined using Sentinel-2 data on the Sentinel-hub Playground [41].

6.3.1 Birtevatn

Freezing data

"Birtevatn" is not frozen at all on 2019-11-05. On 17th November small bays were covered by ice. The whole lake surface was frozen on 30th November 2019.

Time Series

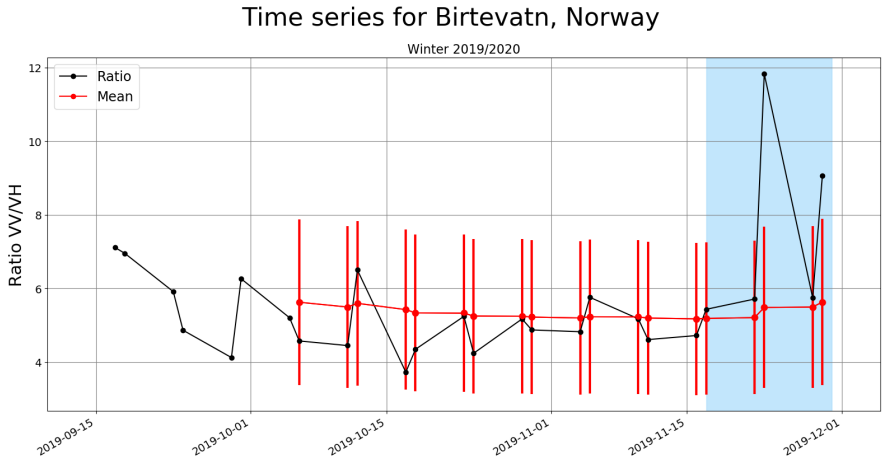

FIGURE 6.3: Validation of the described process on the time series of the ratio VV/VH on "Birtevatn" in winter 2019/2020. The red dots are showing the LTM, the vertical red lines are showing ±40% deviations from the LTM mean. The blue box is showing the freezing periods.

The time series for "Birtevatn" shows - like the lakes in Finland - one high peak in the freezing period. The end of the time series shows an increase again, but there is no sign of a possible miss-classification as ice before the freezing period starts.

6.3.2 Grøssæ

Freezing data

"Grøssæ" shows the same pattern as "Birtevatn". Not frozen on 5th November, partly frozen on 17th November and completely frozen on 30th November 2019.

Time Series

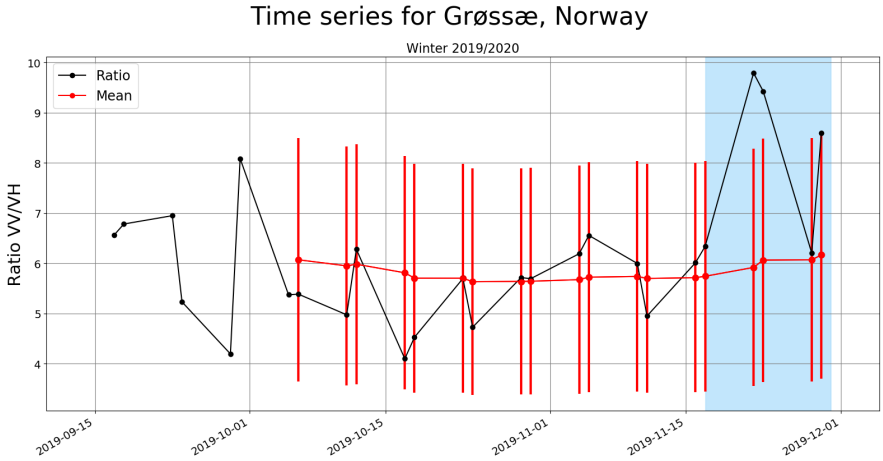

FIGURE 6.4: Validation of the described process on the time series of the ratio VV/VH on "Grøssæ" in winter 2019/2020. The red dots are showing the LTM, the vertical red lines are showing ±40% deviations from the LTM mean. The blue box is showing the freezing periods.

"Grøssæ" shows a good resulting time series for the used method. A single peak consisting of two consecutive values that are far above the long-term mean. There is no sign of miss-classification before the freezing period again.

6.3.3 Mjåvatn

Freezing data

"Mjåvatn" has the same freezing period characteristics as "Birtevatn" and "Grøssæ".

Time Series

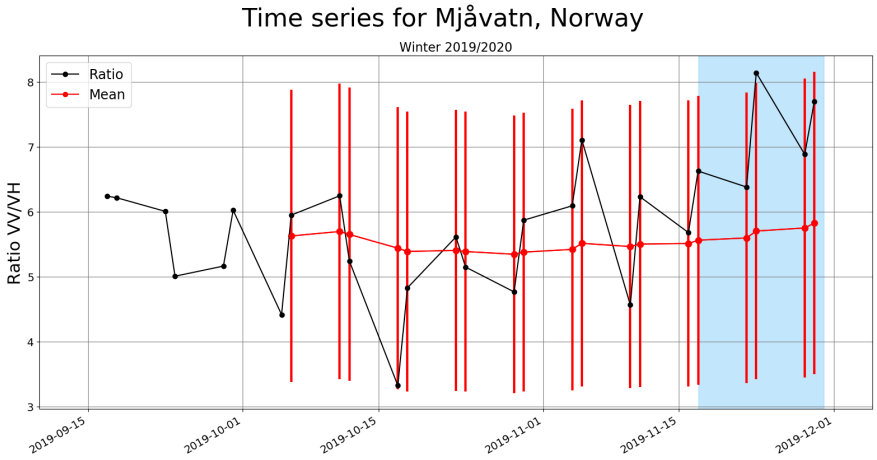

FIGURE 6.5: Validation of the described process on the time series of the ratio VV/VH on the "Mjåvatn" in winter 2019/2020. The red dots are showing the LTM, the vertical red lines are showing ±40% deviations from the LTM mean. The blue box is showing the freezing periods.

The resulting time series for "Mjåvatn" is not as perfect as the previously shown time series for the Norvegian lakes. But the lake would have been classified as frozen on the peak that is in the freezing period and little above the 40% threshold above the LTM.

6.3.4 Topsæ

Freezing data

"Topsæ" is not frozen on the 5th of November, frozen up to ca. 50% (the northern part of the lake) on the 10th of November 2019 and completely frozen on the 30th November 2019.

Time Series

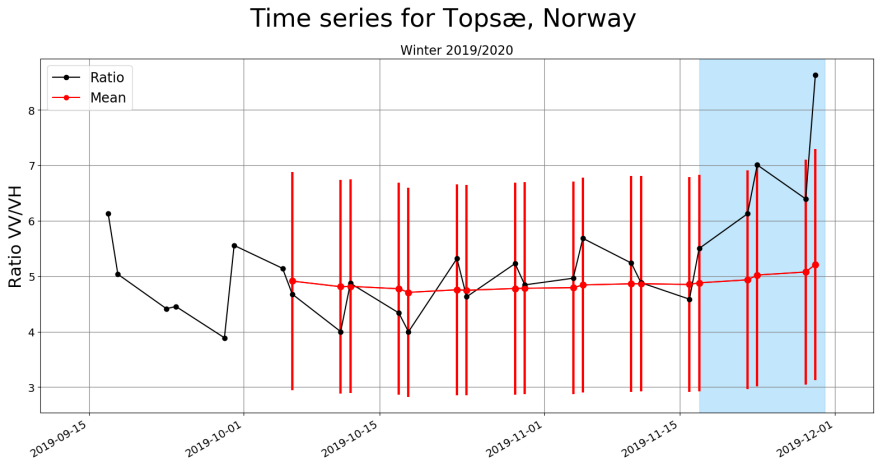

FIGURE 6.6: Validation of the described process on the time series of the ratio VV/VH on the "Topsæ" in winter 2019/2020. The red dots are showing the LTM, the vertical red lines are showing ±40% deviations from the LTM mean. The blue box is showing the freezing periods.

"Topsæ" is showing a good recognition of the complete freezing in the end of November. The higher values between 15th and 31st November could be due to the partial freezing early in November. There is no possible missclassification to early in the year.

6.3.5 Vånarosen and Kjetebuvatn

Freezing data

The small lakes are not frozen at all before 1st November 2019 and froze completely up to 10th November 2019.

Time Series

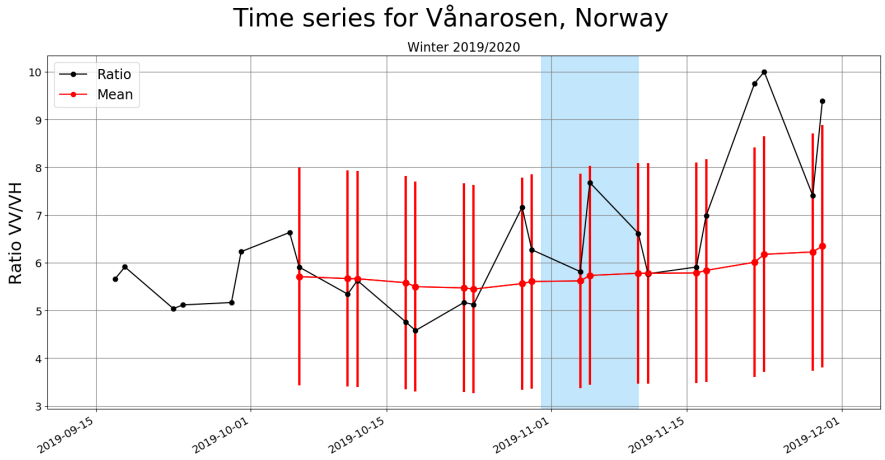

FIGURE 6.7: Validation of the described process on the time series of the ratio VV/VH on the "Vånarosen" in winter 2019/2020. The red dots are showing the LTM, the vertical red lines are showing ±40% deviations from the LTM mean. The blue box is showing the freezing periods.

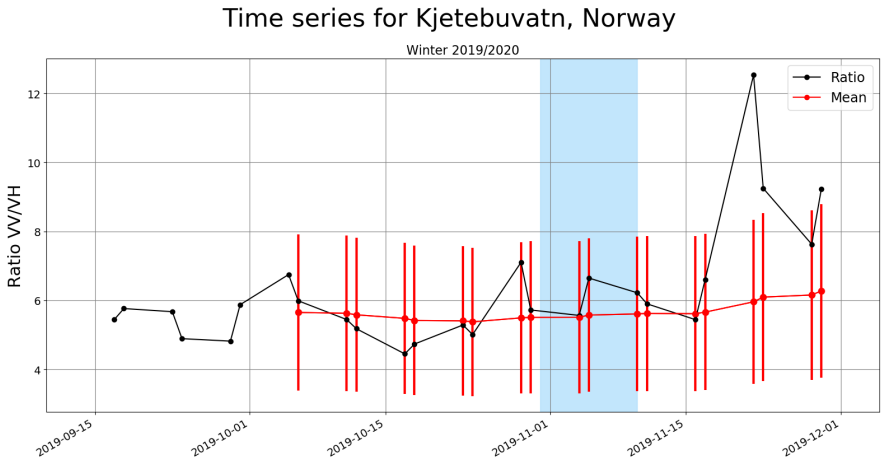

FIGURE 6.8: Validation of the described process on the time series of the ratio VV/VH on the "Kjetebuvatn" in winter 2019/2020. The red dots are showing the LTM, the vertical red lines are showing $\pm 40\%$ deviations from the LTM mean. The blue box is showing the freezing periods.

