Analyzing long-term changes in sand waves patterns in the Marsdiep Inlet.

Earth Sciences Master Thesis

Author:
Sarah Lieke Geessinck
Marine Sciences

Supervisors:
Dr. Maarten van der Vegt
Physical Geography, Utrecht University
Dr. Ir. Johan van der Molen
Royal Netherlands Institute for Sea Research

February 2023

Contents

List of Figures

1 Abstract

The Marsdiep inlet is a part of the Wadden Sea, located between Den Helder and Texel in the Netherlands. It is four kilometers wide, has a depth of maximally 27 meter and is dominated by semi-diurnal tidal currents. Sand waves are present on the sea bed of the Marsdiep, and they migrate under the influence of tidal currents. It is important to investigate these bed form features, to gain better understanding in their characteristics and migration. Two Acoustic Current Doppler Profilers (ADCPs) are mounted under a ferry, which crosses the inlet up to 32 times a day. The ADCPs measure the depth of the sea at 8 different locations per second when the ferry is crossing the inlet, this data was collected from 2010 to 2021. The data was corrected for tides and geometry, placed in grid cells and averaged over one month. The study area was divided into 3 parts, where subsequent sand waves were clearly visible. The sand wave characteristics were analyzed and tracked over time. The most northern area has symmetrical sand waves with an average height and wavelength of 2.16 m and 144 m, respectively. The sand waves did not migrate in the direction of the currents, probably because the area is deep, and currents are weaker at deeper levels. This was not observed by previous research. A seasonal cycle was found in the first four years of the data, with higher sand waves during summer with respect to winter months, which was also observed by previous research. The second area was located in the south of the inlet, but was hard to analyze because one large anomaly was present during the first 6 years of the analysis. The sand waves had a height and wavelength of 3.64 m and 187 m, respectively, and they migrated about 70 m per year, on average. The most southern part analysed in this research was the most stable during the 11 years. It had a sand wave height and length of 2.9 m and 114 m, respectively, and the sand waves migrated on average 80 m per year. The methods for determining the sand wave heights worked quite good, but the methods to find the other characteristics can be improved, for example by cross-correlation techniques.

2 Introduction

2.1 Barrier coast systems

Barrier coast systems consist of barrier islands separated by tidal inlets and backbarrier basins. They are found in for example North America and Europe, and make up approximately 15% of the coastline in the entire world. The barrier islands are formed, deceased and mostly maintained by tidal currents and wind waves. Features that are often found within tidal-inlet/backbarrier basin systems include ebb-tidal deltas, (bifurcating) channels, flood deltas and salt marshes (De Swart et al., 2009). Currents in tidal basins are caused by tidal forces, wind driven forces and density currents (Ridderinkhof, 1990).

2.2 The (Dutch) Wadden Sea

The Wadden Sea (1) is a shallow sea along the North Sea coast of The Netherlands, Germany and Denmark and is a typical tidal-inlet/backbarrier system. It has an unique ecosystem and is therefore an UNESCO world heritage site. The Dutch part of the Wadden Sea has an area of roughly 3000 square kilometers and is separated from the North Sea by barrier islands. The barrier islands are separated by tidal inlets, in which water and sediments are exchanged with the North Sea due to tidal cycles. The area is highly dynamic due to tides, fresh water flows and human interventions. The tides in the Wadden Sea are semi-diurnal and driven by tides incoming from the North Sea, moving from the south west to north east (Oost et al., 1995). The North Sea water is a mixture of saline ocean water coming from the English Channel and fresh river water from the Rhine (Ridderinkhof, 1990). The Wadden Sea is an estuarine environment due to fresh water flowing in from rivers, lakes and the residual flow current along the Dutch coast. Human interventions include dredging, land reclamation and constructing dikes, for example the closure of the Zuiderzee (De Jonge et al., 1993) (Eriksson et al., 2010).

2.3 The Marsdiep tidal basin

The Marsdiep inlet is located between Den Helder and Texel, at the southern west of the Dutch Wadden Sea, see figure 1. The inlet is about 4 km wide and maximal 27 m deep at the location where the ferry crosses (Buijsman et al., 2008a). In the southwestern Wadden Sea, the vertical salinity gradient is only slight and the transport of fresh water is therefore mainly the result of tidal mixing, while transport as a result of differences in density plays only a minor part (Postma, 1954). Only in regions close to larger water bodies like the Lake IJssel, density driven currents will be comparable to tidal currents (Ridderinkhof, 1990). The tidal prism is 990 * 10^6 m^3 and the tidal currents are dominated by semi-diurnal tidal components with surface velocities up to 2 m/s. The tidal asymmetry in the southern two thirds of the inlet is flood dominant, which results in a flood dominated sediment transport (Oost et al., 1995). The tides enter the Marsdiep channel from the south, with a tidal range of 1.4 m, and splits in two channels, the Texelstroom towards the north and the Malzwin channel towards the south (Buijsman et al., 2007). On the sea bed of the Marsdiep inlet, sand waves are present, as analyzed before by Buijsman et al. (2008a) from 1998 to 2005. These sand waves play a crucial role in the sediment transport, which will be explained in the next two sections.

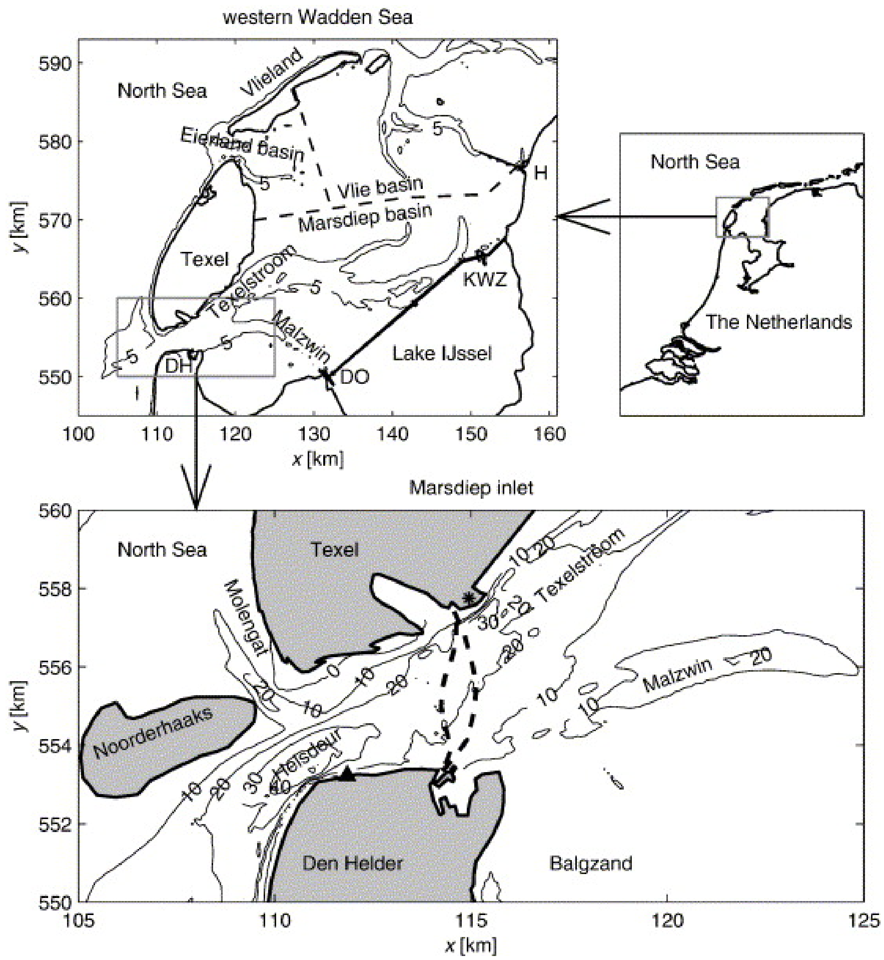


Figure 1: The first figure shows the location of the western Wadden Sea in the Netherlands and the lower figure indicates the Marsdiep. The thin dashed lines in the western Wadden Sea indicate the locations of the watersheds , the tidal channels are marked by isobaths of -5 m relative to mean sea level. The thick dashed lines in the Marsdiep indicate the location where the ferry crosses. The bathymetry is contoured in intervals of 10 meters. The lower figure shows where the Marsdiep inlet is located within the Wadden Sea and the upper right figure shows the Wadden Sea located in the Netherlands (Buijsman et al., 2007).

2.4 Importance of understanding bed forms

Bedform sizes and shapes are indicators of sedimentological and hydro-dynamic conditions and are of great importance for social and scientific reasons (Groeskamp et al., 2011)(Lefebvre et al., 2022). Understanding the mechanisms behind the transport of sand is important for management and budget of sediment in the region. Bed forms can be a hazard for navigation of offshore construction (Bellec et al., 2019). Bed form structures migrate, which constantly alters the water depth. This can be a problem for ships with deeper drafts that are not able to navigate in shallow areas. Offshore structures like oil platforms or wind farms, are dependent on stable foundations. The shifting of bed forms may lead to instability. The local sediment transport can also increase erosion of the offshore structures.

Relevant problems related to sea bed morphology are dredging of the navigation channel and coastal erosion (Barnard et al., 2006). The crests of sand banks or sand waves can be reduce the local water depths, which can cause the stranding of ships. The sand in the crests of the bed form features can be removed by dredging, but they can reform rapidly because the underlying mechanisms that drive sand wave formation are still exist (Veen, 1935) (Besio et al., 2008).

Another possible complication is the safety of pipelines, communication cables and offshore constructions, that can be exposed to the seawater by the movement of the sand. This is avoided by burying the cables deep enough or avoiding sand waves at all, but this is expensive, for example because longer lines are needed. With the understanding of sand wave evolution, this can be avoided (Besio et al., 2008).

There can be made a distinction based on wave length between four different types of bed forms; sandbanks with wavelengths of kilometers, sand waves with wavelengths of hundreds of meters, mega ripples with wavelengths of tens of meters and ripples with wavelengths of centimeters. These bed forms occur on coastal seas when currents are strong enough to move sediment. Dyer et al. (1999) The strength of the currents reflect the rate of sediment transport (Allen, 1980). The characters of bottom profiles are dependent on the materials, the size of sand grains, the velocity of the currents and the width and depth of the water (Veen, 1935).

2.5 Sand waves

Sand waves are flow-transverse bedforms with a typical wavelength of hundreds of meters that are generated by tides (Hulscher, 1996). Sand waves are observed in tide-dominated sandy shallow shelf seas (Borsje et al., 2014). Tidal waves are generated by gravitational forces of earth, moon and sun on the water and can have very long periods (44700 s for a semi-diurnal tide) and wavelengths of more than hundreds of meters. On the contrary, wind waves have periods of seconds and wavelengths up to 200 meter. The motion of fluids traveling over a surface may result in rhythmic surface forms (Veen, 1935) (Allen, 1980). Sand waves are not static, they migrate under residual currents or tidal asymmetry. Sand waves tend to develop more asymmetrical with increasing current asymmetry. Current asymmetry in the direction of the flood flow occurs when the maximum flood velocity exceeds the ebb maximum (Allen, 1980). Sand waves can grow due to a residual vertical circulation caused by tidal flow. The size of sand grain is also important, the larger the grains, the higher the sand waves. This is due to the fact that finer grains are more easily taken into suspension. (Damen et al., 2018)

Whether sediment is transported in the form of bed-load and suspended load, is dependent on the grain size of the bed materials and the current conditions. The bed-load transport is due to the effect of gravity, while in suspended load, the weight of the particle is supported by turbulence (Van Rijn, 1984). Sand wave migration is governed by bed load transport, and when the suspended load transport gets too large, thus when the grain size is small, sand waves are washed out (Buijsman et al., 2008b) (Borsje et al., 2014). Bed load transport is determined based on sand wave parameters, the migration rate, the wave length and the wave height (Buijsman et al., 2008b). The wavelength and sand wave height correlate very poorly with water depth according to Bøe et al. (2009), strong surface geostrophic currents and tidal currents have big influence on the sand wave formation.

Submarine sand waves can be divided into two main classes; symmetric (trochoidal) and asymmetric (progressive). Progressive waves have currents from one direction, in which the sand grains are pushed up the upcurrent slope and dropped by their own weight after having past the crest 3. In a trochoidal wave, the ebb falls in from one side and the flood from each other, resulting in a high wave with a sharp crest 2. When the flood and ebb are of unequal strength, the wave is asymmetrical trochoidal 4. Crests are generally oriented perpendicular to the flow (Hulscher, 1996). Rounded crests indicate that they are presently inactive, their presence may indicate older stronger oceanographic conditions when currents were stronger. Sharp crests in smaller sand waves indicate active but reduced sediment transport in the region (Bellec et al., 2019).

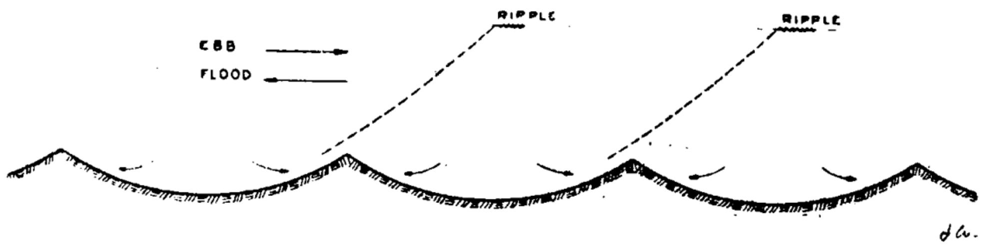


Figure 2: Trochoidal sand waves (Veen, 1935).

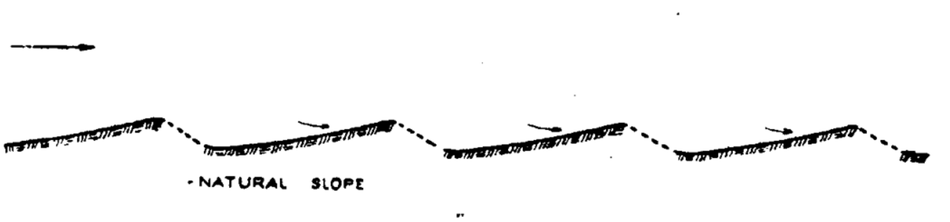


Figure 3: Progressive sand waves (Veen, 1935).

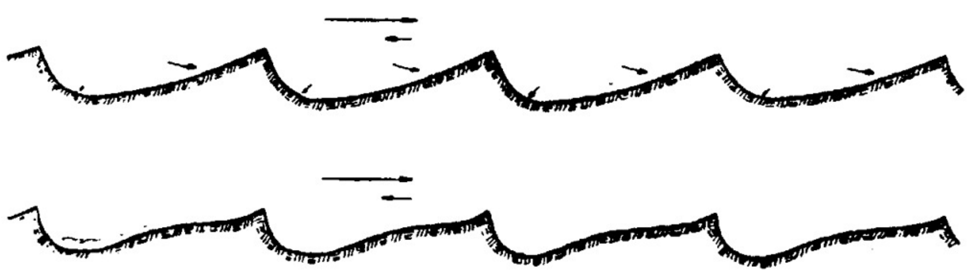


Figure 4: Asymmetrical-trochoidal sand waves (Veen, 1935).

2.6 Sand waves in the Marsdiep inlet

The sand waves in the Marsdiep inlet was observed before by Buijsman et al. (2008a). According to Buijsman et al. (2008a) the Marsdiep inlet can be divided into two areas, the southern area, which is dominated by progressive sand waves, and the northern area, which is dominated by asymmetrical-trochoidal sand waves. Table 1 shows the height, wavelength, crest orientation, migration direction and migration rate of the sandwaves in different areas, which can be found surrounded by the dashed lines in figure 5. Now, twelve more years of data is present, which results in the questions below.

Area	height (m)	length (m)	crest orientation	migration direction	migration rate (m y^{-1})
I	3.1	184	N to S	E in flood direction	52
II	3.1	184	N to S	E in flood direction	60-90
III	2.8	171	NW to SW	E to NE	31
IV	1.9	162	N to S	E to NE	47

Table 1: Sand wave characteristics analyzed by Buijsman et al. (2008a).

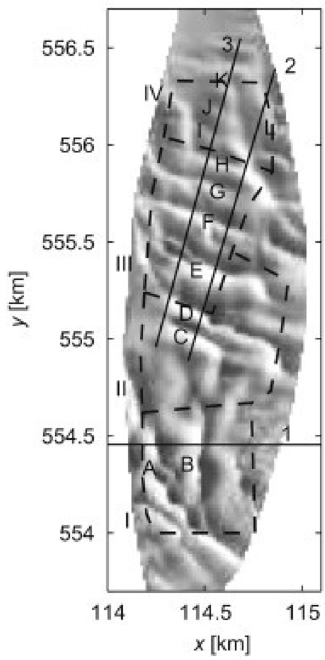


Figure 5: The variance around area-mean depth of the Marsdiep analyzed by Buijsman et al. (2008a). Lighter grey indicates positive height, darker grey indicates negative heigth. The different areas, I, II, III and IV, surrounded with a dashed line, represent areas I, II, III and IV in table 1. The units are in kilometers and in RD-coordinates.

Main Question:
How did the migration, height, length, asymmetry and orientation of sand waves in the Marsdiep inlet change?
Question 1:
What are the sand wave characteristics in the Marsdiep inlet?