Even though both lakes show signs of freezing earlier than expected by looking at the time series, they are not classified as frozen before the 22nd November 2019. This could be due to a relatively fast freezing, that is not observed by Sentinel-1. Both lakes are very small, this should be taken into account since other very small lakes could be missclassified as well.

7 Discussion

Comparing my work to the state of the art is not that easy, because most people [4, 9, 25] are more interested in the thawing period or thawing onset instead of the freeze up.
The influences described in the introduction showed throughout this work: the incidence angle, the different types of ice, the not really represented thin ice, the bad recognition of thin ice on Sentinel-2 data (figure 2.1) and the different time spans of freezing made it really hard to get a working result.
Different time series showed that no real usage was possible besides the ratio of VV and VH. This is mainly because the effects of incidence angle should be similar on both waves which is a limiting factor for the other values [10, 15, 16]. The absolute value of the ratio is still not usable since it is varying a lot over the freezing period [2, 24]. A decrease of the ratio after the freeze-up of the lakes due to snow coverage as seen in some of the time series was also visible in previous work. This is due to an increase of volume scattering leading to an increase of the VH values compared to the surface scattering that is mostly influencing VV values [16].

At the time of ice formation the total backscatter should be low, since the radar pulse is reflected away from the antenna [5, 61], but this does not affect the ratio that much since both waves are reflected away. Floating ice or strong winds lead to higher backscatter values since the rough surface is scattering the incoming wave back to the antenna [5, 11]. Floating ice was detected using co-polarized data before, but is not as good as cross-polarised data when the lake surface is disturbed by wind [11]. Once thin ice is building the backscatter is not changing that much [23], but as soon as it grows, the backscatter increases due to the integration of air bubbles inside the ice [6, 18, 21, 23]. Deep lakes or deep parts of lakes seem to change slower since less air bubbles are produced, mostly because of less bioactivity [6, 23]. While ice is growing the backscatter is increasing until it either saturates the sensor (probably more relevant for older SAR systems) [6], or the ice thickens enough to decrease the influence of new air bubbles added to the pack [6, 18, 23]. This would be useful if one was interested in ice thickness measurements after the initial freeze-up of the lake. For this work only the initial change of the ratio due to different responses of VV and VH values to ice formation is relevant.
The unsupervised classification worked well on big lakes [24, 25], which is also visible in my results. I got a good fit for most of the big lakes I tested, but it did not result in any usable results for smaller lakes. Those were not investigated in the previous work using the classification algorithm. The reason for this could be the limited information stored in the Sentinel-1 data, where only two values are present, the smaller amount of actual data points in the

lakes and the relatively big spread of the values across the same classes. I think the difficulty here is stems from the small amount of information paired with the high variance of the values within each class. Additionally, the huge amount of different ice states [23] make it hard to use an easy classification with only three classes. The coexistence of the several classes will also negatively influence the recognition of ice. Previous work investigated only one point in time and not a time series [25]. They did not have to take changing ice dynamics into account. This made their idea usable for the specific case, but it is not a good way for automatic ice detection over time. The threshold values which are used in the existing literature [5, 10] and by the Canadian Ice Secive (CIS) for the definition of ice is adjusted manually for every scene of radar data. The usage of this approach was also good, as visible in figure 3.10 where the thresholds were defined. This made the initial attempt kind of redundant, but I still wanted to try it, since it would have been one of the easiest and fastest ways imaginable.

Looking at the SLC data and their usage in InSAR applications it is really hard to definitely tell whether a lake is just freezing or if it was already frozen and the ice cover changed in one of many ways, maybe air bubbles formed, snow covered the ice, whether there was already snow cover that got wet/dry... The same impressions were gained in literature [9]: the coherence value can be used to detect a changing environment, but not to exactly map ice. For my work that meant that I could see changes in coherence that fit quite well to the ice extent (see figure 3.20) but there were also some values that would indicate ice which was not present on the same image.
This is one of the difficulties on coherence measurements for change detection on lakes: If a pixel has a very high coherence value, there are different possibilities why this is the case. Firstly, the lake could be calm in both used scenes, there would be no real change and thus a high coherence is occurring. Secondly, there is flat ice, thus it is resulting in a high coherence because on both scenes the reflection would be more or less the same, especially if the ice did not change at all (Not getting thicker, no new snow, no change in the existing snow on the ice ...). Thirdly, there could be calm water and thin flat ice, which again, would reflect pretty much the same signals leading to high coherence. The most important factor are waves. Waves make the signal unpredictable. If the coincidence leads to a comparable wave pattern and to comparable reflection phases, the coherence will be high. But dependent on the current wave structure on the lake the coherence could be high if compared to any of the states. Even ice cracks can have a high coherence in comparison to waves if the random reflection leads to this. The viability of the described *topophase* band produced is not given as well. Even though there is an increase when the lake is freezing (see figure 3.19) the values decrease again just to increase again. It seems like the different states of the ice cover have an impact on the data, which is likely due to the changing height when e.g. snow is accumulating or ice thickens. The high coherence on the shoreline pixels in some results could be due to freezing of the whole water column which would result in a reflection of the ground. This would lead to the same signal since a significant change in the basement of the lake is not

expected [26]. The viability of coherence or interferograms is generally expected to be higher for largely unchanged areas [22], so dynamic lakes, that can be influenced by wind, are not the best study site, even though there was some work done using InSAR before to detect ice on water [6, 9, 21]. Open water and bubble-free ice share the property of low backscatter and therefore low coherence [26, 62]. Furthermore, flooded ice, cracked ice and ice covers with changing ice dynamics show low coherence values for Sentinel-1A imagery [26]. The coherence can not be used to perfectly map the ice extent. If only the coherence over time is watched there is a high possibility that the different ice types get mixed up and wavy water and ice are missclassified as the same.
Due to the different interactions of VV and VH with fresh ice it should be possible to detect the ice by using the X-polarised interferogram. They should represent different surface conditions but unfortunately they do not. All lakes show an increase in both the *topophase* and the *coherence* bands when there is no significant change in lake ice cover, as well as increases when ice is occurring first. But there is not a single lake sampled here that is showing a significant pattern only when ice is present. This is likely caused by a multitude of effects. The basic assumption that ice and water interact differently with the signal is probably not wrong and thus there should be differences in the interferograms. But there are again many other factors that lead to different results: Different ice types and different snow covers lead to different forms of scattering when interacting with the incoming signal. Plus, it is possible that the total amount of backscatter varies a lot. Different penetration depth of VV and VH waves were investigated looking at the *topophase* band, but it did not show a reliable result either, maybe because the relevant change in topography could also be due to little waves. The increase throughout the freezing period is unfortunately occurring later again (figure 3.24) and the quite big difference between the two scenes taken on the same date is alarming. For the other lakes the results are even worse.
Unfortunately I did not find any literature where this was done previously. It makes perfect sense for me to investigate this but maybe the usefulness is so low that no one really published anything about this.

Combining all these results only the used method of the lake wide mean of the ratio $\frac{VV}{VH}$ is resulting in a good detection of the freeze-up, even though it is not usable to detect the ice-extent for obvious reasons. Using the lake wide mean has both positive influences on the computational time, since its easier to calculate and compare one value for the whole lake and on the actual result since the problem of floating-ice on the lake and ground-fast ice on the shoreline is neglegible. Ground-fast ice has low backscatter values compared to the higher values of floating ice [17], hence the pixel-signal would not change much when the whole water column is freezing between two observations, the lake wide mean is changing. At least if there is only ground-fast freezing on the shorelines. "Birtevatn" and the other lakes in Norway show comparably less variability during the observed time series shown in the validation section. This **could** be due to less disturbance of the surface of the lakes since they are located in valleys.

Problems can occur if Lakes change their area compared to the one covered by the shapefile. If the lake is extended not all possible data is used and - more important - if it is shrinking mixed pixels will be taken into account.

Maybe there are other approaches that could be fine tuned or combined for getting the result I want to get, but none of the tested ones is working perfectly. The many different influences on the backscatter lead to uncertainty even on the accepted result. If one wants to increase the accuracy of the result it would be helpful to have more channels of satellite data [1]. Especially the HH and HV bands would be helpful since they showed earlier that they are more sensitive to ice [2, 4]. Another idea would be to use private satellite data with individually set temporal and spatial resolutions. Those could help finding even better results but were not usable for the current project. Coming back to the initially stated research questions, it is apparent that the ice extent is not mapped using the described method since whole lakes and not single lake pixels are looked at. The more important part, the detection of the freeze-up of the lake surface is working quite good for a big amount of lakes in a really short time. Thus I am confident to say, that this method is doing what it is intended to do, even though more research would clearly help to improve the results.