Question 2:
How did the sand wave characteristics change over time?

3 Methods

3.1 Depth maps

3.1.1 ADCP data collection

Depth measurements of two ADCPs were taken between 2009 and 2022. A ferry of the company TESO (Texels eigen stoomboot onderneming) crosses this inlet up to 32 times a day. From 2005 until 2016, the Dokter Wagemaker was crossing the inlet and since 2016, the Texelstroom is used. Measurements were taken every second when the ferry was sailing between Texel to Den Helder and back to Texel. It sailed daily and twice per hour, from 6 am to 10 pm, without turning. The ferry started every day in Texel, and has an average speed of 6 ms^-1, with a slower speed at the start and the end, so a crossing takes about 15 minutes. The ferry was in maintenance for most of January, so less data is available for the first month of each year, and for some months no data. Before 2017, the dokter Wagemaker was in use, after 2017 till the present, the Texelstroom is used.

3.1.2 Acoustic Doppler Current Profiler

At a depth of 4.3 m below water surface, two acoustic doppler current profilers (ADCPs), with each four beams is mounted under the ferry. One on the Texel side and one on the Den Helder side. The ADCP at the Texel side is denoted by tx and the ADCP at the Den Helder side is denoted by hd. The ADCP sends four acoustic signals of a specific frequency every second in four directions, perpendicular to each other. The ADCPs can measure water depths, by measuring the return signal. It also measures the velocities, temperature, location and the errors. The accuracy of the ADCPs can be found in Buijsman et al. (2008a).

3.1.3 Corrections of the data

The depth measurements of the ADCPs were stored in netCDF files and loaded into python scripts. Depth measurements of the four individual beams were stored, as well as an averaged depth, which is the sum of the four depth measurements divided by four. The averaged depth is not corrected for the angle with the vertical and the angle relative to the north, as explained later in section 3.2. The averaged depth as well as the individual depth measurements from the ADCP needed to be corrected for the tides, which are caused by the attraction of the sun and the moon. The tide data was obtained from Rijkswaterstaat (Department of Waterways and Public Works in the Netherlands) and is saved in MET (Middle European Time) winter time. The ADCP data was stored in UTC (Coordinated Universal Time), so the tide data was corrected to UTC times by subtracting one hour. Rijkswaterstaat measured the the water level relatively to NAP (Normaal Amsterdams Peil) once every ten minutes at the location called "Den Helder Veerhaven". The water level measurements were interpolated to get a value for every second. The interpolated deviations of the NAP due to tides were subtracted from the ADCP depth measurements, to obtain bottom depth relative to NAP.

3.1.4 Depth map creation

The four individual depths measured by the ADCPs are corrected for the geometry and the orientation of the ADCPs and the orientation of the ferry relatively to the north, like explained in section 3.2. The locations of the depth measurements, obtained from the netCDF files, given in longitude and latitude, were converted to x and y, the distance in kilometers from the prime meridian and the equator respectively. The Marsdiep inlet is divided into a grid. The size of the grid cell has an influence on the number of data points in each grid cell. The great advantage of a smaller grid cell size is that the depth maps have higher resolutions than depth maps made with larger grid cells. However, the medians are less accurate because of a lower number of measurements, resulting in noisier pictures. Moreover, it takes longer to run and the depth maps contain more gaps (grid cells with no data). On the other side, when using larger grid cells, the risk is higher that smaller bedforms are not visible because of averaging out. Using smaller grid cells gives more detailed results. As can be seen in the figure, there are enough measurements using the four separate beams to fill the majority of the grid cells properly. A grid size of 5 by 5 meter is chosen. It has 684 cells in the x direction and 853 in the y direction, resulting in 583452 grid cells. Depth measurements from an entire month were placed into a grid cell based on their location. The time span of a month was chosen, because it is a balance between enough data that the grid cells are filled, but not too much that migrations are not visible anymore. The median of all the depth measurements in a grid cell was calculated. A depth map was made for every month, with every grid cell displaying their median depth value.

3.1.5 Bedform visualization

The variance around the area-mean depth was calculated in order to analyze the different bed forms. Figure 6 shows how the running mean is computed. The dark blue square represents a grid cell and the light blue square represents the grid cells over which the mean is computed. The window size must be chosen so that no bed forms are visible in the moving average. This can be accomplished by choosing a window size larger than at least twice an average sand wave length. According to Buijsman et al. (2008a), the length of a sand wave is about 180 meter, so a window size of 45 grid cells is chosen. The third figure in figure 7 shows the running mean for December 2021. No bed form elevation migrations are visible in the first and second panel, which implies that the chosen grid size is large enough. The blue and red line represent vertical transects over the running mean, and are plotted in the first en second figure for every month, from 2010 until 2021. The running mean was subsequently subtracted from the depth map, resulting in more visible bedforms, denoted by h'.

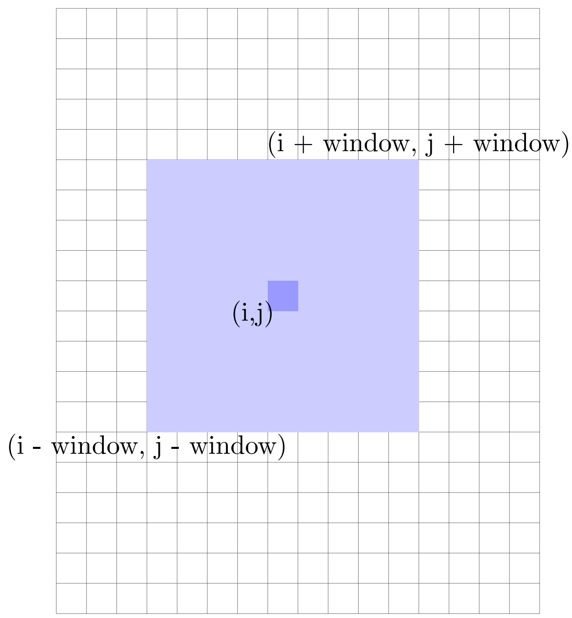


Figure 6: Computing the running mean.

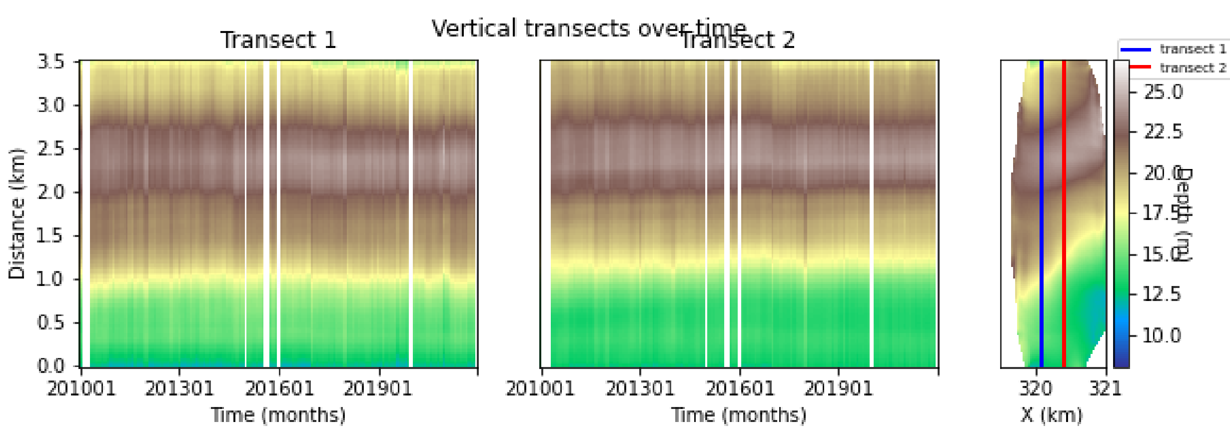


Figure 7: a) and b) the transects with the average depth displayed in c) over time. Green is deep, light brown is shallow. c) the locations at which the transects are taken.

3.1.6 Mask creation

Not every grid cell has a depth value. The number of grid cells that do have a value, is different for every month and is dependent on the sailing route of the ferry and the number of times the ferry has sailed. If a grid cell does have a value, it means that the ferry has sailed at this location at least once. Figure **??** shows the number of months that has a depth value for each grid cell. Different masks were made to find the balance between accuracy and sufficient amount of data to compare. It is ideal to include only grid cells that are passed every month. If other grid cells are included for analyzing, for example, the standard deviation, the analyzed area is not the same for every month. Unfortunately, the number of grid cells that is passed every month is very low, as can be seen in figure 6. Therefore, different masks are made. Figure 16, 17, 18 and 19 show mask100, mask95, mask90 and mask85. In mask100, all the grid cells are passed at least once every month. The grid cells in mask95, mask90 and are passed by the ferry for 95%, 90% and 85% of the months respectively. A lower percentage results in a higher number of passed grid cells, but a higher percentage results in more accurate results. The mask with a coverage of 90% was chosen and multiplied with the depth maps for every month for the analysis of the average sand wave height. Figure 13 shows the number of filled grid cells for every months after multiplying with mask90. Months that have less than seventy thousand filled grid cells are not included in further analysis.

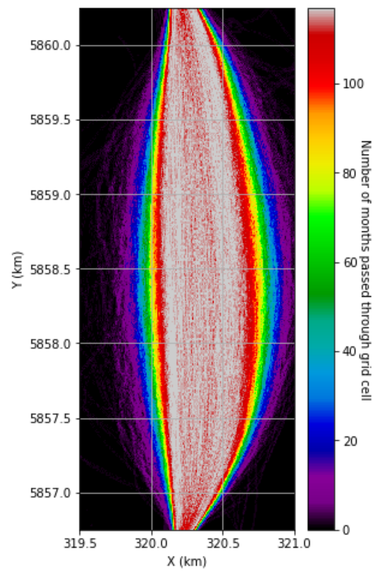


Figure 8: Number of months in which the ferry has crossed the grid cell at least once.

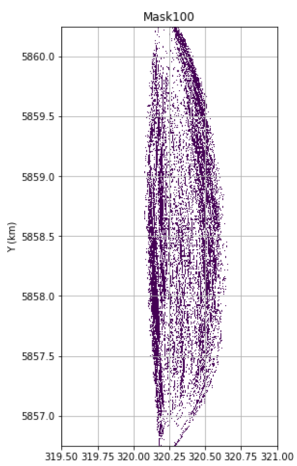


Figure 9: Mask100

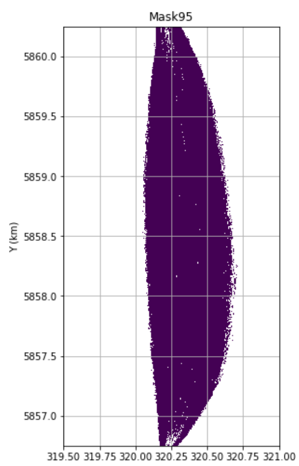


Figure 10: Mask95

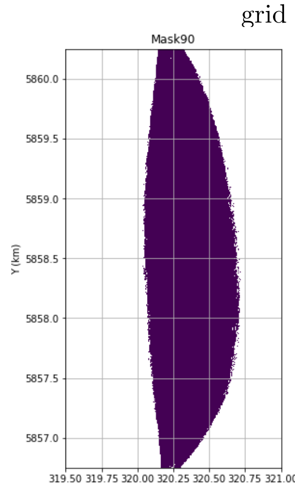


Figure 11: Mask90

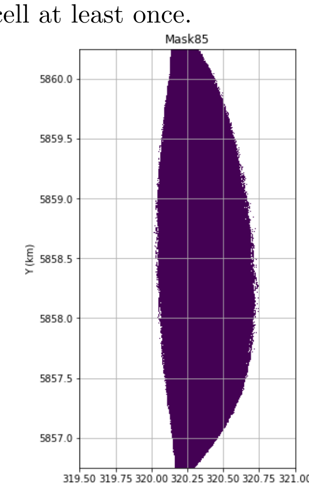


Figure 12: Mask85

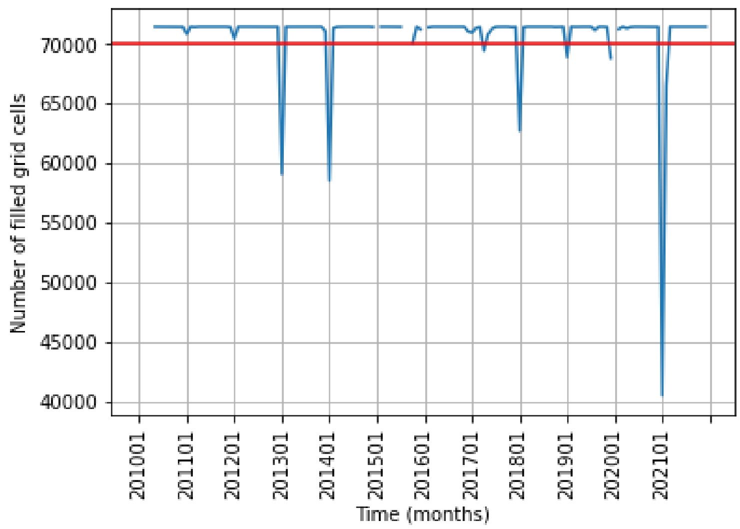


Figure 13: Number of filled grid cells per month. Months that have a value below the red line are not further analyzed.

3.2 Geometry

The distance that is measured by GPS at a certain location is marked with 'Beam 1' in figure 14, this is not the distance between the bottom and the sea surface, so a few corrections need to be done to get the right depth. The ADCP beam makes an angle of 20 °with the vertical, as shown in figure 14. Equation 1 was applied on the measured distance to obtain the correct depth value. The value for ϕ used in this equation is 90 - 20 = 70.

$$Depth_{corrected} = \sin(\phi) * beam_1 \tag{1}$$

The beam makes a 45 °angle with the front of the ferry (when the ferry is sailing towards Texel), as can be seen in figure 15. The ferry has a certain angle with respect to the north, this is denoted by ϕ in figure 15. This varies when the ferry is sailing. The corrected locations of the depth measurements are calculated by formula 2. The value for ϕ is different for each of the four beams, and are calculated by the equations below. H in these equations the heading of the ferry with respect to the north. The hd ADCP, at the rear of the ferry, is oriented 180 °with respect to the front ADCP, so that beam 1 makes a 45 °angle with the rear of the ferry.

$$\begin{pmatrix} N_a \\ E_a \end{pmatrix}_{corrected} = \begin{pmatrix} N \\ E \end{pmatrix}_{measured} + depth_{measured} * \cos(70) \begin{pmatrix} \cos(\phi_a) \\ \sin(\phi_a) \end{pmatrix} \tag{2}$$

ϕ_1 = H - 45
ϕ_2 = H + 90 + 45
ϕ_3 = H + 45
ϕ_4 = H - (90+45)

3.3 Sand wave characteristics

To calculate the sand wave height, polygons were made. These polygons were chosen visually on areas with more or less uniform sand waves. The standard deviations of the bed levels within the polygons were calculated for every month and multiplied by $\sqrt{2}$, because the sand waves are most similar to sine waves (Brakenhof, 2022). Different shapes were made and calculated, to check whether different shapes would influence the calculations, but this was not the case, as long as most of the sand waves were captured in the polygons.

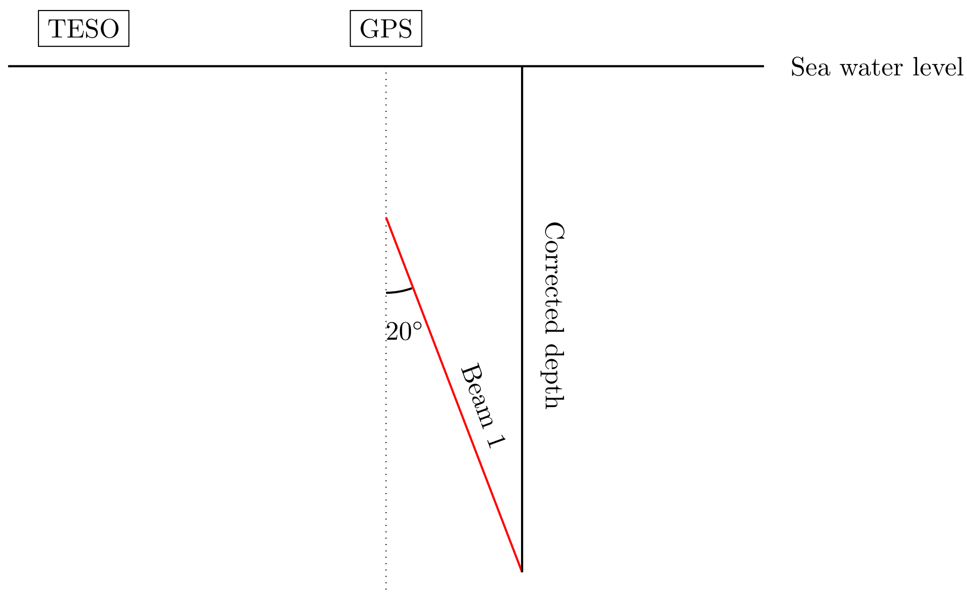


Figure 14: Schematic side view drawing.The ADCP beams make a 20°angle with the vertical.