A Processing source code

A.1 SNAP - GRD preprocessing

```
<graph id="Graph">
  <version>1.0</version>
  <node id="Read">
    <operator>Read</operator>
    <sources/>
    <parameters class="com.bc.ceres.binding.dom.XppDomElement">
      <file>\${Filename}</file>
    </parameters>
  </node>
  <node id="ThermalNoiseRemoval">
    <operator>ThermalNoiseRemoval</operator>
    <sources>
      <sourceProduct refid="Read"/>
    </sources>
    <parameters class="com.bc.ceres.binding.dom.XppDomElement">
      <selectedPolarisations/>
      <removeThermalNoise>true</removeThermalNoise>
      <reIntroduceThermalNoise>false</reIntroduceThermalNoise>
    </parameters>
  </node>
  <node id="Apply-Orbit-File">
    <operator>Apply-Orbit-File</operator>
    <sources>
      <sourceProduct refid="ThermalNoiseRemoval"/>
    </sources>
    <parameters class="com.bc.ceres.binding.dom.XppDomElement">
      <orbitType>Sentinel Precise (Auto Download)</orbitType>
      <polyDegree>3</polyDegree>
      <continueOnFail>true</continueOnFail>
    </parameters>
  </node>
  <node id="Calibration">
    <operator>Calibration</operator>
    <sources>
      <sourceProduct refid="Apply-Orbit-File"/>
    </sources>
    <parameters class="com.bc.ceres.binding.dom.XppDomElement">
```

```
38        <sourceBands/>
39        <auxFile>Product Auxiliary File</auxFile>
40        <externalAuxFile/>
41        <outputImageInComplex>false</outputImageInComplex>
42        <outputImageScaleInDb>false</outputImageScaleInDb>
43        <createGammaBand>false</createGammaBand>
44        <createBetaBand>false</createBetaBand>
45        <selectedPolarisations/>
46        <outputSigmaBand>true</outputSigmaBand>
47        <outputGammaBand>false</outputGammaBand>
48        <outputBetaBand>false</outputBetaBand>
49      </parameters>
50    </node>
51    <node id="Speckle-Filter">
52      <operator>Speckle-Filter</operator>
53      <sources>
54        <sourceProduct refid="Calibration"/>
55      </sources>
56      <parameters class="com.bc.ceres.binding.dom.XppDomElement">
57        <sourceBands/>
58        <filter>Lee Sigma</filter>
59        <filterSizeX>3</filterSizeX>
60        <filterSizeY>3</filterSizeY>
61        <dampingFactor>2</dampingFactor>
62        <estimateENL>true</estimateENL>
63        <enl>1.0</enl>
64        <numLooksStr>1</numLooksStr>
65        <windowSize>5x5</windowSize>
66        <targetWindowSizeStr>3x3</targetWindowSizeStr>
67        <sigmaStr>0.9</sigmaStr>
68        <anSize>50</anSize>
69      </parameters>
70    </node>
71    <node id="Terrain-Correction">
72      <operator>Terrain-Correction</operator>
73      <sources>
74        <sourceProduct refid="Speckle-Filter"/>
75      </sources>
76      <parameters class="com.bc.ceres.binding.dom.XppDomElement">
77        <sourceBands/>
78        <demName>ASTER 1sec GDE</demName>
79        <demResamplingMethod>BILINEAR_INTERPOLATION</demResamplingMethod>
80        <imgResamplingMethod>BILINEAR_INTERPOLATION</imgResamplingMethod>
81        <pixelSpacingInMeter>10.0</pixelSpacingInMeter>
82        <pixelSpacingInDegree>8.983152841195215E-5</pixelSpacingInDegree>
83        <mapProjection>GEOGCS[&quot;WGS84(DD)&quot;, &#xd;
84    DATUM[&quot;WGS84&quot;, &#xd;
```

```
85      SPHEROID[&quot;WGS84&quot;, 6378137.0, 298.257223563]], &#xd;
86    PRIMEM[&quot;Greenwich&quot;, 0.0], &#xd;
87    UNIT[&quot;degree&quot;, 0.017453292519943295], &#xd;
88    AXIS[&quot;Geodetic longitude&quot;, EAST], &#xd;
89    AXIS[&quot;Geodetic latitude&quot;, NORTH]]</mapProjection>
90        <alignToStandardGrid>false</alignToStandardGrid>
91        <standardGridOriginX>0.0</standardGridOriginX>
92        <standardGridOriginY>0.0</standardGridOriginY>
93        <nodataValueAtSea>false</nodataValueAtSea>
94        <saveDEM>false</saveDEM>
95        <saveLatLon>false</saveLatLon>
96        <saveIncidenceAngleFromEllipsoid>false
97              </saveIncidenceAngleFromEllipsoid>
98        <saveLocalIncidenceAngle>false</saveLocalIncidenceAngle>
99        <saveProjectedLocalIncidenceAngle>false
100             </saveProjectedLocalIncidenceAngle>
101       <saveSelectedSourceBand>true</saveSelectedSourceBand>
102       <outputComplex>false</outputComplex>
103       <applyRadiometricNormalization>false</applyRadiometricNormalization>
104       <saveSigmaNought>false</saveSigmaNought>
105       <saveGammaNought>false</saveGammaNought>
106       <saveBetaNought>false</saveBetaNought>
107       <incidenceAngleForSigma0>Use projected local incidence angle from
108       DEM</incidenceAngleForSigma0>
109       <incidenceAngleForGamma0>Use projected local incidence angle from
110       DEM</incidenceAngleForGamma0>
111       <auxFile>Latest Auxiliary File</auxFile>
112       <externalAuxFile/>
113     </parameters>
114   </node>
115   <node id="BandMaths">
116     <operator>BandMaths</operator>
117     <sources>
118       <sourceProduct refid="Terrain-Correction"/>
119     </sources>
120     <parameters class="com.bc.ceres.binding.dom.XppDomElement">
121       <targetBands>
122         <targetBand>
123           <name>Ratio</name>
124           <type>float32</type>
125           <expression>Sigma0_VV / Sigma0_VH</expression>
126           <description/>
127           <unit/>
128           <noDataValue>0.0</noDataValue>
129         </targetBand>
130       </targetBands>
131       <variables/>
```

```
132     </parameters>
133   </node>
134 </graph>
```

A.2 SNAP - SLC processing

```
<graph id="Graph">
  <version>1.0</version>
  <node id="Read">
    <operator>Read</operator>
    <sources/>
    <parameters class="com.bc.ceres.binding.dom.XppDomElement"/>
  </node>
  <node id="Interferogram">
    <operator>Interferogram</operator>
    <sources>
      <sourceProduct refid="Read"/>
    </sources>
    <parameters class="com.bc.ceres.binding.dom.XppDomElement">
      <subtractFlatEarthPhase>true</subtractFlatEarthPhase>
      <srpPolynomialDegree>5</srpPolynomialDegree>
      <srpNumberPoints>501</srpNumberPoints>
      <orbitDegree>3</orbitDegree>
      <includeCoherence>true</includeCoherence>
      <cohWinAz>2</cohWinAz>
      <cohWinRg>10</cohWinRg>
      <squarePixel>true</squarePixel>
      <subtractTopographicPhase>false</subtractTopographicPhase>
      <demName>SRTM 3Sec</demName>
      <externalDEMFile/>
      <externalDEMNoDataValue>0.0</externalDEMNoDataValue>
      <externalDEMApplyEGM>true</externalDEMApplyEGM>
      <tileExtensionPercent>100</tileExtensionPercent>
      <outputElevation>false</outputElevation>
      <outputLatLon>false</outputLatLon>
    </parameters>
  </node>
  <node id="TOPSAR-Deburst">
    <operator>TOPSAR-Deburst</operator>
    <sources>
      <sourceProduct refid="Interferogram"/>
    </sources>
    <parameters class="com.bc.ceres.binding.dom.XppDomElement">
      <selectedPolarisations/>
    </parameters>
  </node>
  <node id="TOPSAR-Merge">
    <operator>TOPSAR-Merge</operator>
    <sources>
      <sourceProduct refid="TOPSAR-Deburst"/>
    </sources>
```

```
    <parameters class="com.bc.ceres.binding.dom.XppDomElement">
      <selectedPolarisations/>
    </parameters>
  </node>
  <node id="TopoPhaseRemoval">
    <operator>TopoPhaseRemoval</operator>
    <sources>
      <sourceProduct refid="TOPSAR-Merge"/>
    </sources>
    <parameters class="com.bc.ceres.binding.dom.XppDomElement">
      <orbitDegree>3</orbitDegree>
      <demName>ASTER 1sec GDEM</demName>
      <externalDEMFile/>
      <externalDEMNoDataValue>0.0</externalDEMNoDataValue>
      <tileExtensionPercent>100</tileExtensionPercent>
      <outputTopoPhaseBand>false</outputTopoPhaseBand>
      <outputElevationBand>false</outputElevationBand>
      <outputLatLonBands>false</outputLatLonBands>
    </parameters>
  </node>
  <node id="GoldsteinPhaseFiltering">
    <operator>GoldsteinPhaseFiltering</operator>
    <sources>
      <sourceProduct refid="TopoPhaseRemoval"/>
    </sources>
    <parameters class="com.bc.ceres.binding.dom.XppDomElement">
      <alpha>1.0</alpha>
      <FFTSizeString>64</FFTSizeString>
      <windowSizeString>3</windowSizeString>
      <useCoherenceMask>false</useCoherenceMask>
      <coherenceThreshold>0.2</coherenceThreshold>
    </parameters>
  </node>
  <node id="Terrain-Correction">
    <operator>Terrain-Correction</operator>
    <sources>
      <sourceProduct refid="GoldsteinPhaseFiltering"/>
    </sources>
    <parameters class="com.bc.ceres.binding.dom.XppDomElement">
      <sourceBands/>
      <demName>ASTER 1sec GDEM</demName>
      <externalDEMFile/>
      <externalDEMNoDataValue>0.0</externalDEMNoDataValue>
      <externalDEMApplyEGM>true</externalDEMApplyEGM>
      <demResamplingMethod>BILINEAR_INTERPOLATION</demResamplingMethod>
      <imgResamplingMethod>BILINEAR_INTERPOLATION</imgResamplingMethod>
      <pixelSpacingInMeter>0.0</pixelSpacingInMeter>
```

```
93      <pixelSpacingInDegree>0.0</pixelSpacingInDegree>
94      <mapProjection>GEOGCS[&quot;WGS84(DD)&quot;, &#xd;
95    DATUM[&quot;WGS84&quot;, &#xd;
96      SPHEROID[&quot;WGS84&quot;, 6378137.0, 298.257223563]], &#xd;
97    PRIMEM[&quot;Greenwich&quot;, 0.0], &#xd;
98    UNIT[&quot;degree&quot;, 0.017453292519943295], &#xd;
99    AXIS[&quot;Geodetic longitude&quot;, EAST], &#xd;
100   AXIS[&quot;Geodetic latitude&quot;, NORTH]]</mapProjection>
101     <alignToStandardGrid>false</alignToStandardGrid>
102     <standardGridOriginX>0.0</standardGridOriginX>
103     <standardGridOriginY>0.0</standardGridOriginY>
104     <nodataValueAtSea>false</nodataValueAtSea>
105     <saveDEM>false</saveDEM>
106     <saveLatLon>false</saveLatLon>
107     <saveIncidenceAngleFromEllipsoid>false
108     </saveIncidenceAngleFromEllipsoid>
109     <saveLocalIncidenceAngle>false</saveLocalIncidenceAngle>
110     <saveProjectedLocalIncidenceAngle>false
111     </saveProjectedLocalIncidenceAngle>
112     <saveSelectedSourceBand>true</saveSelectedSourceBand>
113     <outputComplex>false</outputComplex>
114     <applyRadiometricNormalization>false</applyRadiometricNormalization>
115     <saveSigmaNought>false</saveSigmaNought>
116     <saveGammaNought>false</saveGammaNought>
117     <saveBetaNought>false</saveBetaNought>
118     <incidenceAngleForSigma0>Use projected local incidence angle from
119     DEM</incidenceAngleForSigma0>
120     <incidenceAngleForGamma0>Use projected local incidence angle from
121     DEM</incidenceAngleForGamma0>
122     <auxFile>Latest Auxiliary File</auxFile>
123     <externalAuxFile/>
124   </parameters>
125
126 <applicationData id="Presentation">
127   <Description/>
128   <node id="Read">
129           <displayPosition x="15.0" y="20.0"/> </node>
130   <node id="Interferogram">
131           <displayPosition x="1.0" y="70.0"/></node>
132   <node id="TOPSAR-Deburst">
133           <displayPosition x="108.0" y="70.0"/></node>
134   <node id="TOPSAR-Merge">
135           <displayPosition x="112.0" y="20.0"/> </node>
136   <node id="TopoPhaseRemoval">
137           <displayPosition x="229.0" y="20.0"/></node>
138   <node id="GoldsteinPhaseFiltering">
139           <displayPosition x="217.0" y="70.0"/></node>
```

```
140     <node id="Terrain-Correction">
141             <displayPosition x="379.0" y="70.0"/></node>
142   </applicationData>
143 </graph>
```