Various transects were made to determine the wavelength, asymmetry and migration speed of the sand waves in the three different areas. The transects are ideally perpendicular to sand waves and chosen visually. Noise was reduced using a moving average with a window size of about 30 meter. Subsequently, the number of crests and troughs in each transect were determined. This was done by peak detection, a function in Python. To make sure as little as possible peaks caused by noise were detected, a threshold was used: a minimal horizontal distance of 100 meter was required between two neighbouring peaks and a minimal prominence of 0.15 m. These parameters were chosen by varying the width, height, prominence and threshold, plotting the transects together with the detected peaks and troughs and visually determining the parameters for which the least amount of false positives and false negatives were detected. The troughs were determined by multiplying the moving average by -1 and continue the peak detection same manner as described above. The sand wave length was calculated by averaging the distance between all the peaks on a transect for every transect. For every month, the mean, the standard deviation and the number of the sand waves on a transect were calculated, to check the data quality. The asymmetries were calculated by formula 3, in which i is the crest number and j the trough number, the distance from trough to crest divided by the distance from crest to crest. These distances are calculated from the start of the transect. The asymmetry has a value between 0 and 1 and has no unit. If AS = 0.5, the sand wave is fully symmetric. If AS is between 0 and 0.5, the sand wave is asymmetric and migrating into the direction of the transect (Brakenhof, 2022). The standard deviation of the asymmetry was also calculated. The transects were stacked on top of each other in chronological order and plotted as a Hovmoller diagram. Lines were drawn along the crests in the plots and the slope calculated, to determine the migration speed.

$$AS_i = \frac{trough_j - crest_i}{crest_{i+1} - crest_i} \tag{3}$$

4 Results and Discussion

4.1 Long term trends

The figures in appendix A show the averaged bathymetries of the Marsdiep inlet in July with the area-mean depth subtracted. These bathymetries were made for every month, from January 2010 until December 2021. The bathymetries of July of every year are presented next to each other to visualize the evolution of sand waves in time steps of one year. The long term evolution is analyzed manually by tracking different bedform

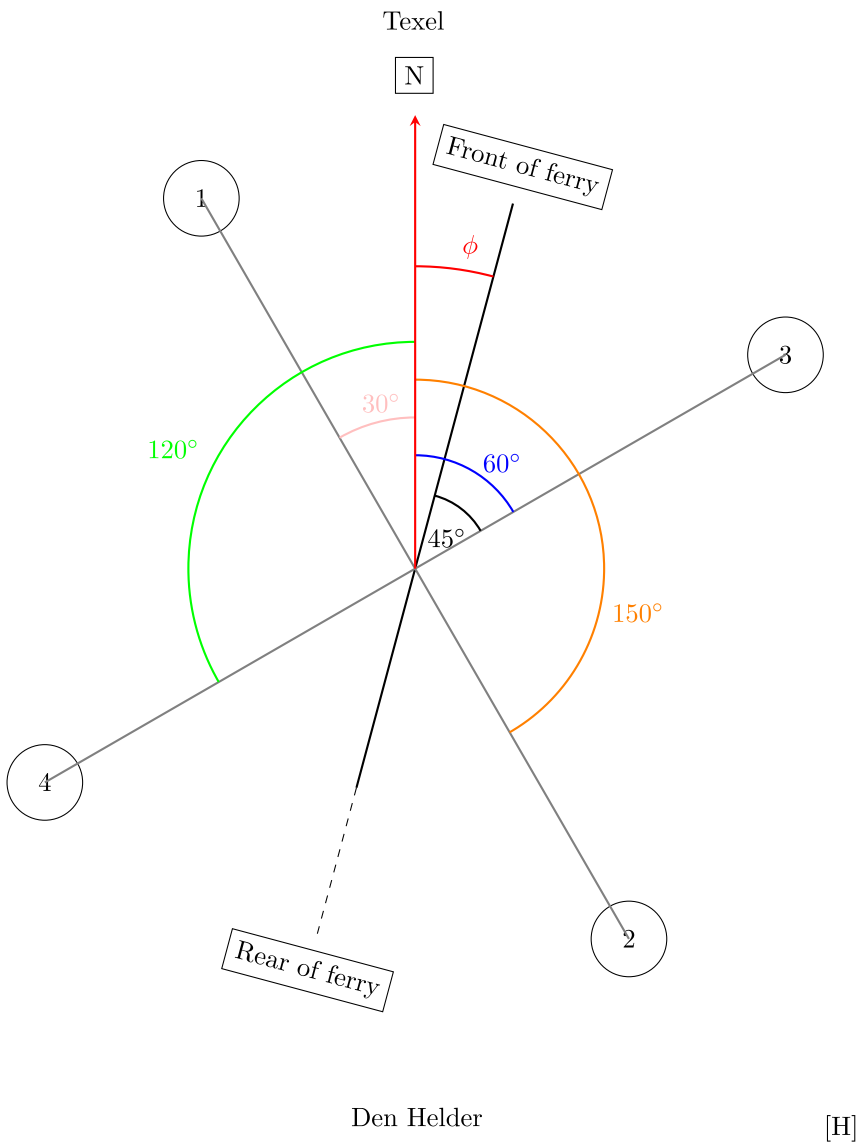
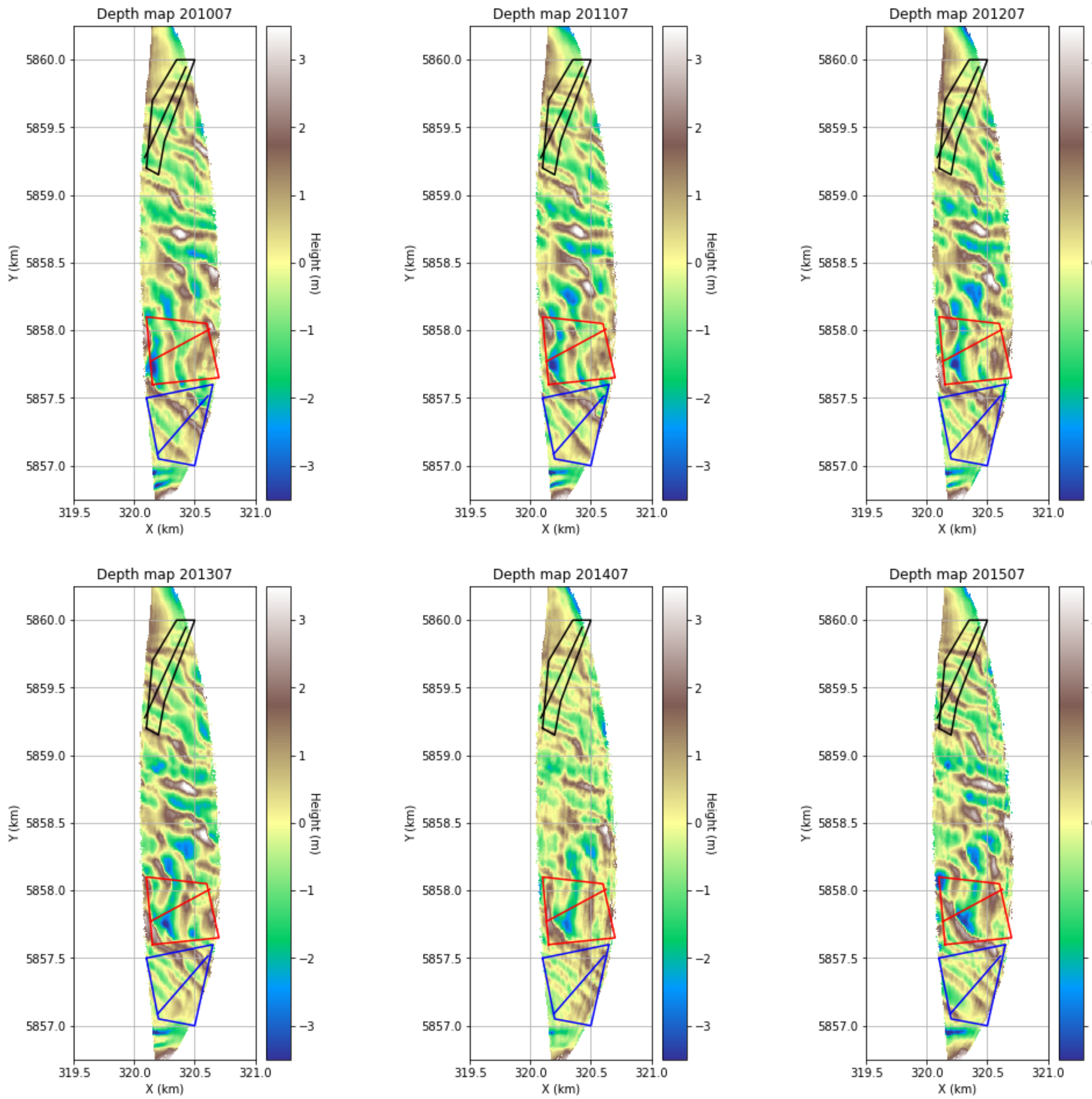
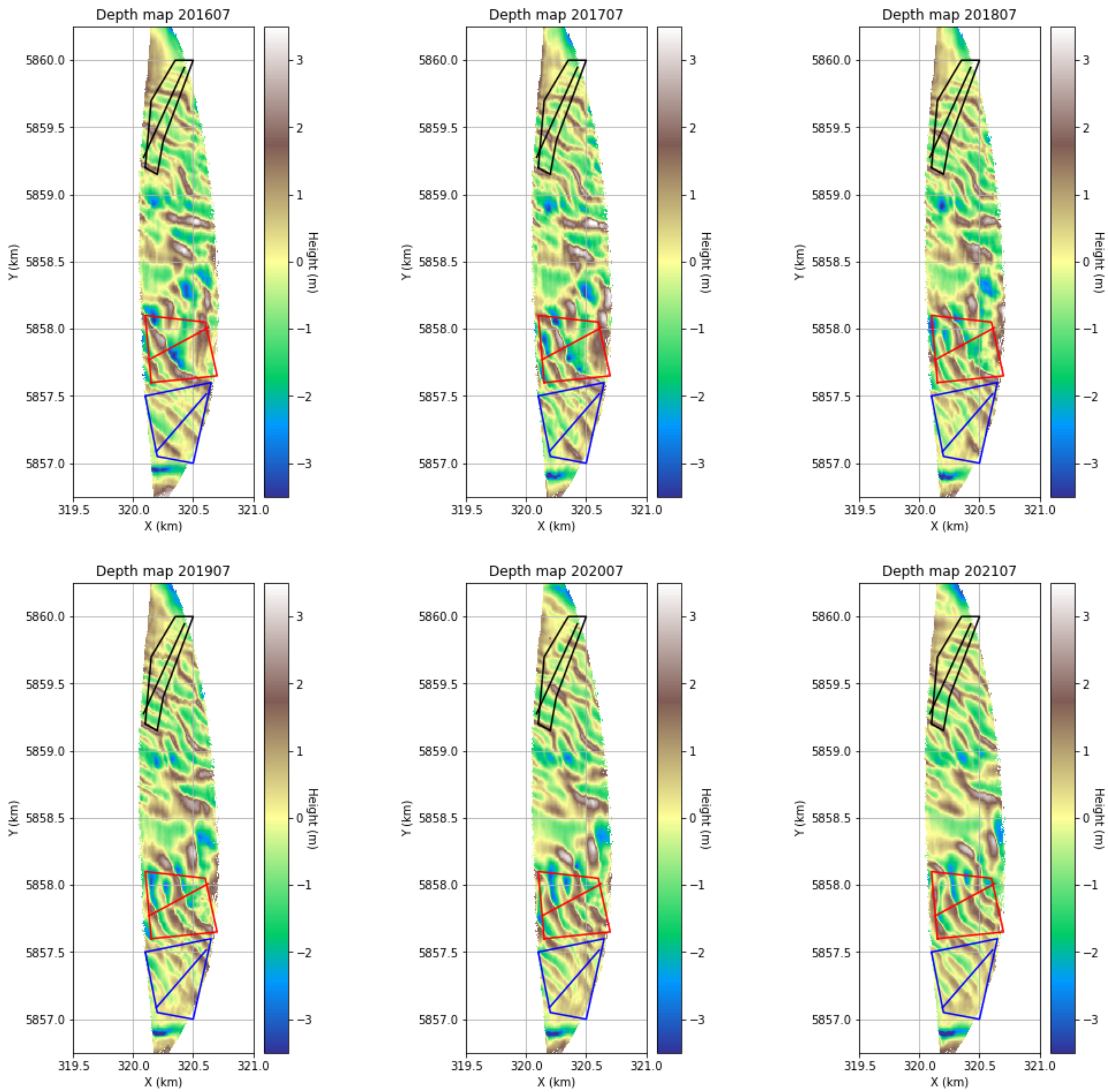


Figure 15: Schematic top view of the Texel (north) side of the ferry with the four ADCP beams (denoted with a number in a circle). Beam 1 (3) makes a negative (positive) 45°angle with the front of the ferry. The heading (ϕ) with respect to the north is in this example 15°. At the Den Helder (south) side of the ferry, beam 1 (3) makes a negative (positive) 45°angle with the rear of the ferry.

features visually.

When looking at the depth map of July 2010, a very deep trench, indicated with dark blue, is visible at 320 km east and 5857.6 km north. This dark blue spot migrates to the east and is visible in every successive year, up to 2016. From then, this deep through decreases in depth, until it eventually disappears.





In three regions, a series of consecutive sand waves were found, these are analyzed further according to the methods described in section 3.3.

4.2 Sand wave characteristics

4.2.1 Sand wave height

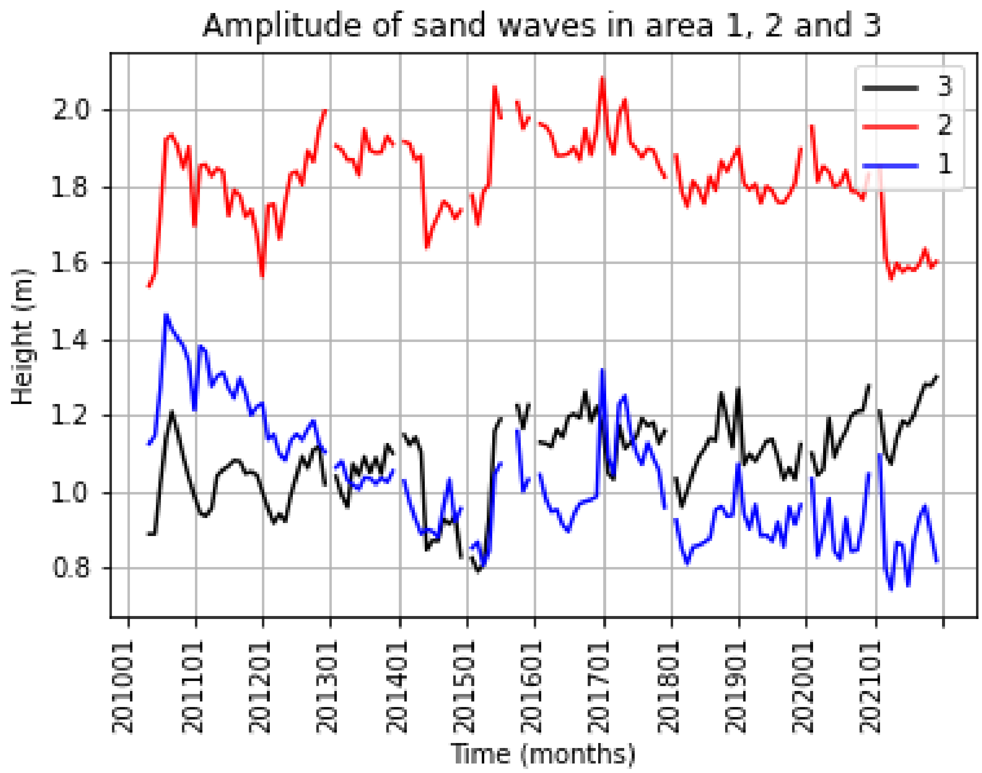


Figure 16: Amplitude of transects 1, 2 and 3.

Figure 16 shows the amplitude for area 1, 2 and 3 in blue, red and black respectively. The y axis still needs to be multiplied by 2 to obtain the sand wave height, as described by Lefebvre et al. (2022). A few things are noticeable when looking at the figure. Between June 2010 and July 2010, the amplitude increases with about 20 centimeter for all the regions. The same increase can be found between May 2015 and June 2015. When looking at the depth map figures in appendix A, it is indeed visible that the colors that represent the sand waves, are more contrasting in July 2010 (June 2015), than in June 2010 (May 2015). This is especially visible in area 3.

Another thing that is remarkable in figure 16, is a declining trend in area 3. This is especially from mid 2010 to begin 2015, and again from begin 2017 to end 2021.

Furthermore, it seems that a seasonal cycle can be found in area 3, with a higher amplitude during summer months and a lower amplitude during winter months. This trend is visible from 2010 to 2014.

4.2.2 Sand wave asymmetry

Figure 17 shows the asymmetry for the sand waves in transect 1. The blue line represents the mean asymmetry of the sand wave lengths and black line represents the number of sand wave crests in this transect. The standard deviation is in light blue. The mean asymmetry of the sand waves in this transect is 0.43 with a standard deviation of 0.07.