A.3 GEE codes

A.3.1 Time series

```
////////////////////////////
// Predefine Functions
////////////////////////////

// Function to add Ratio
var addRatio = function(image) {
  return image.addBands(image.expression("VV/VH",
          {
          "VH": image.select(["VH"]),
          "VV": image.select(["VV"])
          }
          ).rename("Ratio"));
};

// This is a function by Guido Lemoine to apply a Refined-Lee-Speckle
// Filter on Sentinel-1 data.
//(https://code.earthengine.google.com/2ef38463ebaf5ae133a478f173fd0ab5,
// last checked on \nth{27} November 2019)

function RefinedLee(img) {
// img must be in natural units, i.e. not in dB!
// Set up 3x3 kernels
var weights3 = ee.List.repeat(ee.List.repeat(1,3),3);
var kernel3 = ee.Kernel.fixed(3,3, weights3, 1, 1, false);
var mean3 = img.reduceNeighborhood(ee.Reducer.mean(), kernel3);
var variance3 = img.reduceNeighborhood(ee.Reducer.variance(), kernel3);
// Use a sample of the 3x3 windows inside a 7x7 windows to determine
// gradients and directions
var sample_weights = ee.List([[0,0,0,0,0,0,0],
↪  [0,1,0,1,0,1,0],[0,0,0,0,0,0,0],[0,1,0,1,0,1,0],
↪  [0,0,0,0,0,0,0],[0,1,0,1,0,1,0],[0,0,0,0,0,0,0]]);
var sample_kernel = ee.Kernel.fixed(7,7, sample_weights, 3,3, false);
// Calculate mean and variance for the sampled windows and store as 9 bands
var sample_mean = mean3.neighborhoodToBands(sample_kernel);
var sample_var = variance3.neighborhoodToBands(sample_kernel);
// Determine the 4 gradients for the sampled windows
var gradients =
↪  sample_mean.select(1).subtract(sample_mean.select(7)).abs();
gradients = gradients.addBands(sample_mean.select(6).subtract(sample_mean
.select(2)).abs());
gradients = gradients.addBands(sample_mean.select(3).subtract(sample_mean
.select(5)).abs());
gradients = gradients.addBands(sample_mean.select(0).subtract(sample_mean
```

```
.select(8)).abs());

// And find the maximum gradient amongst gradient bands
var max_gradient = gradients.reduce(ee.Reducer.max());
// Create a mask for band pixels that are the maximum gradient
var gradmask = gradients.eq(max_gradient);

// duplicate gradmask bands: each gradient represents 2 directions
gradmask = gradmask.addBands(gradmask);
// Determine the 8 directions
var directions = sample_mean.select(1).subtract(sample_mean.select(4)).
↪  gt(sample_mean.select(4).subtract(sample_mean.select(7))).multiply(1);
directions = directions.addBands(sample_mean.select(6).
↪  subtract(sample_mean.select(4)).gt(sample_mean.select(4).
↪  subtract(sample_mean.select(2))).multiply(2));
directions = directions.addBands(sample_mean.select(3).
↪  subtract(sample_mean.select(4)).gt(sample_mean.select(4).
↪  subtract(sample_mean.select(5))).multiply(3));
directions = directions.addBands(sample_mean.select(0).
↪  subtract(sample_mean.select(4)).gt(sample_mean.select(4).
↪  subtract(sample_mean.select(8))).multiply(4));
// The next 4 are the not() of the previous 4
directions = directions.addBands(directions.select(0).not().multiply(5));
directions = directions.addBands(directions.select(1).not().multiply(6));
directions = directions.addBands(directions.select(2).not().multiply(7));
directions = directions.addBands(directions.select(3).not().multiply(8));
// Mask all values that are not 1-8
directions = directions.updateMask(gradmask);
// "collapse" the stack into a singe band image (due to masking, each
// pixel has just one value (1-8) in it"s directional band, and is
// otherwise masked)
directions = directions.reduce(ee.Reducer.sum());
// Generate stats
var sample_stats = sample_var.divide(sample_mean.multiply(sample_mean));
// Calculate localNoiseVariance
var sigmaV = sample_stats.toArray().arraySort().arraySlice(0,0,5).
↪  arrayReduce(ee.Reducer.mean(),
↪  [0]);
// Set up the 7*7 kernels for directional statistics
var rect_weights = ee.List.repeat(ee.List.repeat(0,7),3).cat(ee.List.
↪  repeat(ee.List.repeat(1,7),4));

// Set weights
var diag_weights = ee.List([[1,0,0,0,0,0,0], [1,1,0,0,0,0,0],
↪  [1,1,1,0,0,0,0], [1,1,1,1,0,0,0], [1,1,1,1,1,0,0], [1,1,1,1,1,1,0],
↪  [1,1,1,1,1,1,1]]);
```

```
var rect_kernel = ee.Kernel.fixed(7,7, rect_weights, 3, 3, false);
var diag_kernel = ee.Kernel.fixed(7,7, diag_weights, 3, 3, false);
// Create stacks for mean and variance using the original kernels.
// Mask with relevant direction.
var dir_mean = img.reduceNeighborhood(ee.Reducer.mean(),
  rect_kernel).updateMask(directions.eq(1));
var dir_var = img.reduceNeighborhood(ee.Reducer.variance(),
  rect_kernel).updateMask(directions.eq(1));
dir_mean = dir_mean.addBands(img.reduceNeighborhood(ee.Reducer.
  mean(),diag_kernel).updateMask(directions.eq(2)));
dir_var = dir_var.addBands(img.reduceNeighborhood(ee.Reducer.
  variance(),diag_kernel).updateMask(directions.eq(2)));

// and add the bands for rotated kernels
for (var i=1; i<4; i++) {
dir_mean = dir_mean.addBands(img.reduceNeighborhood(ee.Reducer.
  mean(),rect_kernel.rotate(i)).updateMask(directions.eq(2*i+1)));
dir_var = dir_var.addBands(img.reduceNeighborhood(ee.Reducer.
  variance(),rect_kernel.rotate(i)).updateMask(directions.eq(2*i+1)));
dir_mean = dir_mean.addBands(img.reduceNeighborhood(ee.Reducer.
  mean(),diag_kernel.rotate(i)).updateMask(directions.eq(2*i+2)));
dir_var = dir_var.addBands(img.reduceNeighborhood(ee.Reducer.
  variance(),diag_kernel.rotate(i)).updateMask(directions.eq(2*i+2)));
}
// "collapse" the stack into a single band image (due to masking, each
  pixel
// has just one value in it"s directional band, and is otherwise masked)
dir_mean = dir_mean.reduce(ee.Reducer.sum());
dir_var = dir_var.reduce(ee.Reducer.sum());
// A finally generate the filtered value
var varX = dir_var.subtract(dir_mean.multiply(dir_mean).multiply(sigmaV)).
  divide(sigmaV.add(1.0));
var b = varX.divide(dir_var);
var result = dir_mean.add(b.multiply(img.subtract(dir_mean)));
return(result.arrayFlatten([["sum"]]));
}

// Function to use the Lee-Filter on the two bands and add them to the
  image.
function UseLee(img){
  img = img.addBands(RefinedLee(img.select("VV")))
  img = img.addBands(RefinedLee(img.select("VH")))
            .rename("V","H","angle","VV","VH")
  return ee.Image(img.copyProperties(img)
        .copyProperties({source: img, properties: ["system:time_start"]}))
}
```

```
110
111 ////////////////////////////////////////////////
112 // VARIABLES
113 ////////////////////////////////////////////////
114 // Define start and end date of the time series
115 var start = "2017-12-01"
116 var end = "2018-03-01"
117 // aoi is the lake polygon minus 10m per side.
118 var aoi = geometry
119
120 // Create image collection covering the lake from start to end dates
121 var s1 = ee.ImageCollection("COPERNICUS/S1_GRD_FLOAT")
122         .filterBounds(aoi)
123         .filterDate(start, end)
124         .filterMetadata("transmitterReceiverPolarisation",
125         ."equals", ["VV", "VH"])
126         .filterMetadata("resolution_meters", "equals" , 10)
127         .sort("system:time_start")
128
129 // Use speckle filter and ratio-adding function on the image collection
130 s1 = s1.map(UseLee)
131 s1 = s1.map(addRatio)
132
133 // Create the time series with the mean ratio of the lake.
134 var tempTimeSeries = ui.Chart.image.seriesByRegion(
135     s1, geometry, ee.Reducer.mean(), "Ratio", 50,
136         "system:time_start", "label")
137         .setChartType("ScatterChart")
138         .setOptions({
139           title: "Ratio",
140           vAxis: {title: "VV/VH"},
141           lineWidth: 1,
142           pointSize: 4,
143           series: {
144             0: {color: "FF0000"},
145 }});
146
147
148 // Print the time series
149 print(tempTimeSeries)
```

A.3.2 Thresholds

```
////////////////////////////
// Predefine Functions
////////////////////////////

// This is a function by Guido Lemoine to apply a Refined-Lee-Speckle
// Filter on Sentinel-1 data.
//(https://code.earthengine.google.com/2ef38463ebaf5ae133a478f173fd0ab5,
// last checked on \nth{27} November 2019)

function RefinedLee(img) {
// img must be in natural units, i.e. not in dB!
// Set up 3x3 kernels
var weights3 = ee.List.repeat(ee.List.repeat(1,3),3);
var kernel3 = ee.Kernel.fixed(3,3, weights3, 1, 1, false);
var mean3 = img.reduceNeighborhood(ee.Reducer.mean(), kernel3);
var variance3 = img.reduceNeighborhood(ee.Reducer.variance(), kernel3);
// Use a sample of the 3x3 windows inside a 7x7 windows to determine
// gradients and directions
var sample_weights = ee.List([[0,0,0,0,0,0,0],
↪  [0,1,0,1,0,1,0],[0,0,0,0,0,0,0], [0,1,0,1,0,1,0],
↪  [0,0,0,0,0,0,0],[0,1,0,1,0,1,0],[0,0,0,0,0,0,0]]);
var sample_kernel = ee.Kernel.fixed(7,7, sample_weights, 3,3, false);
// Calculate mean and variance for the sampled windows and store as 9 bands
var sample_mean = mean3.neighborhoodToBands(sample_kernel);
var sample_var = variance3.neighborhoodToBands(sample_kernel);
// Determine the 4 gradients for the sampled windows
var gradients =
↪  sample_mean.select(1).subtract(sample_mean.select(7)).abs();
gradients = gradients.addBands(sample_mean.select(6).subtract(sample_mean.
↪  select(2)).abs());
gradients = gradients.addBands(sample_mean.select(3).subtract(sample_mean.
↪  select(5)).abs());
gradients = gradients.addBands(sample_mean.select(0).subtract(sample_mean.
↪  select(8)).abs());

// And find the maximum gradient amongst gradient bands
var max_gradient = gradients.reduce(ee.Reducer.max());
// Create a mask for band pixels that are the maximum gradient
var gradmask = gradients.eq(max_gradient);

// duplicate gradmask bands: each gradient represents 2 directions
gradmask = gradmask.addBands(gradmask);
// Determine the 8 directions
var directions = sample_mean.select(1).subtract(sample_mean.select(4)).
↪  gt(sample_mean.select(4).subtract(sample_mean.select(7))).multiply(1);
```

```
directions = directions.addBands(sample_mean.select(6).subtract(sample_mean.select(4)).gt(sample_mean.select(4).subtract(sample_mean.select(2))).multiply(2));
directions = directions.addBands(sample_mean.select(3).subtract(sample_mean.select(4)).gt(sample_mean.select(4).subtract(sample_mean.select(5))).multiply(3));
directions = directions.addBands(sample_mean.select(0).subtract(sample_mean.select(4)).gt(sample_mean.select(4).subtract(sample_mean.select(8))).multiply(4));
// The next 4 are the not() of the previous 4
directions = directions.addBands(directions.select(0).not().multiply(5));
directions = directions.addBands(directions.select(1).not().multiply(6));
directions = directions.addBands(directions.select(2).not().multiply(7));
directions = directions.addBands(directions.select(3).not().multiply(8));
// Mask all values that are not 1-8
directions = directions.updateMask(gradmask);
// "collapse" the stack into a singe band image (due to masking, each
// pixel has just one value (1-8) in it"s directional band, and is
// otherwise masked)
directions = directions.reduce(ee.Reducer.sum());
// Generate stats
var sample_stats = sample_var.divide(sample_mean.multiply(sample_mean));
// Calculate localNoiseVariance
var sigmaV = sample_stats.toArray().arraySort().arraySlice(0,0,5).arrayReduce(ee.Reducer.mean(),
[0]);
// Set up the 7*7 kernels for directional statistics
var rect_weights = ee.List.repeat(ee.List.repeat(0,7),3).cat(ee.List.repeat(ee.List.repeat(1,7),4));

// Set weights
var diag_weights = ee.List([[1,0,0,0,0,0,0], [1,1,0,0,0,0,0],
[1,1,1,0,0,0,0],[1,1,1,1,0,0,0], [1,1,1,1,1,0,0], [1,1,1,1,1,1,0],
[1,1,1,1,1,1,1]]);
var rect_kernel = ee.Kernel.fixed(7,7, rect_weights, 3, 3, false);
var diag_kernel = ee.Kernel.fixed(7,7, diag_weights, 3, 3, false);
// Create stacks for mean and variance using the original kernels.
// Mask with relevant direction.
var dir_mean = img.reduceNeighborhood(ee.Reducer.mean(),
rect_kernel).updateMask(directions.eq(1));
var dir_var = img.reduceNeighborhood(ee.Reducer.variance(),
rect_kernel).updateMask(directions.eq(1));
dir_mean = dir_mean.addBands(img.reduceNeighborhood(ee.Reducer.mean(),diag_kernel).updateMask(directions.eq(2)));
dir_var = dir_var.addBands(img.reduceNeighborhood(ee.Reducer.variance(),diag_kernel).updateMask(directions.eq(2)));
```

```
70
71  // and add the bands for rotated kernels
72  for (var i=1; i<4; i++) {
73  dir_mean = dir_mean.addBands(img.reduceNeighborhood(ee.Reducer.
    ↪  mean(),rect_kernel.rotate(i)).updateMask(directions.eq(2*i+1)));
74  dir_var = dir_var.addBands(img.reduceNeighborhood(ee.Reducer.
    ↪  variance(),rect_kernel.rotate(i)).updateMask(directions.eq(2*i+1)));
75  dir_mean = dir_mean.addBands(img.reduceNeighborhood(ee.Reducer.
    ↪  mean(),diag_kernel.rotate(i)).updateMask(directions.eq(2*i+2)));
76  dir_var = dir_var.addBands(img.reduceNeighborhood(ee.Reducer.
    ↪  variance(),diag_kernel.rotate(i)).updateMask(directions.eq(2*i+2)));
77  }
78  // "collapse" the stack into a single band image (due to masking, each
    ↪  pixel
79  // has just one value in it"s directional band, and is otherwise masked)
80  dir_mean = dir_mean.reduce(ee.Reducer.sum());
81  dir_var = dir_var.reduce(ee.Reducer.sum());
82  // A finally generate the filtered value
83  var varX = dir_var.subtract(dir_mean.multiply(dir_mean).multiply(sigmaV))
84  .divide(sigmaV.add(1.0));
85  var b = varX.divide(dir_var);
86  var result = dir_mean.add(b.multiply(img.subtract(dir_mean)));
87  return(result.arrayFlatten([["sum"]]));
88  }
89
90  // Function to use the Lee-Filter on the two bands and
91  // add them to the image.
92  function UseLee(img){
93    img = img.addBands(RefinedLee(img.select("VV")))
94    img = img.addBands(RefinedLee(img.select("VH")))
95              .rename("V","H","angle","VV","VH")
96    return ee.Image(img.copyProperties(img)
97          .copyProperties({source: img, properties: ["system:time_start"]}))
98  }
99
100
101 // ---------------------------------------------------------------------
102 // Functions to add band containing normalized difference
103 // between VH and VV, the Ratio and the Product
104 // ---------------------------------------------------------------------
105 function addRatio(image) {
106   return image.addBands(image.expression("VV/VH",
107           {
108           "VH": image.select(["VH"]),
109           "VV": image.select(["VV"])
110           }
111           ).rename("Ratio"))}
```

```
112
113 function addProduct(image) {
114   return image.addBands(image.expression("VV*VH / 100",
115           {
116           "VH": image.select(["VH"]),
117           "VV": image.select(["VV"])
118           }
119           ).rename("Product"))}
120
121 function addDiff(image) {
122   return image.addBands(image.expression("100*(VV-VH)/(VV+VH)",
123           {
124           "VH": image.select(["VH"]),
125           "VV": image.select(["VV"])
126           }
127           ).rename("Diff"))}
128
129 // Get imageCollection that is covering the geometry which is
130 // a imported shapefile of the lake.
131 var S1 = ee.ImageCollection("COPERNICUS/S1_GRD_FLOAT")
132         .filterBounds(geometry)
133         .filterDate("2018-03-03", "2018-03-04")
134         .filterMetadata("transmitterReceiverPolarisation",
135             "equals", ["VV", "VH"])
136         .filterMetadata("resolution_meters", "equals" , 10)
137
138 // Mosaic image to cover whole lake area if necessary
139 S1 = S1.mosaic()
140
141 // Use speckle filter and add Ratio etc.
142 S1 = UseLee(S1)
143 S1 = addRatio(S1)
144 S1 = addDiff(S1)
145 S1 = addProduct(S1)
146 Map.addLayer(S1)
147
148
149 // Define the thresholds
150 var R = 6.6
151 var VV = 0.01
152 var VH = 0.0018
153
154 // Select only those pixels that are above/below the thresholds.
155 var S1_VV = (S1.select("VV").gt(VV))
156 var S1_VH = (S1.select("VH").lt(VH))
157 var S1_Ratio = (S1.select("Ratio").gt(R))
158
```

```
159 // Add layers to the map
160 Map.addLayer(S1_VV)
161 Map.addLayer(S1_VH)
162 Map.addLayer(S1_Ratio)
```