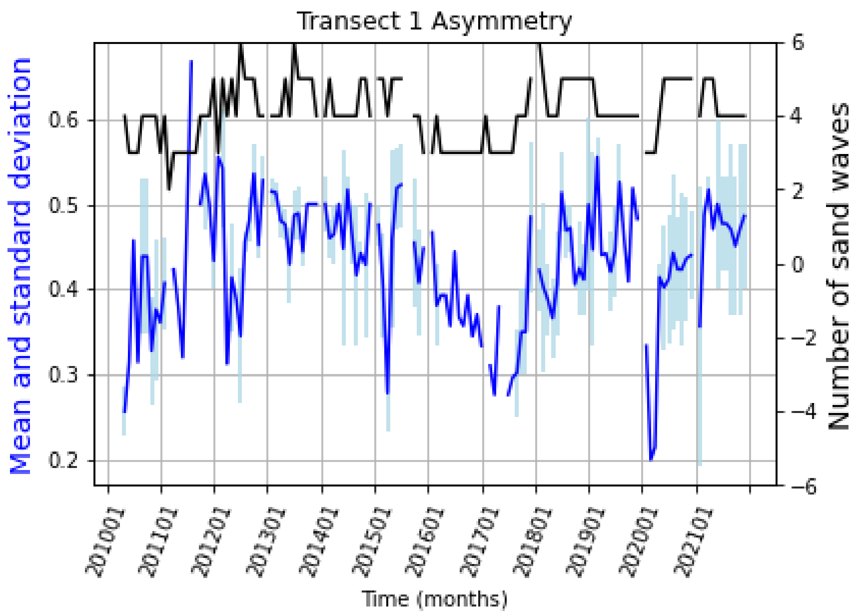


Figure 17: Asymmetry transect 1.

Figure 18 shows the asymmetry for the sand waves in transect 2. The red line represents the mean asymmetry of the sand wave lengths and black line represents the number of sand wave crests in this transect and the pink line represent the number of sand wave troughs. The standard deviation is represented by the vertical pink lines. The asymmetry is missing for a lot of months. This can be explained by a very large and deep trench along the transect. This trench starts in 2010 and lasts until the end of 2015. This can be seen in the depth map figures in the appendix, but even more clear in the lower left in figure 26. The cause of this is unknown, and it is not observed by Buijsman et al. (2008a), so it is unknown when the depth started to be that deep. It is not considered a sand wave, because the dimensions do not fall with in sand wave ranges, therefore, it is hard to determine the mean asymmetry, because the numbers are not very represent. However, the mean asymmetry and the standard deviation of the sand waves in this transect were still calculated and were 0.44 and 0.07 respectively.

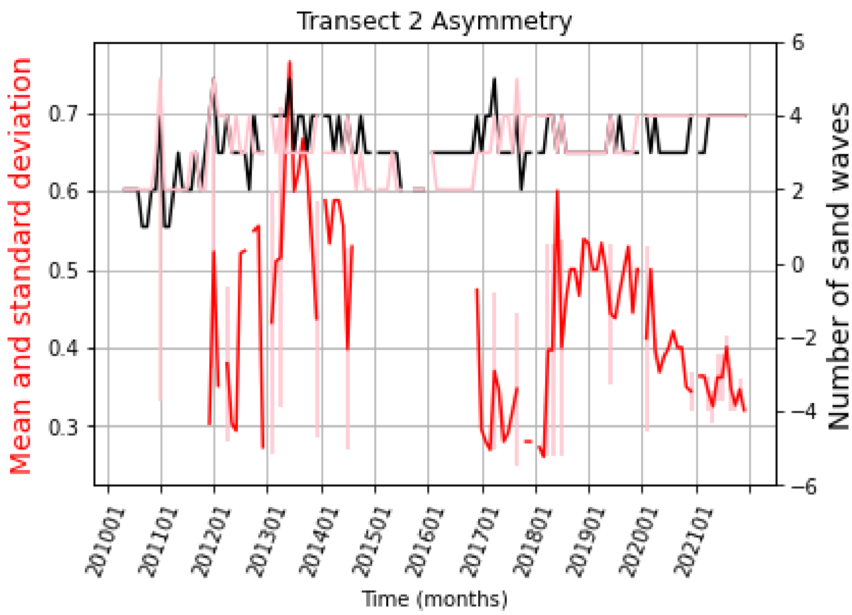


Figure 18: Asymmetry transect 2.

Figure 19 shows the asymmetry for the sand waves in transect 3. The black line represents the mean asymmetry of the sand wave lengths and green line represents the number of sand wave crests in this transect. The standard deviation is in gray. The mean asymmetry of the sand waves in this transect is 0.5 with a standard deviation of 0.09, which indicates that the sand waves would be symmetric. However, looking at the figure, a lot of variability is visible. This is not a result of poor peak detecting, as can be seen in 26, the red peaks align quite good with the observable sand waves. It is rather due to problems with data collection, which will be explained later.

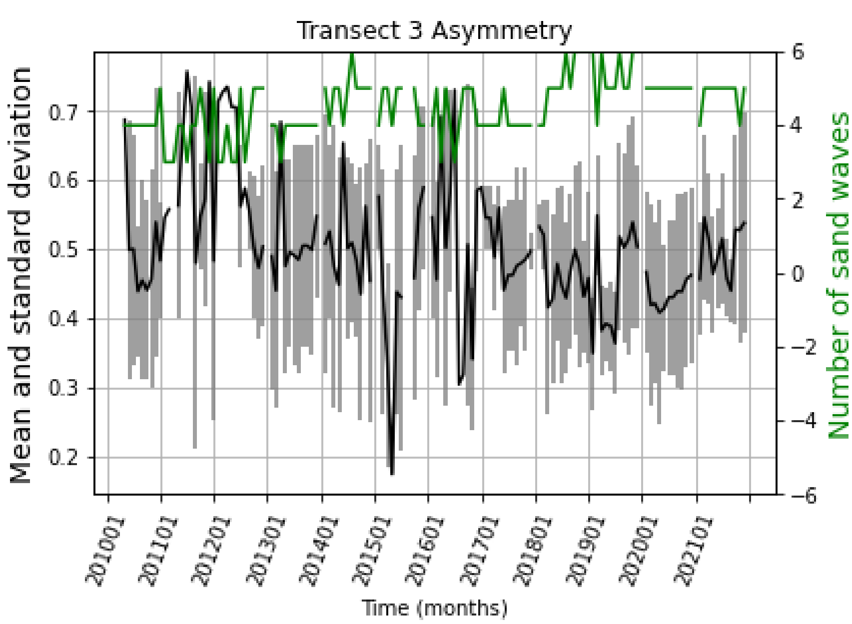


Figure 19: Asymmetry transect 3.

4.2.3 Sand wave length

Figure 20 shows the average sand wave length for each month, in transect 1. The average sand wave length is in blue and the number of sand wave crests is in black and te standard deviation in light blue. The sand wave length in this transect is higher during 2015, 2016 and 2017. In the same years, we see a lower value for asymmetry, which aligns with literature. The average sand wave length is 144 m and the mean standard deviation of the wavelength 14 meter. This is much lower than any sand wave observed by Buijsman et al. (2008a), but this part of the Marsdiep is not analyzed by them. The shorter sand waves are probably due to the water depth, which is shallower than the rest of the inlet, as can be seen in figure 7 a) and b).

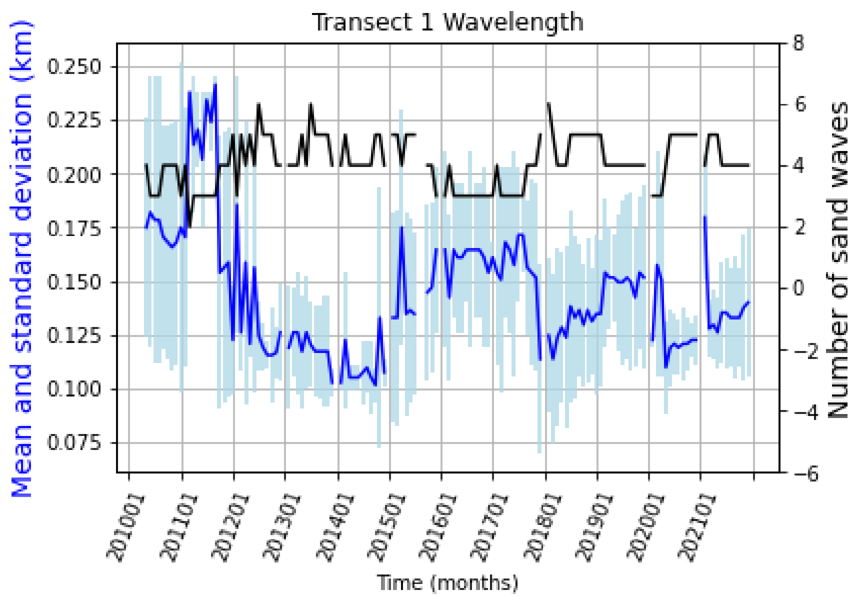


Figure 20: The mean sand wave length in transect 1 for each month in blue and the number of sand wave crests in transect 1 in black.

Figure 21 shows the average sand wave length for each month, in transect 2. The average sand wave length is in red and the number of sand wave crests is in blue and the standard deviation in pink. The mean wave length for this transect is 187 meter with a standard deviation of 7 meter. These numbers are probably exaggerated, due to the long length in the years 2015 and 2016. As explained before, the sand waves observed in these years are not representative, due to the big trench along the transect. The average wavelength from 2019 on is shorter, which is quite visible in the upper right corner in figure 25. The wave length between 2010 and 2013 is very variable, this is because a large number of false positives, which can be clearly observed in the lower right corner of figure 26. The thresholds could be chosen differently, but this would result in false negatives in the upper right corner. For future research, it is recommended to split this transect, either in time or in space, so the sand wave peaks can be detected accurately.

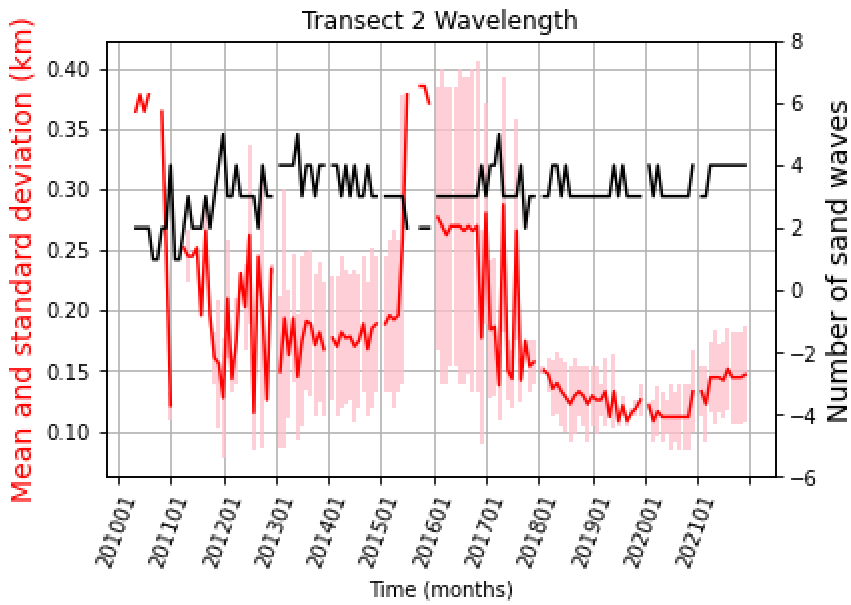


Figure 21: The mean sand wave length in transect 2 for each month in red and the number of sand wave crests in transect 2 in blue.

Figure 22 shows the average sand wave length for each month, in transect 3. The average sand wave length is in black and the number of sand wave crests is in green. The standard deviation is in grey. The average sand wave length for this transect is 148 meter with a standard deviation of 2 meter. This is lower than observed by Buijsman et al. (2008a) in this area. This can be caused by the high amount of suspended sediment transport, instead of bed load transport, as described by Buijsman et al. (2008b). Figure 23 shows the difference between suspended (right panel) and bed load (left panel) transport in the Marsdiep inlet, and the suspended transport in the northern part is stronger than the bedload transport.

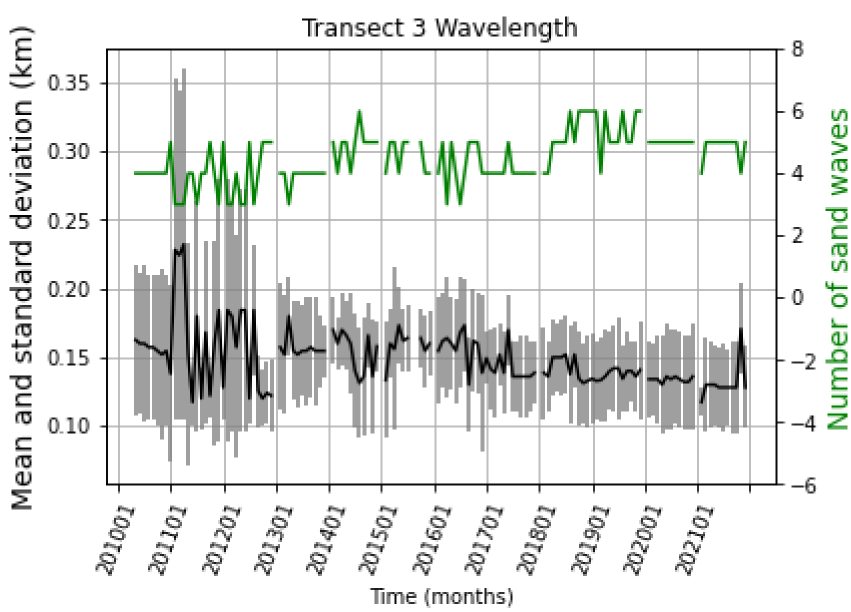


Figure 22: The mean sand wave length in transect 1 for each month in black and the number of sand wave crests in transect 1 in green.

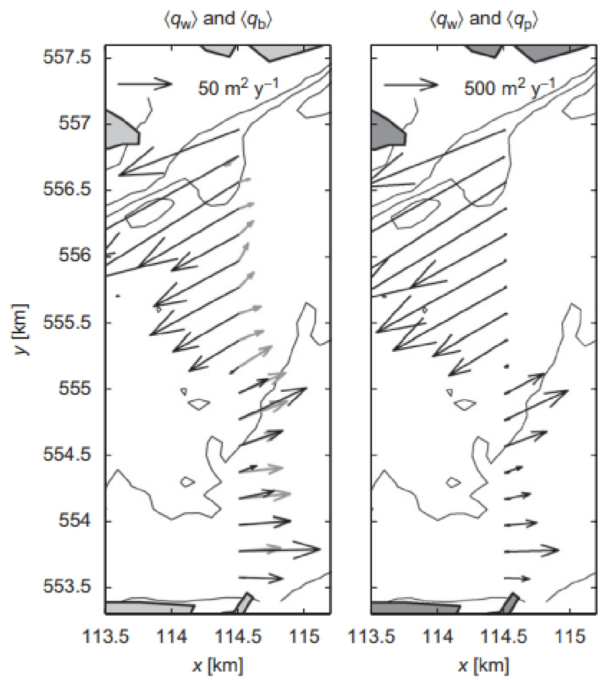


Figure 23: Comparison between mean 'measured' (grey vectors) and mean predicted bedload transport (left panel; black vectors) and suspended load transport (right panel; black vectors) based on the depth-averaged currents of data set I for the period 1999–2002. Note the difference in scales between the left and right panels (Buijsman et al., 2008b).

4.2.4 Sand wave migration

Figure 24, 25 and 26 show the migration of the sand waves of transect 1, 2 and 3 respectively. The x-axis in each figure represents the transect, on which the colors represent the sand wave height in meters. Brown/white indicate positive sand wave heights, which represent crests, and green/blue indicate negative sand wave heights, which represent troughs, as can be seen in the color bars found right next to the figures. The location of transect 1 (2,3) can be found in blue (red, black) in the figures in appendix A. The y-axis is the time in months. A few white spaces are visible in the figures, these white lines represent the months for which it was not possible to make a depth plot, due to a low amount of data.

Figure 24 shows the migration of the sand waves in transect 1. The red dots indicate the sand wave peaks detected, to check if they are aligning with the sand waves. In 2010, 4 sand wave crests were present in this transect. In 2011, two of these sand wave (last two) crests merged into one, and the first crest splits into two separate sand waves. In 2020, the first sand wave splits into two sand waves. The sand waves are moving north east with an average speed of 80 m per year, perpendicular to the direction of the chosen transect. This is because the maximum flood in this area, which is to the east, is higher than the maximum ebb, which was observed by Buijsman et al. (2008b) and can be seen in figure 27.

The migration of the sand waves along transect 2 are visualized in figure 25. The sand wave are migrating to the north east as well, but with an average migration speed of 70 m per year. The difference with the more southern area in transect 1, is due to the deeper laying bed forms and the smaller difference between the maximum flood and the maximum ebb.