A.3.3 K-Means classification

```
var addRatio = function(image) {
  return image.addBands(image.expression("VV/VH",
          {
          "VH": image.select(["VH"]),
          "VV": image.select(["VV"])
          }
          ).rename("Ratio"));
};


// This is a function by Guido Lemoine to apply a Refined-Lee-Speckle
// Filter on Sentinel-1 data.
//(https://code.earthengine.google.com/2ef38463ebaf5ae133a478f173fd0ab5,
// last checked on \nth{27} November 2019)

function RefinedLee(img) {
// img must be in natural units, i.e. not in dB!
// Set up 3x3 kernels
var weights3 = ee.List.repeat(ee.List.repeat(1,3),3);
var kernel3 = ee.Kernel.fixed(3,3, weights3, 1, 1, false);
var mean3 = img.reduceNeighborhood(ee.Reducer.mean(), kernel3);
var variance3 = img.reduceNeighborhood(ee.Reducer.variance(), kernel3);
// Use a sample of the 3x3 windows inside a 7x7 windows to determine
// gradients and directions
var sample_weights = ee.List([[0,0,0,0,0,0,0],
 ↪  [0,1,0,1,0,1,0],[0,0,0,0,0,0,0], [0,1,0,1,0,1,0],
 ↪  [0,0,0,0,0,0,0],[0,1,0,1,0,1,0],[0,0,0,0,0,0,0]]);
var sample_kernel = ee.Kernel.fixed(7,7, sample_weights, 3,3, false);
// Calculate mean and variance for the sampled windows and store as 9 bands
var sample_mean = mean3.neighborhoodToBands(sample_kernel);
var sample_var = variance3.neighborhoodToBands(sample_kernel);
// Determine the 4 gradients for the sampled windows
var gradients =
 ↪  sample_mean.select(1).subtract(sample_mean.select(7)).abs();
gradients = gradients.addBands(sample_mean.select(6).subtract(sample_mean.
 ↪  select(2)).abs());
gradients = gradients.addBands(sample_mean.select(3).subtract(sample_mean.
 ↪  select(5)).abs());
gradients = gradients.addBands(sample_mean.select(0).subtract(sample_mean.
 ↪  select(8)).abs());

// And find the maximum gradient amongst gradient bands
var max_gradient = gradients.reduce(ee.Reducer.max());
// Create a mask for band pixels that are the maximum gradient
var gradmask = gradients.eq(max_gradient);
```

```
40
41 // duplicate gradmask bands: each gradient represents 2 directions
42 gradmask = gradmask.addBands(gradmask);
43 // Determine the 8 directions
44 var directions = sample_mean.select(1).subtract(sample_mean.select(4)).
   ↪ gt(sample_mean.select(4).subtract(sample_mean.select(7))).multiply(1);
45 directions = directions.addBands(sample_mean.select(6).
   ↪ subtract(sample_mean.select(4)).gt(sample_mean.select(4).
   ↪ subtract(sample_mean.select(2))).multiply(2));
46 directions = directions.addBands(sample_mean.select(3).
   ↪ subtract(sample_mean.select(4)).gt(sample_mean.select(4).
   ↪ subtract(sample_mean.select(5))).multiply(3));
47 directions = directions.addBands(sample_mean.select(0).
   ↪ subtract(sample_mean.select(4)).gt(sample_mean.select(4).
   ↪ subtract(sample_mean.select(8))).multiply(4));
48 // The next 4 are the not() of the previous 4
49 directions = directions.addBands(directions.select(0).not().multiply(5));
50 directions = directions.addBands(directions.select(1).not().multiply(6));
51 directions = directions.addBands(directions.select(2).not().multiply(7));
52 directions = directions.addBands(directions.select(3).not().multiply(8));
53 // Mask all values that are not 1-8
54 directions = directions.updateMask(gradmask);
55 // "collapse" the stack into a singe band image (due to masking, each
56 // pixel has just one value (1-8) in it"s directional band, and is
57 // otherwise masked)
58 directions = directions.reduce(ee.Reducer.sum());
59 // Generate stats
60 var sample_stats = sample_var.divide(sample_mean.multiply(sample_mean));
61 // Calculate localNoiseVariance
62 var sigmaV = sample_stats.toArray().arraySort().arraySlice(0,0,5).
   ↪ arrayReduce(ee.Reducer.mean(),
   ↪ [0]);
63 // Set up the 7*7 kernels for directional statistics
64 var rect_weights = ee.List.repeat(ee.List.repeat(0,7),3).cat(ee.List.
   ↪ repeat(ee.List.repeat(1,7),4));
65
66 // Set weights
67 var diag_weights = ee.List([[1,0,0,0,0,0,0], [1,1,0,0,0,0,0],
   ↪ [1,1,1,0,0,0,0],[1,1,1,1,0,0,0], [1,1,1,1,1,0,0], [1,1,1,1,1,1,0],
   ↪ [1,1,1,1,1,1,1]]);
68 var rect_kernel = ee.Kernel.fixed(7,7, rect_weights, 3, 3, false);
69 var diag_kernel = ee.Kernel.fixed(7,7, diag_weights, 3, 3, false);
70 // Create stacks for mean and variance using the original kernels.
71 // Mask with relevant direction.
72 var dir_mean = img.reduceNeighborhood(ee.Reducer.mean(),
   ↪ rect_kernel).updateMask(directions.eq(1));
```

```
var dir_var = img.reduceNeighborhood(ee.Reducer.variance(),
 ↪  rect_kernel).updateMask(directions.eq(1));
dir_mean = dir_mean.addBands(img.reduceNeighborhood(ee.Reducer.
 ↪  mean(),diag_kernel).updateMask(directions.eq(2)));
dir_var = dir_var.addBands(img.reduceNeighborhood(ee.Reducer.
 ↪  variance(),diag_kernel).updateMask(directions.eq(2)));

// and add the bands for rotated kernels
for (var i=1; i<4; i++) {
dir_mean = dir_mean.addBands(img.reduceNeighborhood(ee.Reducer.
 ↪  mean(),rect_kernel.rotate(i)).updateMask(directions.eq(2*i+1)));
dir_var = dir_var.addBands(img.reduceNeighborhood(ee.Reducer.
 ↪  variance(),rect_kernel.rotate(i)).updateMask(directions.eq(2*i+1)));
dir_mean = dir_mean.addBands(img.reduceNeighborhood(ee.Reducer.
 ↪  mean(),diag_kernel.rotate(i)).updateMask(directions.eq(2*i+2)));
dir_var = dir_var.addBands(img.reduceNeighborhood(ee.Reducer.
 ↪  variance(),diag_kernel.rotate(i)).updateMask(directions.eq(2*i+2)));
}
// "collapse" the stack into a single band image (due to masking, each
 ↪  pixel
// has just one value in it"s directional band, and is otherwise masked)
dir_mean = dir_mean.reduce(ee.Reducer.sum());
dir_var = dir_var.reduce(ee.Reducer.sum());
// A finally generate the filtered value
var varX = dir_var.subtract(dir_mean.multiply(dir_mean).multiply(sigmaV))
.divide(sigmaV.add(1.0));
var b = varX.divide(dir_var);
var result = dir_mean.add(b.multiply(img.subtract(dir_mean)));
return(result.arrayFlatten([["sum"]]));
}


// Function to use the Lee-Filter on the two bands and add them to the
 ↪  image.
function UseLee(img){
  img = img.addBands(RefinedLee(img.select("VV")))
  img = img.addBands(RefinedLee(img.select("VH")))
            .rename("V","H","angle","VV","VH")
  return ee.Image(img.copyProperties(img)
        .copyProperties({source: img, properties: ["system:time_start"]}))
}



////////////////////////////////////////////////////////////////
/////// VARIABLES
```

```
////////////////////////////////////////////////////////////////////

// Set the date of the investigation
var date = "2018-02-16"
// Geometry has to be a shapefile imported in the GEE asset
var aoi = geometry


// Get Images in ImageCollection
var s1 = ee.ImageCollection("COPERNICUS/S1_GRD_FLOAT")
        .filterBounds(aoi)
        .filterDate(date, ee.Date(date).advance(1, "day"))
        // Filter to get images with VV and VH dual polarization.
        .filterMetadata("transmitterReceiverPolarisation",
                "equals", ["VV", "VH"])
        .filterMetadata("resolution_meters", "equals" , 10)
        .map(function(image){return image.clip(geometry)});

// See if there are scenes on the given date.
print(s1)

// Mosaic images to one image if there is more then 1 image needed
// to see a whole lake. (Only necessary to do for the "Müritz" in this
↪  work)
var ms1 = s1.mosaic()

// Add Ratio to the image
ms1 = addRatio(ms1)

////////////////////////////////////
//////    Clustering      //////////
////////////////////////////////////

// Define region
// The region of interest is the lake area.
var region = geometry

// Define bands
var bands = ("VV","VH")

// training region is the full image
var training = ms1.select(bands).sample({
  region: region,
  scale: 30,
  numPixels: 10e14});

// train cluster on image, the first number equals the number
```

```
156 // of produced clusters, the second number is defining the start
157 // using the k-means++ initialization.
158 var clusterer = ee.Clusterer.wekaKMeans(3,1).train(training);
159
160 // cluster the complete image
161 var result = ms1.select(bands).cluster(clusterer);
162
163
164 // Display the clusters.
165 Map.addLayer(result,
166     {min: 0, max: 2, palette:["Red","Green","Navy"]},
167     "clusters");
```

B Automatisation code

B.1 Whole process

```
sentinelsat -g Lake_Norway.json -s 20181101 -e 20181105 -d --producttype GRD

for %%X in (*.zip) do (gpt GRD_Preprocessing.xml -PFilename="%%X" -t "D:\%%~nX")

python CreatingTable.py
for /R G:\GRD\NEW %%R in (*.img) do python27 zonalstatistics.py %%R lakes.shp
python TimeSeries.py
python27 ReintroduceArcpy.py
```

The GRD_preprocessing.xml file is already inclueded in Appendix A, thus it will not be covered here.

B.2 CreatingTable.py

```
import numpy as np
import pandas as pd

# Set number of rows to the number of lakes in the FID
rows = 1
abc = np.linspace(0,rows-1,rows)

# Lake.csv
# Create dataframe with FID = index.
Lakes = pd.DataFrame(columns=['FID_'], index = abc)
Lakes['FID_'] = abc
# Save Dataframe
pd.DataFrame(Lakes).to_csv("G:\Lakes.csv", index = False)

# State.csv
# Create dataframe with FID = index and empty columns State and Process.
State = pd.DataFrame(columns=['FID_','State'], index = abc)
State['FID_'] = abc
# Save Dataframe
pd.DataFrame(State).to_csv("G:\State.csv", index = False)

# LTM.csv
# Create dataframe with only FID = index.
LTM = pd.DataFrame(columns=['FID_'], index = abc)
LTM['FID_'] = abc
# Save Dataframe
pd.DataFrame(LTM).to_csv("G:\LTM.csv", index = False)
```

B.3 Zonalstatistics.py

```
# Create function for tonalstuts
def zonalstats(raster,shape):
        # Set local variables
        inData = shape
        zoneField = "FID"
        PATH = os.getcwd()
        # Execute ZonalStatisticsAsTable
        Zon = ZonalStatisticsAsTable(inData, zoneField,
        ↪  raster,"Zonalstatistics.dbf", "NODATA", "ALL")
        # Convert Table to Numpy-Array including the Polygon_ID, the Count
        ↪  and the SUM
        Zonal_Stats =
        ↪  arcpy.da.TableToNumPyArray(Zon,('FID_','Count','Sum'))
        Outname_Mean=str(raster)[:-23]+str("Zst.txt")
        np.savetxt(Outname_Mean,Zonal_stats)
        np.savetxt(str(raster)+str("Zst.txt"),Zonal_stats)

if __name__ == "__main__":
        # Import modules
        import arcpy
        import numpy as np
        from arcpy import env
        from arcpy.sa import *
        import sys
        import os
        # Set variables
        env.workspace = r"G:\vali\Validation.gdb"
        os.environ['ESRI_SOFTWARE_CLASS']='Professional'
        arcpy.CheckOutExtension("Spatial")
        arcpy.env.overwriteOutput = True
        raster = sys.argv[1]
        shape = sys.argv[2]
        zonalstats(raster)
```