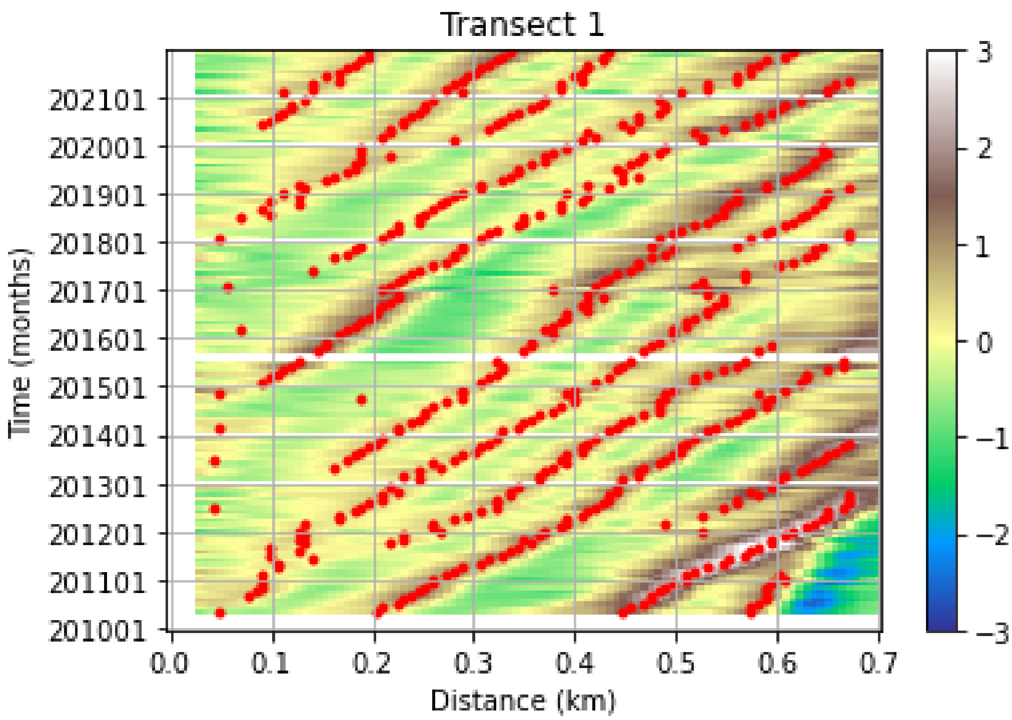


Figure 24: Timestack transect 1.

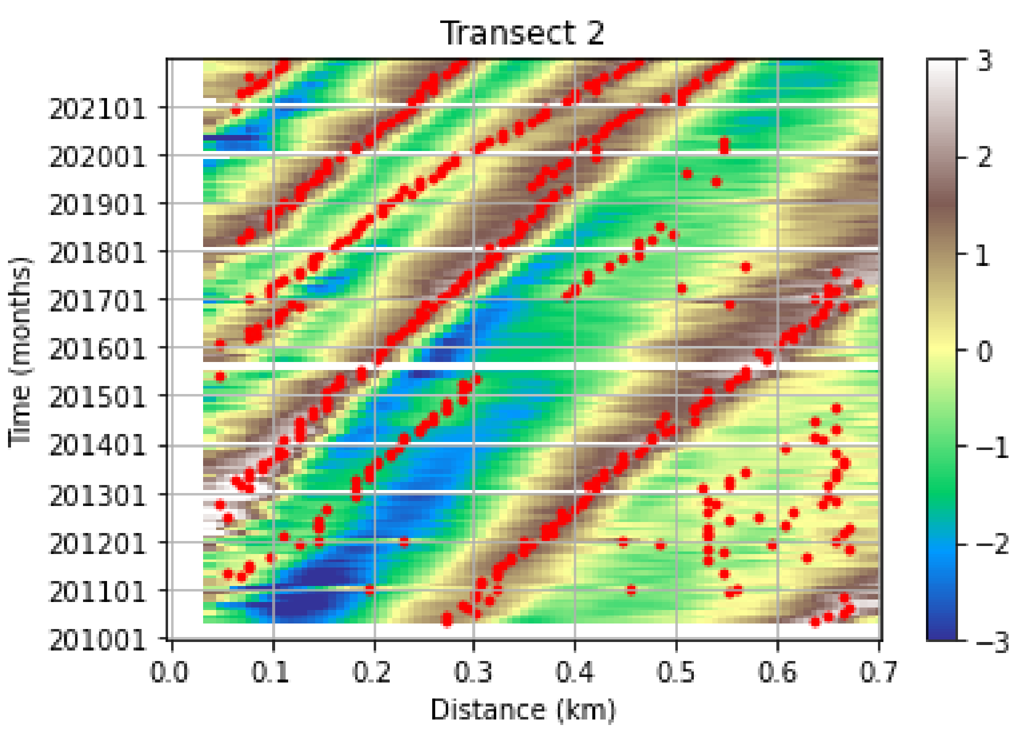


Figure 25: Timestack transect 2.

Figure 26 shows the Hovmoller plot for the third transect. Due to the delay in data storage, which will be explained later in the methodological limitations section, the sand waves aligned are not along a nice line. Therefore, it was not possible to calculate the sand wave migration. When looking at the individual depth maps, no sand wave migration is visible. This can be explained by the water depth, which is deeper in this area than for the other areas, because deeper waters have lower tidal currents compared to shallower areas. Also, the maximum flood is of similar strenght as the maximum ebb in this area.

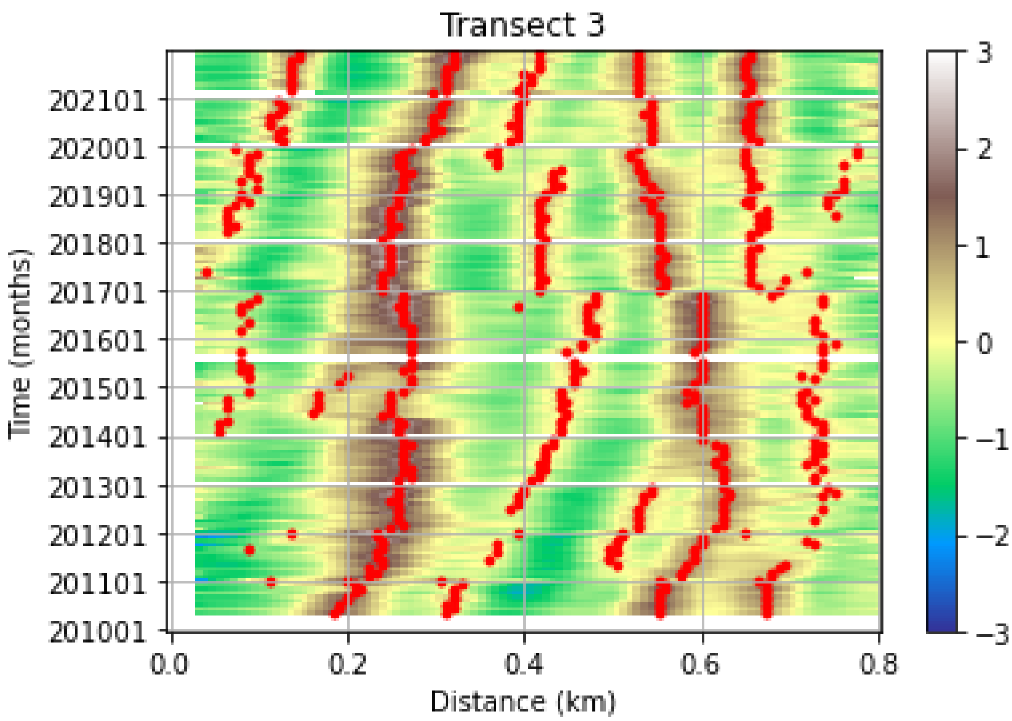


Figure 26: Timestack transect 3.

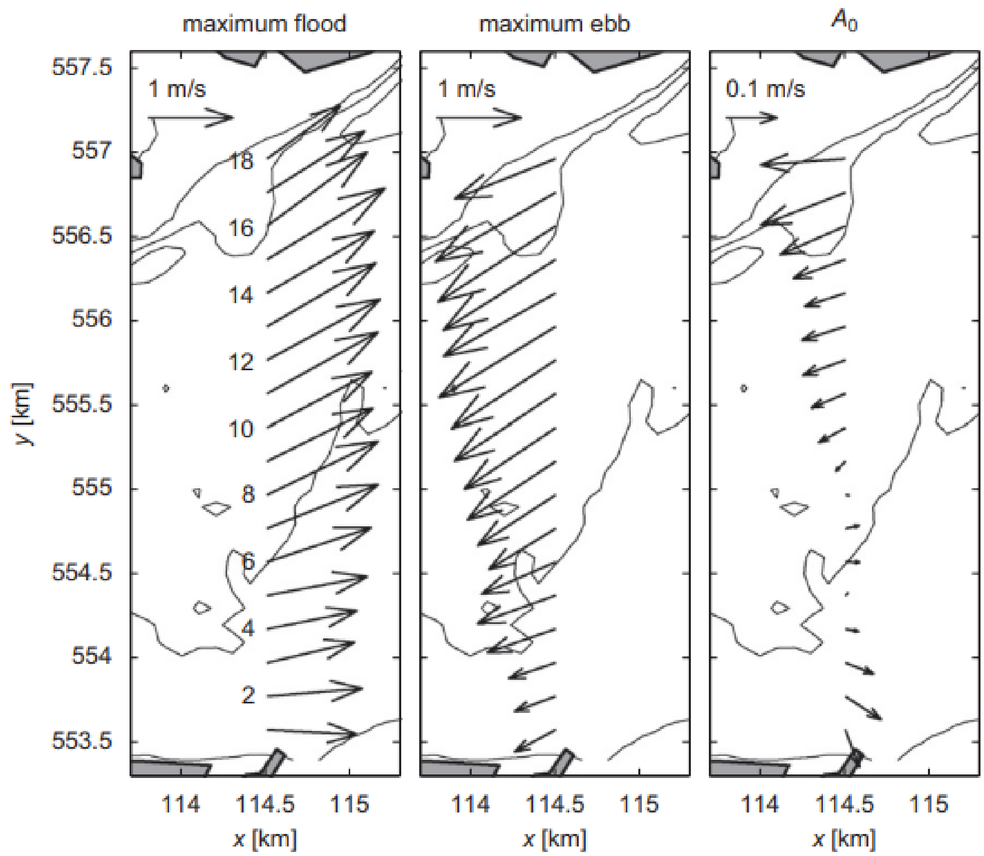


Figure 27: Left and second panel: measured depth-averaged maximum spring flood and ebb currents on May 6, 2000. Numbers refer to the stations. Third panel: tidal-mean currents determined with a harmonic analysis for the year 2000 (Buijsman et al., 2008b).

4.3 Research methods limitations

A delay between the position and the direction of the ferry can occur, but they are saved at the same timestamp. This delay can be up to 19 seconds and is differs between days. This is related to the number of fixed shore stations that the differential GPS can contact and calculate the positions with. Looking at the migration plot of transect 3, figure 26, the sand waves seem to 'jump' a few meters in January 2017. This can be the result of the change of the ferries. Before 2017, 1 GPS in the middle of the ferry saved the location for both ADCPs and corrected for it. After 2017, both ADCPs had their own GPS, but one was broken for a while, so the GPS of the other ADCP location was used, which was not corrected for. Water depth data from Rijkswaterstaat is measured every 10 minutes at Den Helder Veerhaven. Water depth from ADCP is measured every second at location between Den Helder and Texel. The Rijkswaterstaat data was interpolated to do the tide correction.

To test whether the correction of the geometry was proper, the locations of the beams of several depth measurements were calculated using equation 2. It is expected that beam 1 (3) and 2 (4) are at opposite sides and that beam 1 and 2 are perpendicular to beam 3 and 4, as can be seen in figure 15. As can be seen in figures 29 and 28, the locations of the depth measurements are corrected as expected. The pink, orange, blue and green dots represent the locations of the depth measurements of beam 1, beam 2, beam 3 and beam 4 respectively. The white dot represents the GPS location and the white line respects the direction in which the ferry is sailing. Figure 30 shows the depth measurements of the Den Helder ADCP and the Texel ADCP in the same figure at the same moment. The heading is the same for both ADCPs, but it was expected that the two GPS locations were located on the same line, which is not the case. Because of this, measurements could be placed into the wrong grid cell, which could interfere with further analysis.

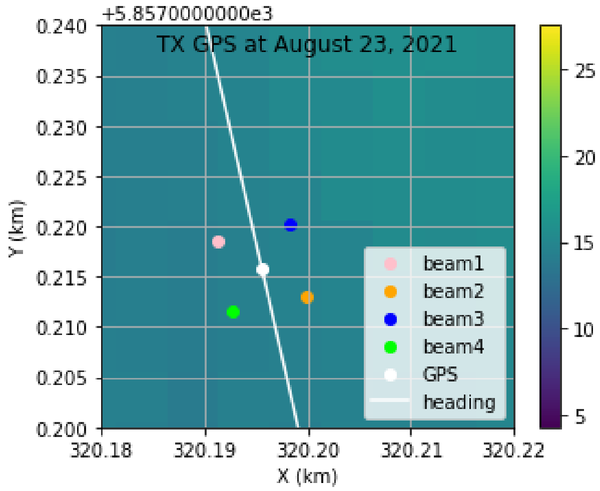


Figure 28: The ADCP at the north of the ferry (Texel side).

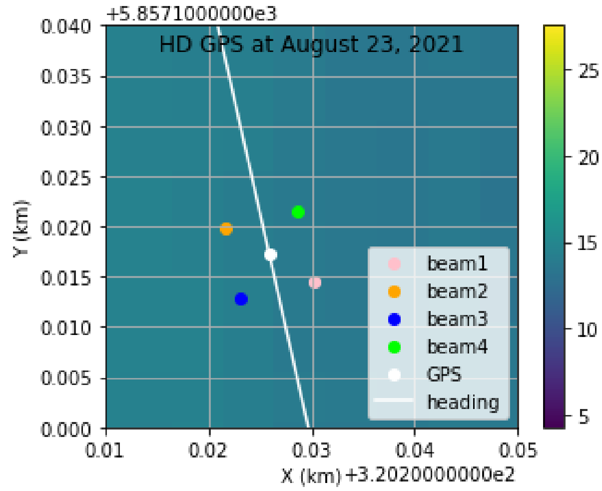


Figure 29: The ADCP at the south of the ferry (Den Helder side).

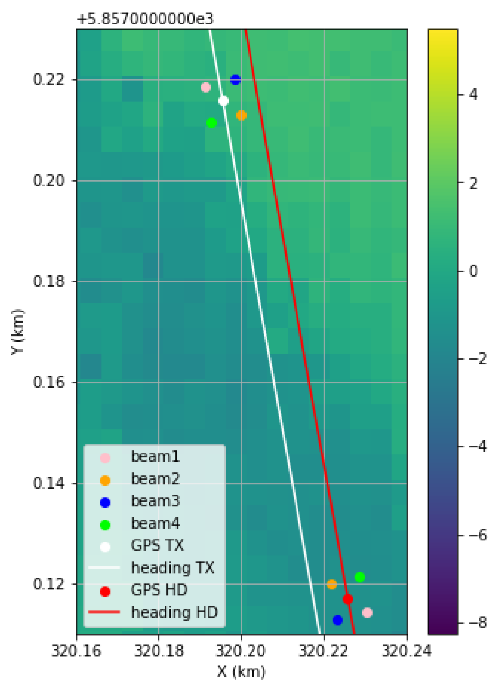


Figure 30: Locations of the depth measurements.

The method to detect the sand wave crests and troughs is not perfect. Both false positives and false negatives were observed, which may result in inaccurate calculations. A lot of different methods were tried to improve the peak detection, but the data is too variable to fit a universal method. A better way would be to do 2D peak detection. The method to calculate the migration speeds was also not perfect, since it was only the average for 10+ years. For example, in figure 26, the left sand wave in 2019 splits into two sand waves, so the migration speed can not be the same for the two separate sand waves. For future research, cross correlation techniques like described in Buijsman et al. (2008a) could be used to improve the tracking of sand wave migration. This was also tried, but turned out to be very challenging.

5 Conclusion

5.1 What are the sand wave characteristics?

Table 2 shows the sand wave characteristics in the three different areas (amplitude) and the different transects (wavelength, asymmetry, migration).

	Area 1		Area 2		Area 3	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
Amplitude (m)	2.9	0.22	3.64	0.11	2.16	0.11
Wave length (km)	0.144	0.14	0.187	0.07	0.148	0.02
Asymmetry	0.43	0.07	0.44	0.07	0.5	0.09
Migration (m y^{-1})	80		70			

Table 2: Mean sand wave characteristics between 2010 and 2021.

5.2 How did the sand wave characteristics change over time?

When looking at figure 24, the sand wave crests in 2020 and 2021 seem to have a more gentle slope than the years before. This can indicate a faster migration into the direction of the east. The increases in sand wave height, described in 4.2.1, can be due to storms, however, according to the The Royal Netherlands Meteorological Institute (KNMI), there were no heavy storms during the months where the increases happened. The cause of these increases remains unknown. Buijsman et al. (2008a) analyzed the sea bed of the Marsdiep inlet using 1 ADCP with three depth measures every second. This research collected the data by using 2 ADCPs with both four depth measurements every second, so more data was present. This results in more detailed depth maps. Area 2 corresponds to area I in 1 and area 3 corresponds to area III and IV in 1. The crest orientation did not change significantly. The sand wave length, height and asymmetry were smaller than observed before, this can be a result of more suspended sediment transport, instead of bed load transport via sand waves.

6 Acknowledgement

I would like to take this opportunity to thank my supervisors Johan and Maarten. First, for giving me the chance to do my thesis at NIOZ, and above all, for all their help, knowledge, insights and patience. We had a meeting every week, and even though I was sometimes insecure about my work, I always felt so good after the meetings, with fresh input for the next week! I learned so much during the thesis, both academically as personally. Although I was sometimes struggling (in particular with the coding), I certainly enjoyed my time as well, especially the few months I was living on Texel!

References

Buijsman, M.C. and H. Ridderinkhof (2007). "Long-term ferry-ADCP observations of tidal currents in the Marsdiep inlet". In: *Journal of Sea Research* 57.4, pp. 237–256.

Veen, J. Van (1935). "Sand waves in the North Sea". In: *The International Hydrographic Review*.

Buijsman, M.C. and H. Ridderinkhof (2008a). "Long-term evolution of sand waves in the Marsdiep inlet. I: High-resolution observations". In: *Continental Shelf Research* 28.9, pp. 1190–1201. ISSN: 0278-4343. DOI: https://doi.org/10.1016/j.csr.2007.10.011. URL: https://www.sciencedirect.com/science/article/pii/S0278434308000903.

– (2008b). "Long-term evolution of sand waves in the Marsdiep inlet. II: Relation to hydrodynamics". In: *Continental Shelf Research* 28.9, pp. 1202–1215. ISSN: 0278-4343. DOI: https://doi.org/10.1016/j.csr.2008.02.014. URL: https://www.sciencedirect.com/science/article/pii/S0278434308000861.

De Swart, HE and JTF Zimmerman (2009). "Morphodynamics of tidal inlet systems". In: *Annual review of fluid mechanics* 41, pp. 203–229.

Ridderinkhof, H. (1990). *Residual currents and mixing in the Wadden Sea*. Rijksuniversiteit te Utrecht.

Oost, Albert Peter et al. (1995). *Dynamics and sedimentary developments of the Dutch Wadden Sea with a special emphasis on the Frisian Inlet: a study of the barrier islands, ebb-tidal deltas, inlets and drainage basins*. Faculteit Aardwetenschappen.

De Jonge, VN, K Essink, and R Boddeke (1993). "The Dutch Wadden Sea: a changed ecosystem". In: *Hydrobiologia* 265, pp. 45–71.

Eriksson, Britas Klemens et al. (2010). "Major changes in the ecology of the Wadden Sea: human impacts, ecosystem engineering and sediment dynamics". In: *Ecosystems* 13, pp. 752–764.

Postma, H. (1954). "Hydrography of the Dutch Wadden sea". In: *Arch. Neerl. Zool* 10, pp. 405–511.

Groeskamp, S., J.J. Nauw, and L.R.M. Maas (2011). "Observations of estuarine circulation and solitary internal waves in a highly energetic tidal channel". In: *Ocean Dynamics* 61.11, pp. 1767–1782.

Lefebvre, A. et al. (2022). "Morphology of estuarine bedforms, Weser Estuary, Germany". In: *Earth Surface Processes and Landforms* 47.1, pp. 242–256. DOI: https://doi.org/10.1002/esp.5243. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/esp.5243. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/esp.5243.

Bellec, V.K. et al. (2019). "Sandbanks, sandwaves and megaripples on Spitsbergenbanken, Barents Sea". In: *Marine Geology* 416, p. 105998.

Barnard, P.L. et al. (2006). "Giant sand waves at the mouth of San Francisco Bay". In: *Eos, Transactions American Geophysical Union* 87.29, pp. 285–289.

Besio, G. et al. (2008). "The morphodynamics of tidal sand waves: A model overview". In: *Coastal engineering* 55.7-8, pp. 657–670.

Dyer, K.R. and D.A. Huntley (1999). "The origin, classification and modelling of sand banks and ridges". In: *Continental shelf research* 19.10, pp. 1285–1330.

Allen, JRL (1980). "Sand waves: a model of origin and internal structure". In: *Sedimentary Geology* 26.4, pp. 281–328.

Hulscher, S. J. M. H. (1996). "Tidal-induced large-scale regular bed form patterns in a three-dimensional shallow water model". In: *Journal of geophysical research: Oceans* 101.C9, pp. 20727–20744.

Borsje, B. W. et al. (2014). "The role of suspended load transport in the occurrence of tidal sand waves". In: *Journal of Geophysical Research: Earth Surface* 119.4, pp. 701–716.

Damen, J.M, T.A.G.P. van Dijk, and S.J.M.H Hulscher (2018). "Spatially varying environmental properties controlling observed sand wave morphology". In: *Journal of Geophysical Research: Earth Surface* 123.2, pp. 262–280.

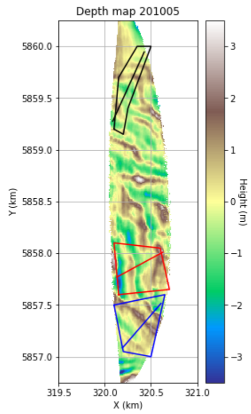
Van Rijn, L. C. (1984). "Sediment transport, part I: bed load transport". In: *Journal of hydraulic engineering* 110.10, pp. 1431–1456.

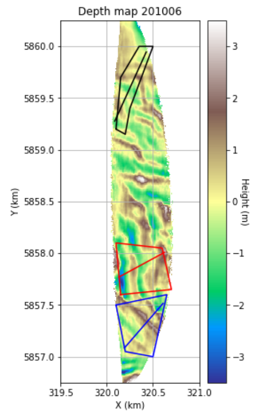
Bøe, R. et al. (2009). "Giant sandwaves in the Hola glacial trough off Vesterålen, North Norway". In: *Marine Geology* 267.1-2, pp. 36–54.

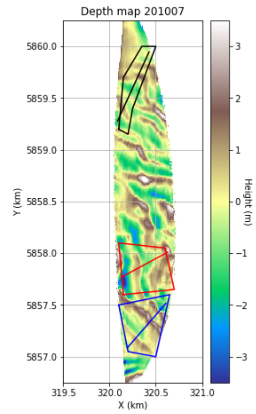
Brakenhof, L. (Mar. 2022). "Bedforms and their effect on sediment transport on ebb-tidal deltas". PhD thesis. DOI: 10.4233/uuid:c2fe811c-dc2e-4e1f-bb0c-dc43f11cd1eb.

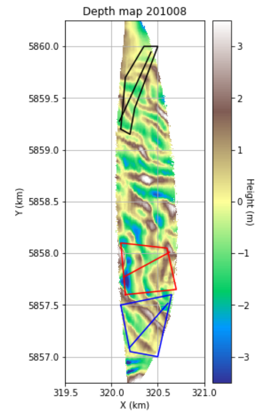
A Appendix

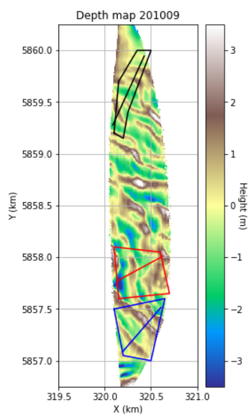
Depth profiles of every month between May 2010 and December 2021.