B.4 TimeSeries.py

```
import numpy as np
import pandas as pd
import glob
import os

Path = os.getcwd()
txts = glob.glob("*Zst.txt")

Dates = []
for x in txts:
     if x[0:6] not in Dates:
       Dates.append(x[0:6])

for Date in Dates:
    files = glob.glob(Date+str("*"))
    for i in range(len(files)):
        Stats =  np.loadtxt(files[i])
        if Stats.size == 0:
            print("HEy")
            continue
        if Stats.shape == (3,):
            if i == 0:
                Values = pd.DataFrame({'FID_':Stats[0],'Count':Stats[1],'S
                ↪ um':Stats[2]}, index =
                ↪ [0])
                for p in range(len(Values)):
                    if Values.iloc[p,2] == 0:
                        Values.iloc[p,1] = 0
            else:
                Count = 'Count'+str(i)
                Sum = 'Sum'+str(i)
                Values_new = pd.DataFrame({'FID_':Stats[0],Count:Stats[1],
                ↪ Sum:Stats[2]}, index =
                ↪ [0])
                for p in range(len(Values_new)):
                    if Values_new.iloc[p,2] == 0:
                        Values_new.iloc[p,1] = 0
                Values = pd.merge(Values, Values_new, on = 'FID_',
                ↪ how='outer')
        else:
            if i ==0:
                Values = pd.DataFrame({'FID_':Stats[:,0],'Count':Stats[:,1
                ↪ ],'Sum':Stats[:,2]})
                for p in range(len(Values)):
                    if Values.iloc[p,2] == 0:
```

```
                        Values.iloc[p,1] = 0
            else:
                Count = 'Count'+str(i)
                Sum = 'Sum'+str(i)
                Values_new = pd.DataFrame({'FID_':Stats[:,0],Count:Stats[:,1],Sum:Stats[:,2]})
                for p in range(len(Values_new)):
                    if Values_new.iloc[p,2] == 0:
                        Values_new.iloc[p,1] = 0
                Values = pd.merge(Values, Values_new, on = 'FID_', how='outer')

    # Sort the dataframe by name
    Values = Values.reindex(sorted(Values.columns), axis=1)

    # Drop FID and insert it as first column
    FID = Values['FID_']
    Values.drop(labels=['FID_'], axis=1,inplace = True)
    Values.insert(0, 'FID_',FID)

    # Create a new target dataframe for the final values
    Values_all = pd.DataFrame({'FID_': Values['FID_'], 'Count': np.nan, 'Sum': np.nan, 'Mean': np.nan}, dtype = float)

    for r in range(len(Values_all)):
        # nbr is the amount of text files for the currently investigated date
        nbr = len(files)
        # Sum up all counts for a lake
        Values_all.iloc[r,1] = Values.iloc[r,1:(1+nbr)].sum(axis=0)
        # Sum up all Sums for a lake
        Values_all.iloc[r,2] = Values.iloc[r,(1+nbr):(1+(2*nbr))].where(Values.iloc[r,(1+nbr):(1+(2*nbr))]>0).sum(axis=0)
        # Calculate the mean values by dividing the total sum by the total count
        Values_all.iloc[r,3] = Values_all.iloc[r,2]/Values_all.iloc[r,1]

    # Extract the Mean values for this day and the FID columns.
    DayMean = Values_all[['FID_','Mean']]
    # Rename the Mean column to add the Date to the column in the lakes.csv
    DayMean.columns = ['FID_',Date[0:2]+str("-")+Date[2:4]+str("-")+Date[4:7]]

    # Import the Lakes.csv containing the mean-values of the previous days.
    Lakes = pd.read_csv(Path+r"\\Lakes.csv")
    # Merge the new value to the Lakes-dataframe
    Lakes = pd.merge(Lakes, DayMean,on='FID_',how='outer')
```

```
80
81      # Export the Lakes.csv
82      pd.DataFrame(Lakes).to_csv(Path+r"\\Lakes.csv", index = False, na_rep=
        ↪ -9999)
83
84
85      LTM = pd.DataFrame()
86      State = pd.DataFrame()
87  # Current state:
88  # Lakes dataframe contains all Mean-values up to this day.
89      # if there are more then 10 datapoints, calculate the long-term mean
        ↪ (LTM)
90      # as a mean of all the values up to this day.
91      if Lakes.shape[1] > 10:
92          # Import the LTM_upd.csv of the previous day.
93          LTM = pd.read_csv(Path+r"\\LTM.csv")
94          # Add a new empty line for the day
95          LTM[Date] = np.nan
96          # Calculate a new LTM for each lake.
97          for p in range(Lakes.shape[0]):
98              LTM.iloc[p,-1] =
                ↪ Lakes.iloc[p,1:].where(Lakes.iloc[p,1:]>-1000).mean()
99
100         # Export the LTM-csv
101         pd.DataFrame(LTM).to_csv(Path+r"\\LTM.csv", index = False, na_rep=
            ↪ -9999)
102
103
104     # If the LTM exists import the State.csv
105     if len(LTM) >= 1:
106         State = pd.read_csv(Path+r"\\State.csv")
107
108     # If the current mean of the lake is above 1.4 times the LTM up to
        ↪ this date
109     # the state is changed to 1.
110         for h in range(Lakes.shape[0]):
111
112             if Lakes.iloc[h,-1] >= 1.4*LTM.iloc[h,-1]:
113                 State.iloc[h,1] = 1
114
115             else:
116                 State.iloc[h,1] = State.iloc[h,1]
117
118         pd.DataFrame(State).to_csv(Path+r"\\State.csv", index = False,
            ↪ na_rep= -9999)
```

B.5 ReintroduceArcpy.py

```
import numpy as np
import arcpy
import os
import datetime as dt
from arcpy import env

env.workspace = r"G:\GRD_TEst\Final\Test.gdb"
os.environ['ESRI_SOFTWARE_CLASS']='Professional'
arcpy.CheckOutExtension("Spatial")
arcpy.env.overwriteOutput = True
Date = "2018-02-055"

# Load Shapefile and State.csv updated on the day.
Shape = r'C:\Users\Felix\Desktop\Lakes.shp'
State = np.genfromtxt(r"C:\Users\Felix\Desktop\State.csv", delimiter=',',
↪  names=True, case_sensitive=True)

mxd = arcpy.mapping.MapDocument(r"C:\Users\Felix\Desktop\Mapping.mxd")
# Delete the State field of the shape.
arcpy.DeleteField_management(Shape,'State')
# Merge the updated state to the shapefile.
arcpy.da.ExtendTable(Shape,"FID", State,"FID_")
# Export the data as a map in PDF format.
Outname = r"C:/Users/Felix/Desktop/"+str(Date)+".pdf"
arcpy.mapping.ExportToPDF(mxd,Outname)
```

C IGB data

The data for ice coverage of the Müggelsee was provided by Rita Adrian, Leibniz-insitute for freshwater ecology and freshwater fishery. The data for the "Großer Stechlinsee" and the "Nehmitzsee" was provided by Peter Casper who is working at the IGB as well.

Date in [DD/MM/YYYY]	Ice cover in %
09/01/2018	25.0
14/01/2018	no measurement
23/01/2018	5.0
30/01/2018	No ice
06/02/2018	100.0
15/02/2018	100.0
22/02/2018	100.0
01/03/2018	100.0
08/03/2018	100.0
23/03/2018	80.0
28/03/2018	1.0
05/04/2018	No ice
04/01/2019	10.0
24/01/2019	100.0
30/01/2019	99.0
18/02/2019	18.0
26/02/2019	No ice

TABLE C.1: Ice coverage on the "Nehmitzsee" for winter 2017/2018 and 2018/2019 provided by the IGB.

Date in [DD/MM/YYYY]	**Ice cover in %**
09/01/2018	2.0
14/01/2018	2.0
23/01/2018	No ice
30/01/2018	No ice
06/02/2018	28.0
15/02/2018	41.0
22/02/2018	38.0
01/03/2018	42.0
08/03/2018	45.0
23/03/2018	17.0
28/03/2018	1.5
05/04/2018	No ice
28/11/2018	Little ice on shore
24/01/2019	12.0
29/01/2019	1.0
15/02/2019	50.0
18/02/2019	53.0
25/02/2019	30.0
26/02/2019	No ice

TABLE C.2: Ice coverage on the "Großer Stechlinsee" for winter 2017/2018 and 2018/2019 provided by the IGB.

Date in [DD/MM/YYYY]	**Ice cover in %**	**Ice thickness in cm**
06/02/18	30-50	no measurement
07/02/18	50	no measurement
08/02/18	80	1
09/02/18	**?**	2
10/02/18	**?**	2
11/02/18	0	no measurement
26/02/18	0	no measurement
27/02/18	<50	no measurement
28/02/18	100	2
01/03/18	100	5
02/03/18	100	9
04/03/18	100	13
05/03/18	100	13
06/03/18	100	12
08/03/18	100	9
13/03/18	80	3
14/03/18	0	no measurement
19/03/18	0	no measurement
20/03/18	**?**	1
21/03/18	**?**	2
22/03/18	**?**	2
23/03/18	**?**	1.5
24/03/18	>80	1
25/03/18	<10	no measurement

TABLE C.3: Ice coverage on the "Müggelsee" for winter 2017/2018 provided by the IGB. For some dates there are no values for the ice cover.

Bibliography

[1] Ya-Lun S. Tsai, Andreas Dietz, Natascha Oppelt, and Claudia Kuenzer. Remote Sensing of Snow Cover Using Spaceborne SAR: A Review. *Remote Sensing*, 2019. doi:10.3390/rs11121456.

[2] Claude R. Duguay, Monique Bernier, Yves Gauthier, and Alexei Kouraev. Remote sensing of lake and river ice. In *Tedesco, M. :Remote Sensing of the Cryosphere*, pages 273–306. John Wiley & Sons, Ltd, Chichester, UK, 2014. doi:10.1002/9781118368909.ch12.

[3] Martin O. Jeffries, Kim Morris, and Nickolai Kozlenko. Ice Characteristics and Processes, and Remote Sensing of Frozen Rivers and Lakes. In *Duguay, Claude R. and Pietroniro, Alain:Geophysical Monograph Series*, pages 63–90. American Geophysical Union, Washington, D. C., 2013. doi:10.1029/163GM05.

[4] Torsten Geldsetzer. Mapping and monitoring lake ice using SAR satellites, 2010. Accessed: 2019-12-11.

[5] Martin O. Jeffries, Kim Morris, Wilford F. Weeks, and Hiroyuki Wakabayashi. Structural and stratigraphie features and ERS 1 synthetic aperture radar backscatter characteristics of ice growing on shallow lakes in NW Alaska, winter 1991–1992. *Journal of Geophysical Research: Oceans*, 1994. doi:10.1029/94JC01479.

[6] Matt Nolan, Glen Liston, Peter Prokein, Julie Brigham-Grette, Virgil L. Sharpton, and Rachel Huntzinger. Analysis of lake ice dynamics and morphology on Lake El'gygytgyn, NE Siberia, using synthetic aperture radar (SAR) and Landsat. *Journal of Geophysical Research*, 2002. doi:10.1029/2001JD000934.

[7] Dono Giuli. Polarization Diversity in Radars, 1986. doi:10.1109/PROC.1986.13457.

[8] Martti Hallikainen. Microwave Dielectric Properties of Materials. In *Njoku, E.G.: Encyclopedia of Remote Sensing*, Encyclopedia of Earth Sciences Series. Springer, New York, NY, 2014. doi:10.1007/978-0-387-36699-9.

[9] Joost J. van der Sanden, Hugo Drouin, and Yong Bian. Repeat Pass InSAR Observations of River and Lake Ice Cover: A Preliminary Evaluation of Information Content. In *Proceedings of the 17th Workshop on River Ice, Edmonton, AB, Canada*, 2013.

[10] Cristina Surdu, Claude Duguay, Homa Pour, and Laura Brown. Ice Freeze-up and Break-up Detection of Shallow Lakes in Northern Alaska with Spaceborne SAR. *Remote Sensing*, 2015. doi:10.3390/rs70506133.

[11] Joost van der Sanden. *Mapping and Monitoring Lake Ice by Using SAR Satellites*. 2012.

[12] Helena Łos and Bogusław Pawlowski. The Use of Sentinel-1 Imagery in the Analysis of River Ice Phenomena on the Lower Vistula in the 2015-2016 Winter. 2017. doi:10.1109/SPS.2017.8053663.

[13] Helmut Rott and Thomas Nagler. Capabilities of ers-1sar for snow and glacier monitoring in alpine areas. 1993.

[14] Son V. Nghiem and George A. Leshkevich. Satellite SAR Remote Sensing of Great Lakes Ice Cover, Part 1. Ice Backscatter Signatures at C Band. *Journal of Great Lakes Research*, 33(4):722 – 735, 2007. doi:10.3394/0380-1330(2007)33[722:SSRSOG]2.0.CO;2.