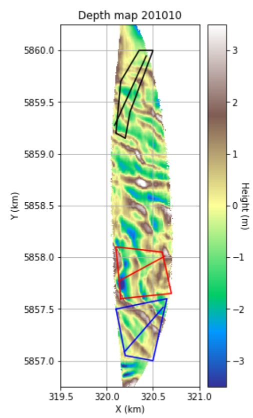


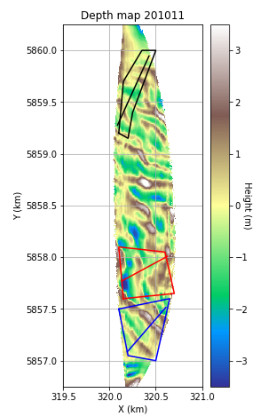


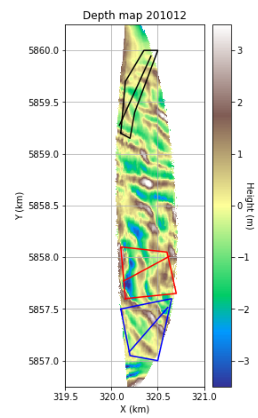


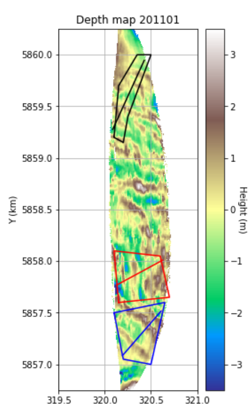


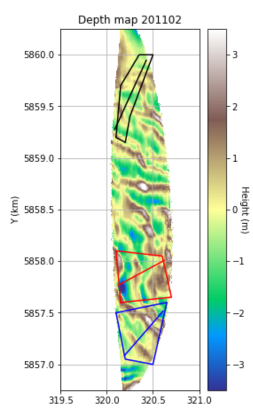


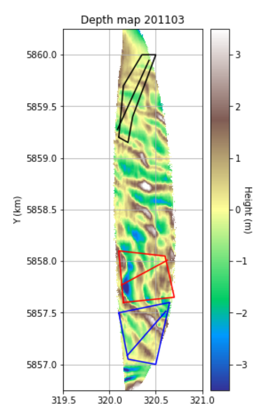


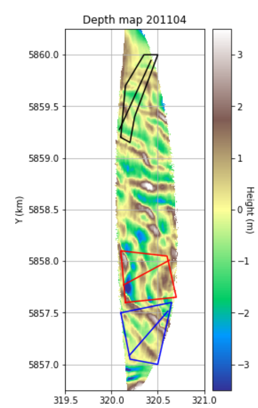


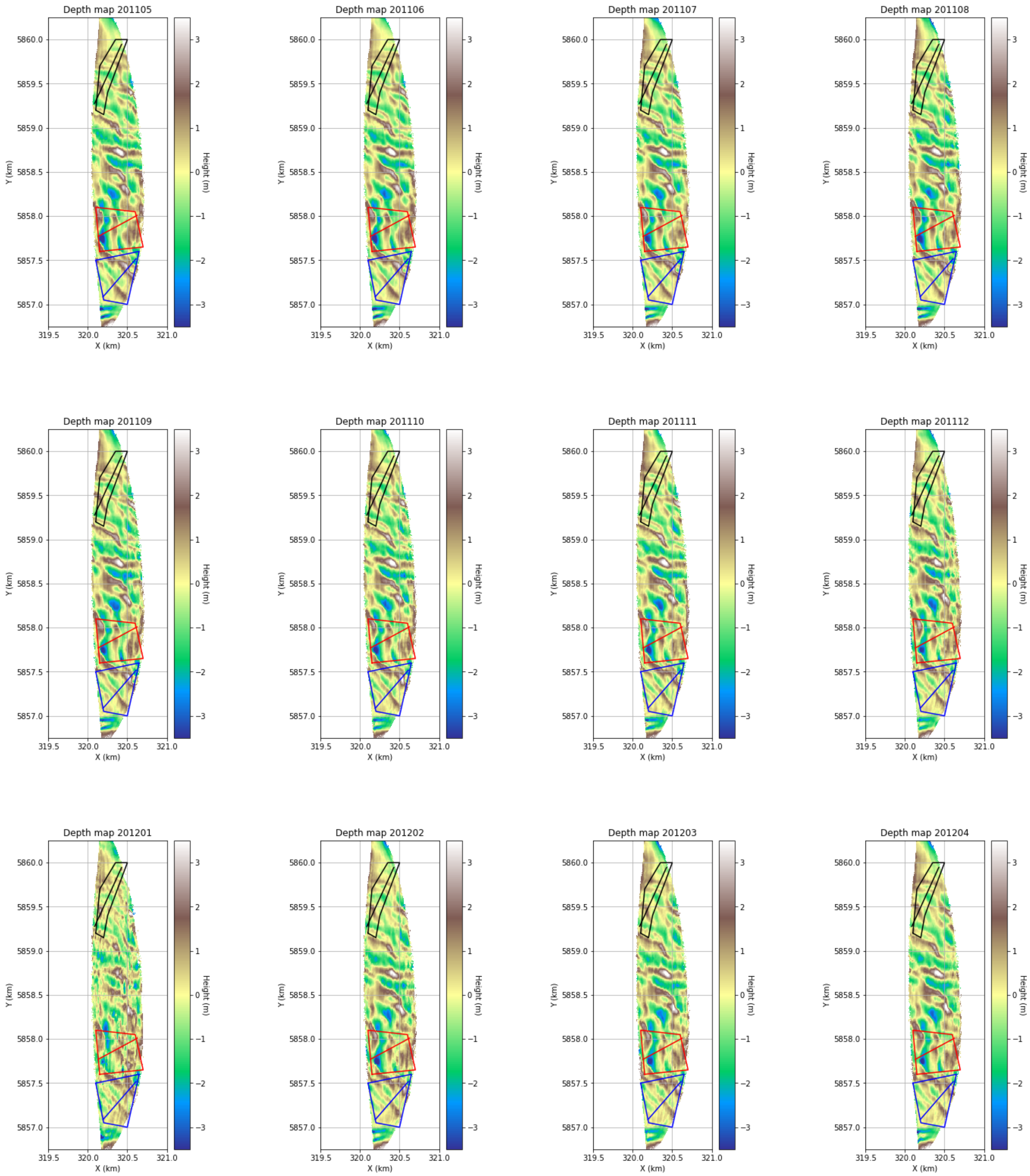


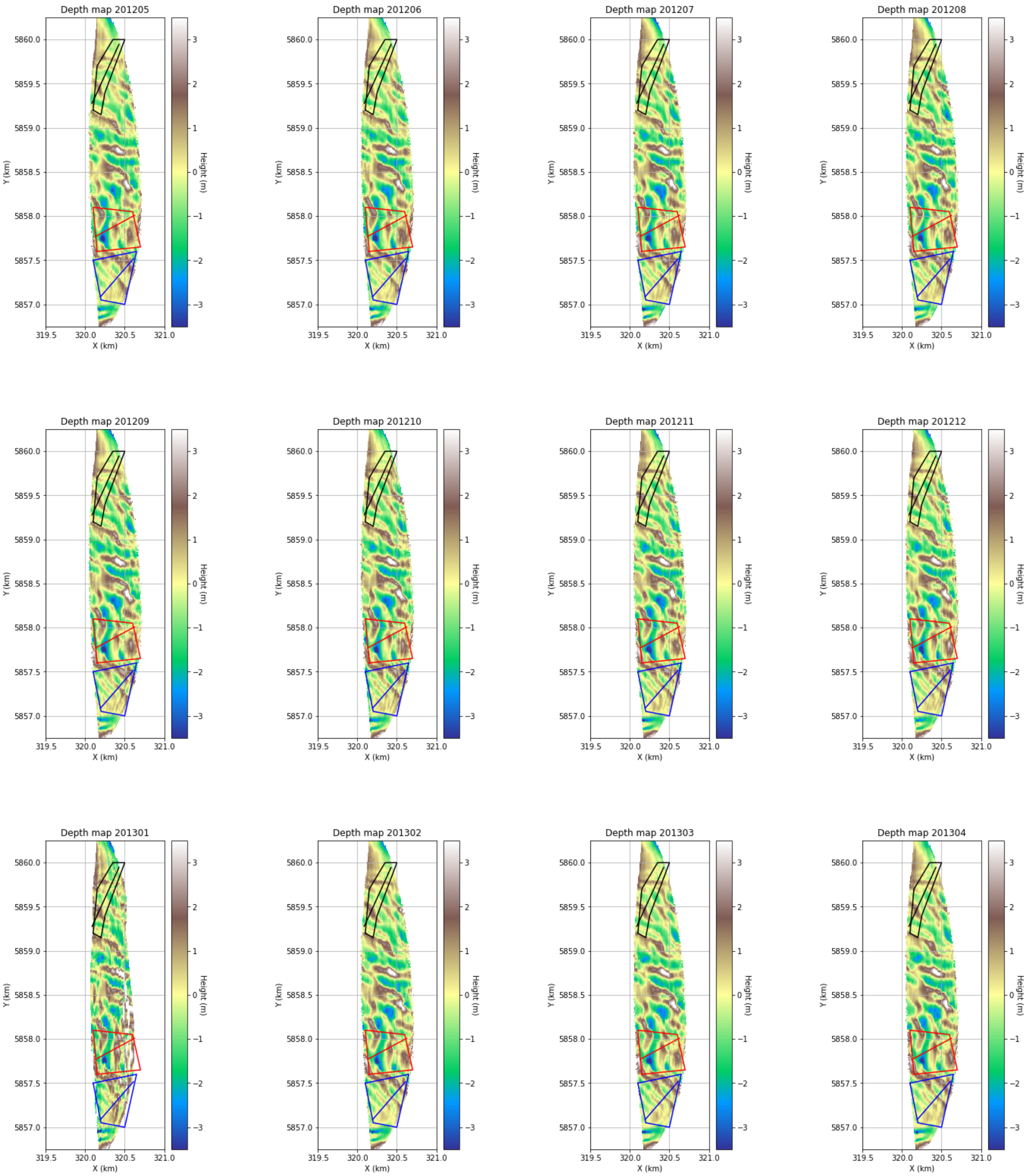


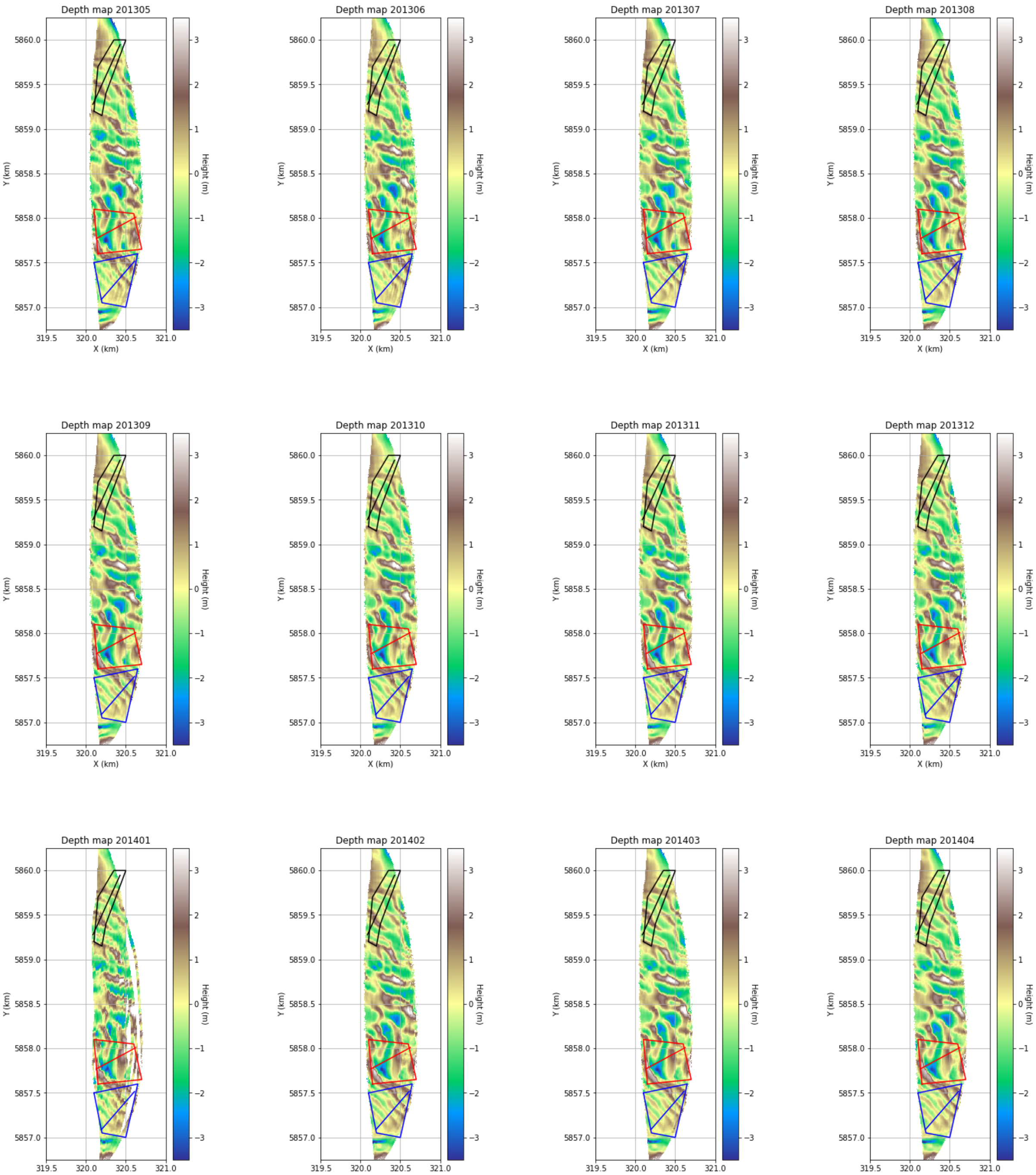


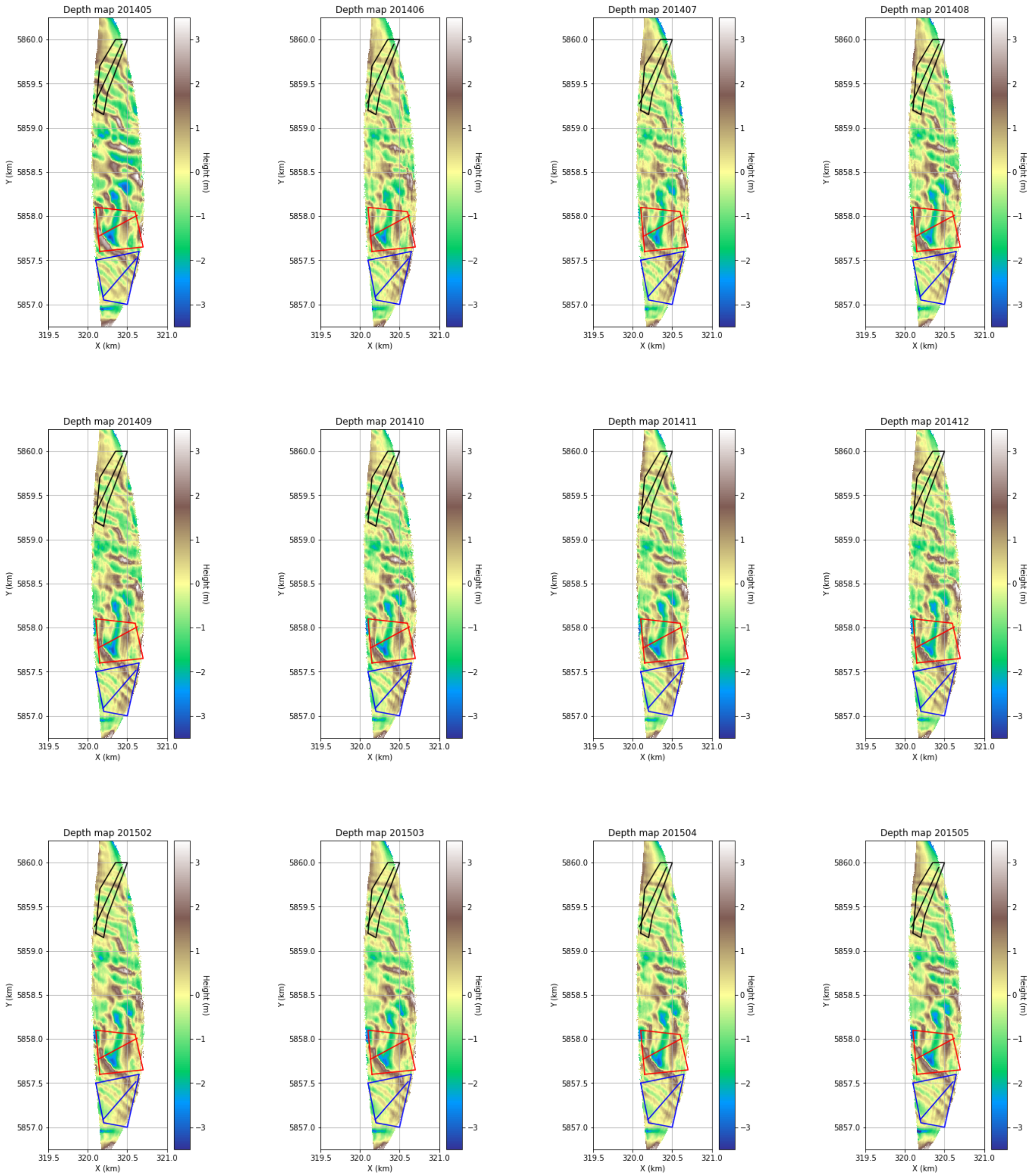


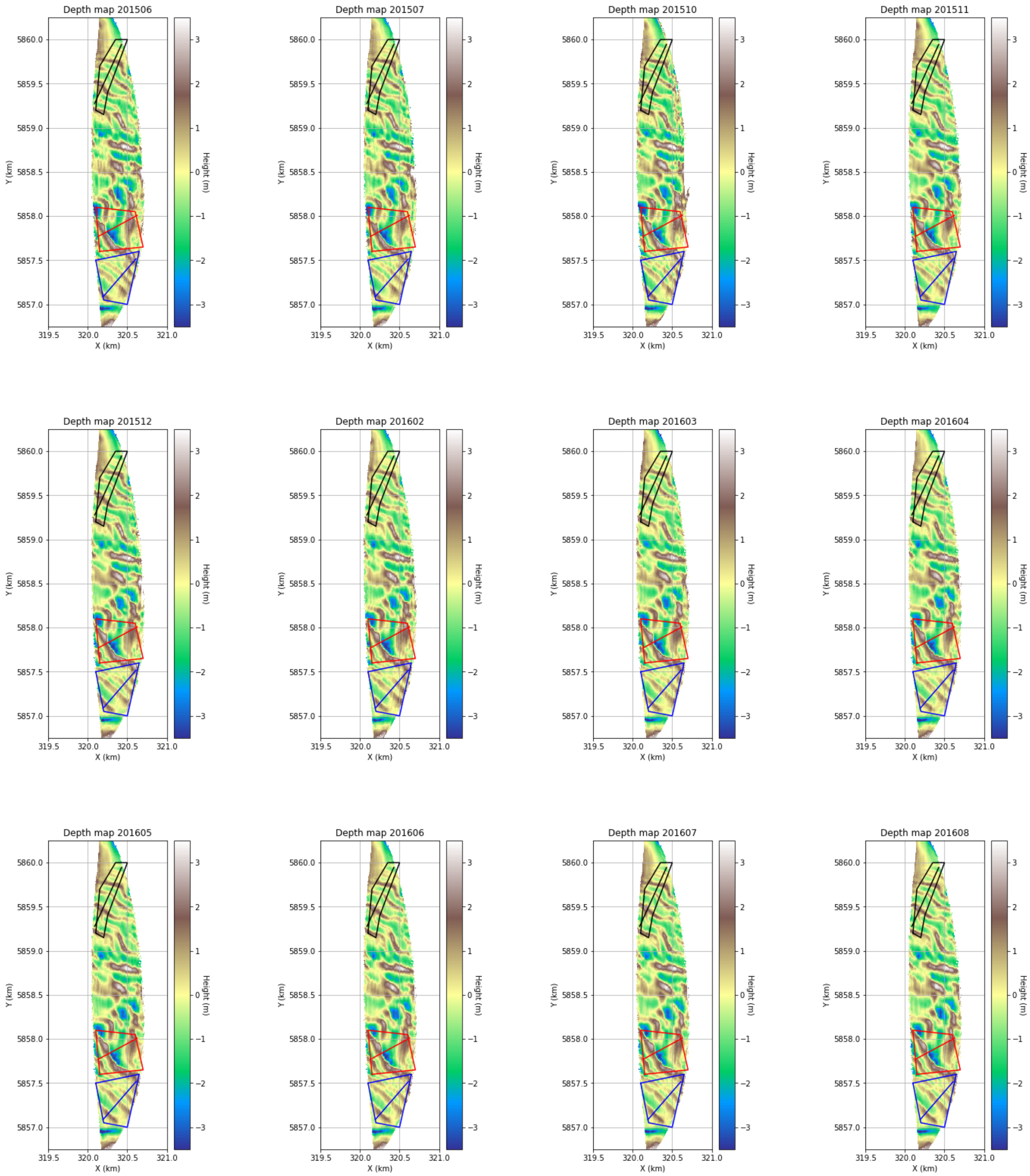


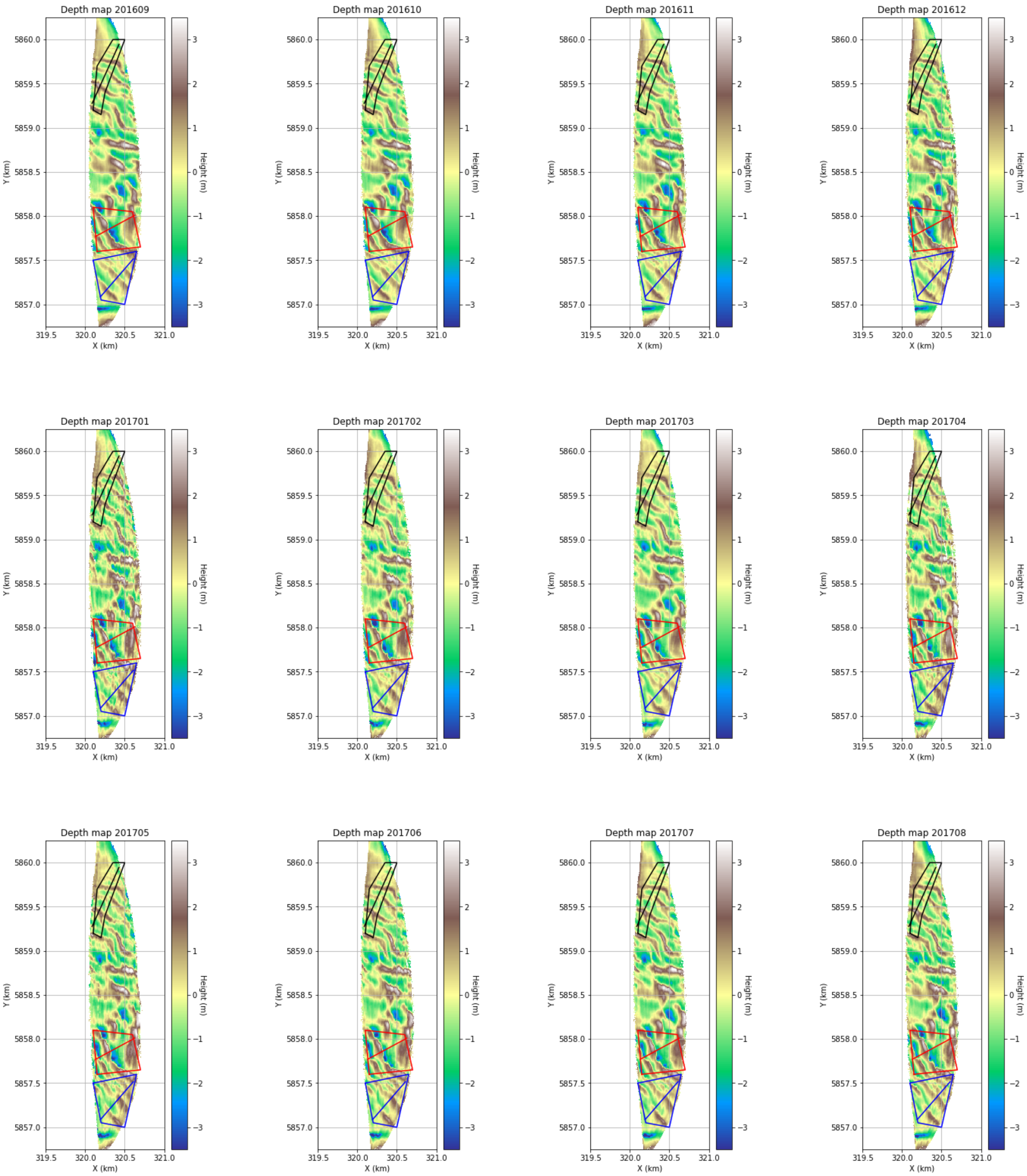

Depth map 201105
Depth map 201106
Depth map 201107
Depth map 201108
Depth map 201109
Depth map 201110
Depth map 201111
Depth map 201112
Depth map 201201
Depth map 201202
Depth map 201203
Depth map 201204
Y (km)
X (km)
Height (m)