[15] Mari-Ann N. Moen, Stian N. Anfinsen, Anthony P. Doulgeris, Angelika H.H. Renner, and Sebastian Gerland. Assessing polarimetric SAR sea-ice classifications using consecutive day images. *Annals of Glaciology*, 56(69):285–294, 2015. doi:10.3189/2015AoG69A802.

[16] Hans Lievens, Matthias Demuzere, Hans-Peter Marshall, Rolf H. Reichle, Ludovic Brucker, Isis Brangers, Patricia de Rosnay, Marie Dumont, Manuela Girotto, Walter W. Immerzeel, Tobias Jonas, Edward J. Kim, Inka Koch, Christoph Marty, Tuomo Saloranta, Johannes Schöber, and Gabrielle J. M. De Lannoy. Snow depth variability in the Northern Hemisphere mountains observed from space. *Nature Communications*, 10(1), 2019. doi:10.1038/s41467-019-12566-y.

[17] Georg Pointner, Annett Bartsch, Bruce C. Forbes, and Timo Kumpula. The role of lake size and local phenomena for monitoring ground-fast lake ice. *International Journal of Remote Sensing*, 40(3):832–858, 2019. doi:10.1080/01431161.2018.1519281.

[18] Reginald R. Muskett. L-Band InSAR Penetration Depth Experiment, North Slope Alaska. *Journal of Geoscience and Environment Protection*, 05(03):14–30, 2017. doi:10.4236/gep.2017.53002.

[19] Wilford F. Weeks, Paul Sellmann, and William J. Campbell. Interesting features of RADAR imagery of icecovered north slope lakes. 18(78):129–136, 1977. doi:10.1017/S0022143000021572.

[20] Wilford F. Weeks, Andrew G. Fountain, Mark L. Bryan, and Charles Elachi. Differences in radar return from ice-covered North Slope Lakes. *Journal of Geophysical Research*, 83(C8):4069–4073, 1978. doi:10.1029/JC083iC08p04069.

[21] Joost J. van der Sanden, Naomi H. Short, and Hugo Drouin. InSAR coherence for automated lake ice extent mapping: TanDEM-X bistatic and pursuit monostatic results. *International Journal of Applied Earth Observation and Geoinformation*, 73:605–615, December 2018. doi:10.1016/j.jag.2018.08.009.

[22] Dyre Dammann, Leif Eriksson, Andrew Mahoney, Christopher Stevens, Joost J. van der Sanden, Hajo Eicken, Franz Meyer, and Craig Tweedie. Mapping Arctic Bottomfast Sea Ice Using SAR Interferometry. *Remote Sensing*, 10(5):720, May 2018. doi:10.3390/rs10050720.

[23] Claude R. Duguay and Peter M. Lafleur. Determining depth and ice thickness of shallow sub-Arctic lakes using space-borne optical and SAR data. *International Journal of Remote Sensing*, 24(3):475–489, January 2003. doi:10.1080/01431160304992.

[24] Jennifer Sobiech and Wolfgang Dierking. Observing lake- and river-ice decay with SAR: Advantages and limitations of the unsupervised k -means classification approach. *Annals of Glaciology*, 54(62):65–72, 2013. doi:10.3189/2013AoG62A037.

[25] Heidi Hindberg and Eirik Malnes. Mapping of Lake Ice in Northern Europe Using Dual-Polarization Radarsat-2 Data. 2013.

[26] Zhaoqin Li and Karl-Erich Lindenschmidt. Coherence of Radarsat-2, Sentinel-1, and ALOS-1 PALSAR for monitoring spatiotemporal variations of river ice covers. *Canadian Journal of Remote Sensing*, 44(1):11–25, January 2018. doi:10.1080/07038992.2018.1419424.

[27] Helmut Rott, Thomas Nagler, and Rolf Scheiber. Snow mass retrieval by means of SAR interferometry. page 6, 2003. Paper presented at the 3rd FRINGE-Workshop, European Space Agency, Earth Observation, Frascati, Italy.

[28] Sentinel-1 SAR - Acquisition Modes. https://sentinel.esa.int/web/sentinel/user-guides/sentinel-1-sar/acquisition-modes. Accessed: 2019-12-11.

[29] Sentinel-1 SAR - Level-1. https://sentinel.esa.int/web/sentinel/user-guides/sentinel-1-sar/product-types-processing-levels/level-1. Accessed: 2019-12-11.

[30] Sentinel-1 - Instrument Payload. https://sentinel.esa.int/web/sentinel/missions/sentinel-1/instrument-payload. Accessed: 2019-12-11.

[31] Sentinel-1 SAR - Stripmap. https://sentinel.esa.int/web/sentinel/user-guides/sentinel-1-sar/acquisition-modes/stripmap. Accessed: 2019-12-11.

[32] F. De Zan and A. Monti Guarnieri. TOPSAR: Terrain Observation by Progressive Scans. *IEEE Transactions on Geoscience and Remote Sensing*, 44(9):2352–2360, September 2006. doi:10.1109/TGRS.2006.873853.

[33] Sentinel-1 SAR - Interferometric Wide Swath. https://sentinel.esa.int/web/sentinel/user-guides/sentinel-1-sar/acquisition-modes/interferometric-wide-swath. Accessed: 2019-12-11.

[34] Sentinel-1 SAR - Extra Wide Swath. https://sentinel.esa.int/web/sentinel/user-guides/sentinel-1-sar/acquisition-modes/extra-wide-swath. Accessed: 2019-12-11.

[35] Sentinel-1 SAR - Wave. https://sentinel.esa.int/web/sentinel/user-guides/sentinel-1-sar/acquisition-modes/wave. Accessed: 2019-12-11.

[36] Level-1 SLC Processing Algorithms - Sentinel-1 SAR Technical Guide. https://earth.esa.int/web/sentinel/level-1-slc-processing-algorithms. Accessed: 2019-12-11.

[37] IW GRD Resolutions - Sentinel-1 SAR Technical Guide. https://sentinel.esa.int/web/sentinel/technical-guides/sentinel-1-sar/products-algorithms/level-1-algorithms/ground-range-detected/iw, note = Accessed: 2019-12-11.

[38] SNAP | STEP. Accessed: 2019-12-11.

[39] FAQ – Google Earth Engine. https://earthengine.google.com/faq/. Accessed: 2019-12-11.

[40] ISCE Documentation. https://www.unavco.org/education/professional-development/short-courses/course-materials/insar/2014-insar-isce-course-materials/ISCE-manual.pdf, 2013. Accessed: 2019-12-11.

[41] Sentinel-hub Playground. https://apps.sentinel-hub.com/sentinel-playground/. Accessed: 2019-12-11.

[42] Steffen Thorsen. Timeanddate.com. https://www.timeanddate.com/. Accessed: 2019-12-11.

[43] Federico Filipponi. Sentinel-1 GRD Preprocessing Workflow. *Proceedings*, 2019. doi:10.3390/ECRS-3-06201.

[44] S1tbx - esa sentinel-1 toolbox - documentation. http://step.esa.int. Accessed: 2019-12-11.

[45] Jeong-Won Park, Anton Korosov, Mohamed Babiker, Stein Sandven, and Joong-Sun Won. Efficient thermal noise removal for sentinel-1 topsar cross-polarization channel. *IEEE Transactions on Geoscience and Remote Sensing*, 19:1–11, 12 2017. doi:10.1109/TGRS.2017.2765248.

[46] Guillaume Hajduch and Nuno Miranda. Masking "no-value" pixels on grd products generated by the sentinel-1 esa ipf, version 2.0. Technical report, 2018.

[47] Jong-Sen Lee, Igor Jurkevich, Piet Dewaele, Patrick Wambacq, and Andre Oosterlinck. Speckle filtering of synthetic aperture radar images: A review. *Remote Sensing Reviews*, 8(4), 1994. doi:10.1080/02757259409532206.

[48] Melanie Engram, Christopher D. Arp, Benjamin M. Jones, Olaniyi A. Ajadi, and Franz J. Meyer. Analyzing floating and bedfast lake ice regimes across Arctic Alaska using 25 years of space-borne SAR imagery. *Remote Sensing of Environment*, 209:660–676, 2018. doi:10.1016/j.rse.2018.02.022.

[49] GEE –Sentinel-1 Algorithms. https://developers.google.com/earth-engine/sentinel1. Accessed: 2019-12-11.

[50] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. volume 8, pages 1027–1035, 2007. doi:10.1145/1283383.1283494.

[51] James Macqueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[52] Iain H. Woodhouse. *Introduction to Microwave Remote Sensing*. CRC Press, 2005.

[53] Luis Veci. TOPS Interferometry Tutorial. 2015.

[54] Dan Weydahl. Analysis of ers tandem sar coherence from glaciers, valleys, and fjord ice on svalbard. *Geoscience and Remote Sensing, IEEE Transactions on*, 39:2029 – 2039, 2001. doi:10.1109/36.951093.

[55] Paul A. Rosen, Scott Hensley, Ian R. Joughin, Fuk K. Li, Søren N. Madsen, Ernesto Rodríguez, and Richard M. Goldstein. Synthetic aperture radar interferometry. *Proceedings of the IEEE*, 88:333 – 382, 2000. doi:10.1109/5.838084.

[56] Franz Meyer. Sentinel-1 InSAR Processing using the SNAP Toolbox. https://media.asf.alaska.edu/uploads/pdf/s-1tbx_insar_recipe_6-16-17_final.pdf, 2017. Tutorial based on coursework at the Alaska Satellite Facility, Accessed: 2019-12-11.

[57] Richard M. Goldstein and Charles L. Werner. Radar interferogram filtering for geophysical applications. geophysical research letters, 25, 4035-4038. *Geophysical Research Letters*, 25:4035–4038, 1998. doi:10.1029/1998GL900033.

[58] David Small and Adrian Schubert. Guide to ASAR Geocoding Issue 1.01. http://www.geo.uzh.ch/microsite/rsl-documents/research/publications/other-sci-communications/2008_RSL-ASAR-GC-AD-v101-0335607552/2008_RSL-ASAR-GC-AD-v101.pdf, 2018. Accessed: 2019-12-11.

[59] ISCE Manual. https://github.com/isce-framework/isce2-docs/blob/master/Notebooks/TOPS/Tops.ipynb. Accessed: 2019-12-11.

[60] Marcel Wille. Sentinelsat. https://github.com/sentinelsat/sentinelsat, 2016. Accessed: 2019-12-11.

[61] Tazio Strozzi, Urs Wegmuller, and Christian Matzler. Mapping wet snowcovers with SAR interferometry. *International Journal of Remote Sensing*, 20(12):2395–2403, 1999. doi:10.1080/014311699212083.

[62] Sang-Wan Kim, Shimon Wdowinski, Falk Amelung, Timothy H. Dixon, and Joong-Sun Won. Interferometric Coherence Analysis of the Everglades Wetlands, South Florida. *IEEE Transactions on Geoscience and Remote Sensing*, 51(12):5210–5224, December 2013. doi:10.1109/TGRS.2012.2231418.