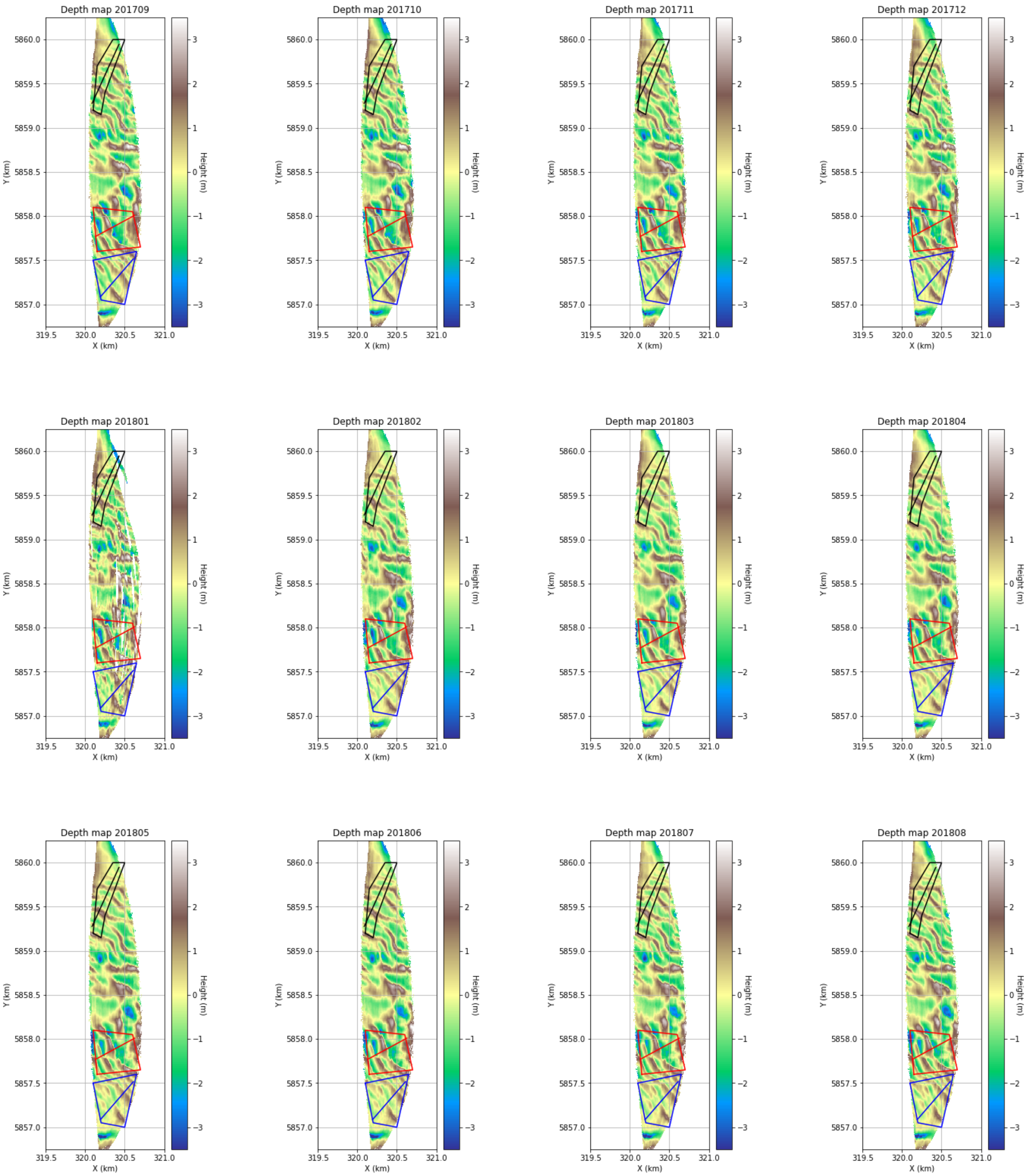

Depth map 201205
Depth map 201206
Depth map 201207
Depth map 201208
Depth map 201209
Depth map 201210
Depth map 201211
Depth map 201212
Depth map 201301
Depth map 201302
Depth map 201303
Depth map 201304
Y (km)
X (km)
Height (m)

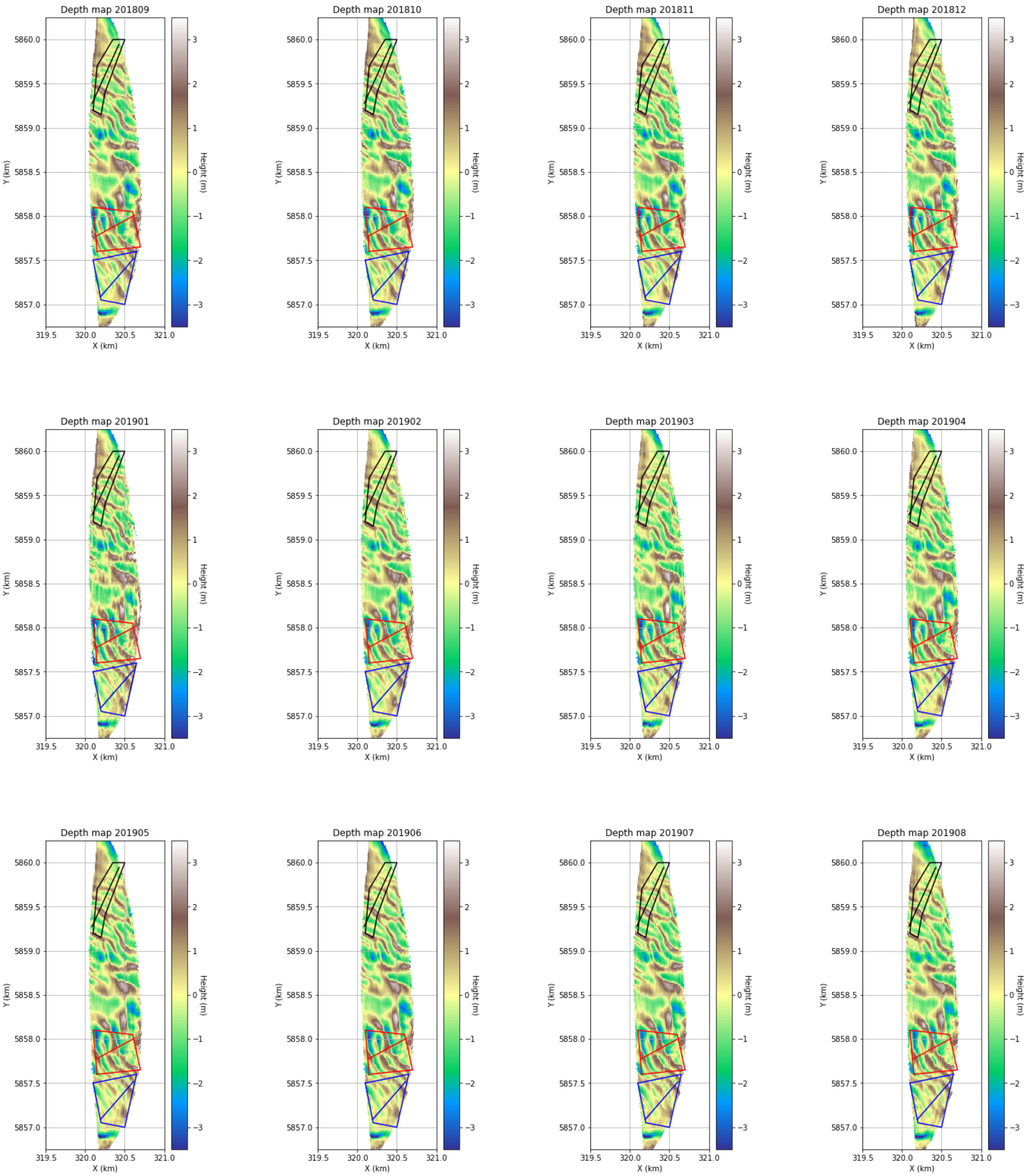

Depth map 201305
Depth map 201306
Depth map 201307
Depth map 201308
Depth map 201309
Depth map 201310
Depth map 201311
Depth map 201312
Depth map 201401
Depth map 201402
Depth map 201403
Depth map 201404
Y (km)
X (km)
Height (m)

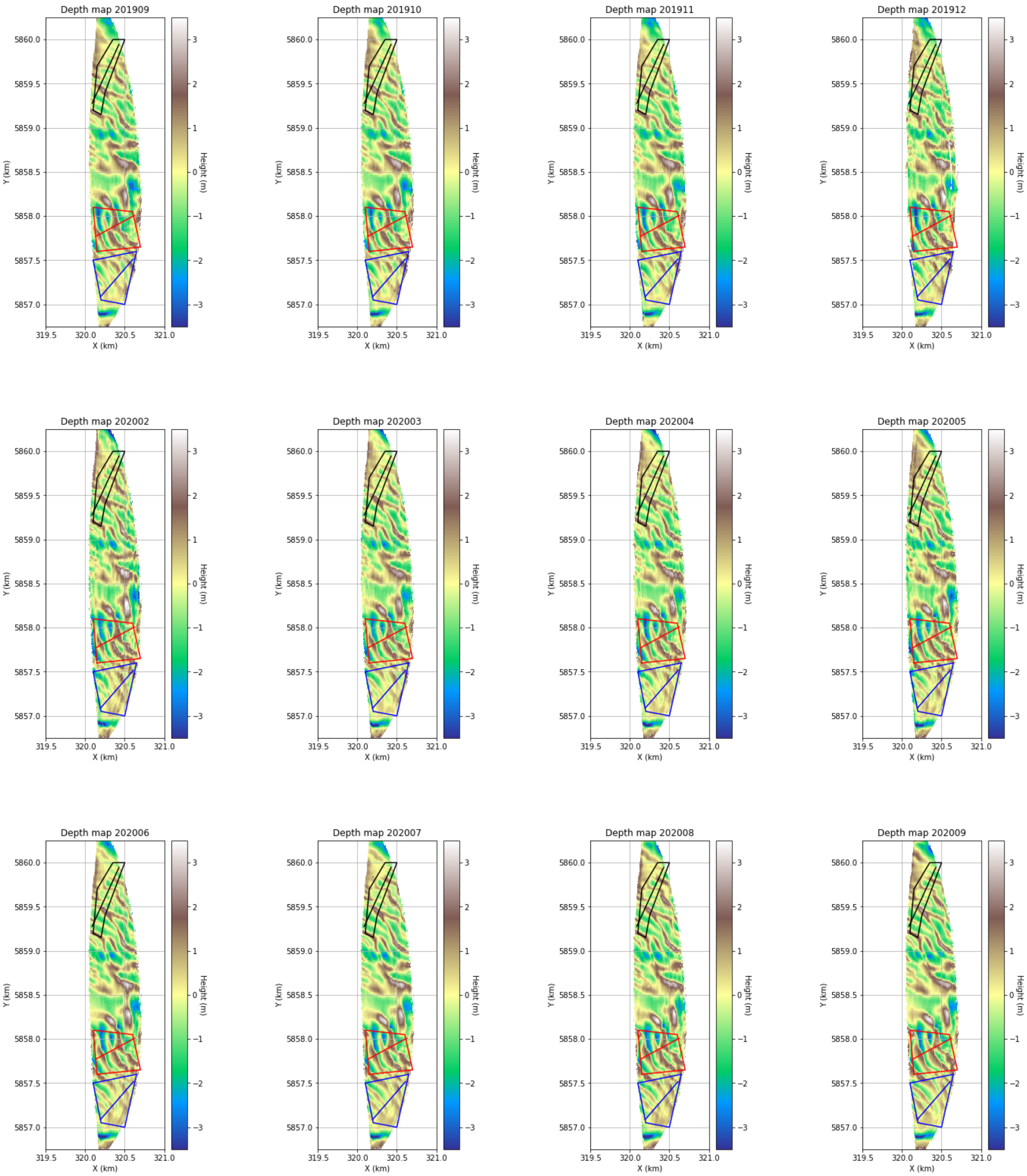

Depth map 201405
Depth map 201406
Depth map 201407
Depth map 201408
Depth map 201409
Depth map 201410
Depth map 201411
Depth map 201412
Depth map 201502
Depth map 201503
Depth map 201504
Depth map 201505
Y (km)
X (km)
Height (m)

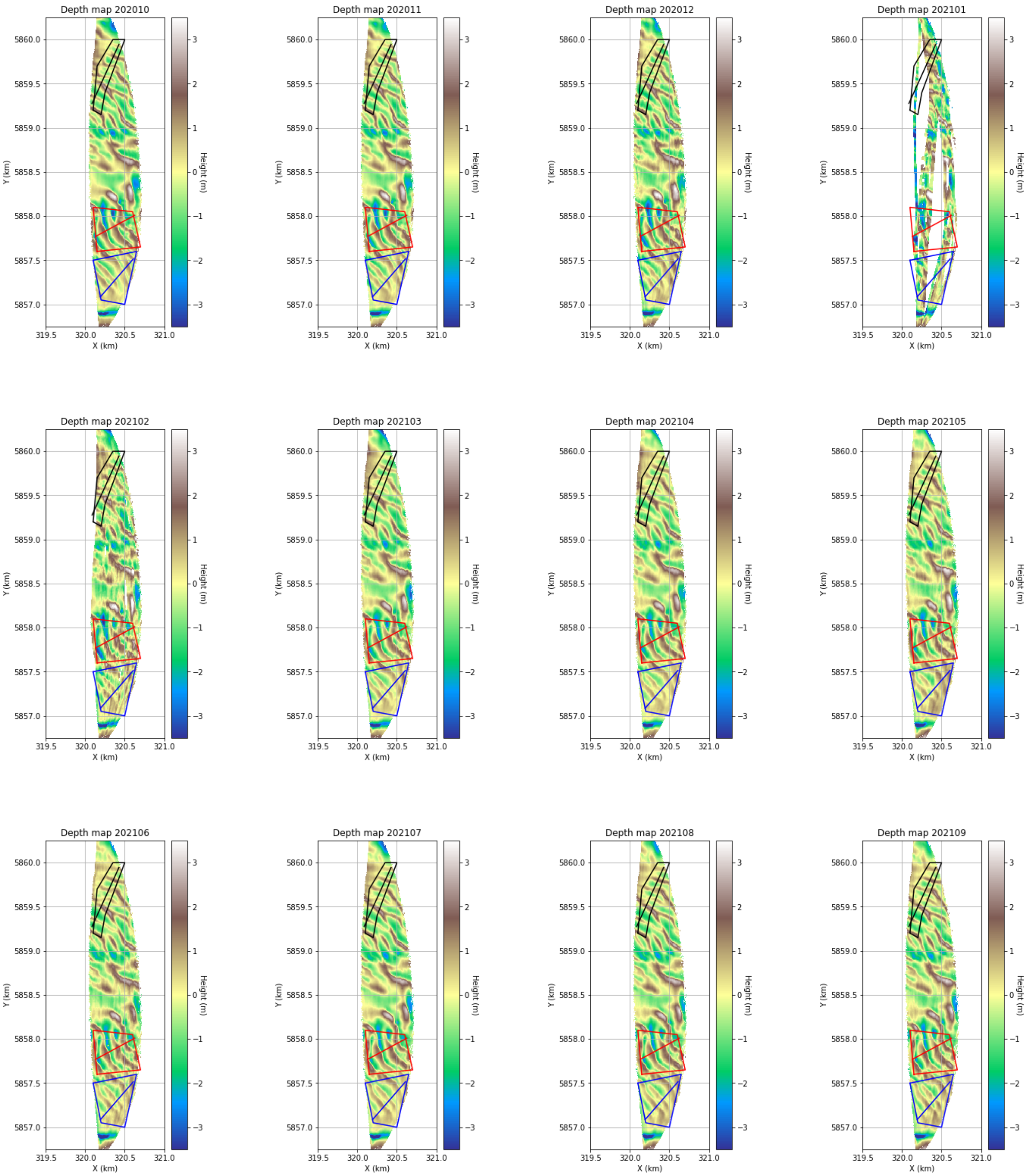

Depth map 201506
Depth map 201507
Depth map 201510
Depth map 201511
Depth map 201512
Depth map 201602
Depth map 201603
Depth map 201604
Depth map 201605
Depth map 201606
Depth map 201607
Depth map 201608
X (km)
Y (km)
Height (m)

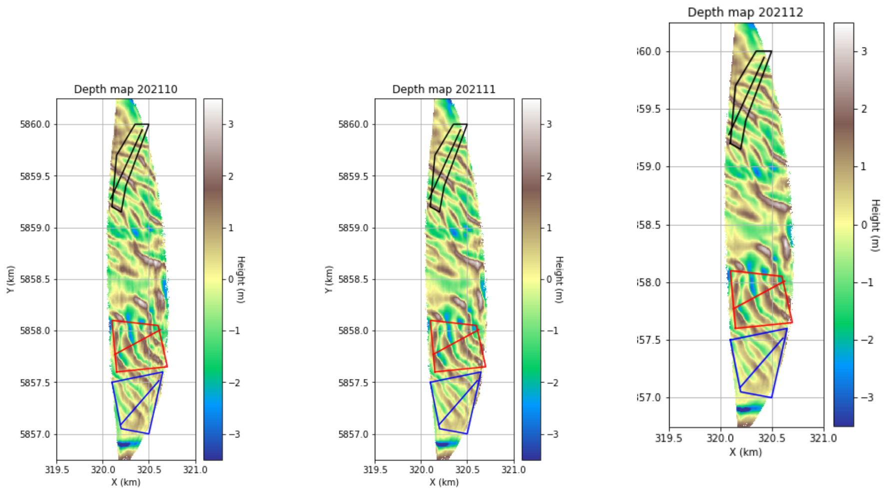

Depth map 201609
Depth map 201610
Depth map 201611
Depth map 201612
Depth map 201701
Depth map 201702
Depth map 201703
Depth map 201704
Depth map 201705
Depth map 201706
Depth map 201707
Depth map 201708
Y (km)
X (km)
Height (m)


Depth map 201709
Depth map 201710
Depth map 201711
Depth map 201712
Depth map 201801
Depth map 201802
Depth map 201803
Depth map 201804
Depth map 201805
Depth map 201806
Depth map 201807
Depth map 201808
Y (km)
X (km)
Height (m)


Depth map 201809
Depth map 201810
Depth map 201811
Depth map 201812
Depth map 201901
Depth map 201902
Depth map 201903
Depth map 201904
Depth map 201905
Depth map 201906
Depth map 201907
Depth map 201908
Y (km)
X (km)
Height (m)
5860.0
5859.5
5859.0
5858.5
5858.0
5857.5
5857.0
319.5
320.0
320.5
321.0
3
2
1
0
−1
−2
−3


Depth map 201909
Depth map 201910
Depth map 201911
Depth map 201912
Depth map 202002
Depth map 202003
Depth map 202004
Depth map 202005
Depth map 202006
Depth map 202007
Depth map 202008
Depth map 202009
Y (km)
X (km)
Height (m)


Depth map 202010
Depth map 202011
Depth map 202012
Depth map 202101
Depth map 202102
Depth map 202103
Depth map 202104
Depth map 202105
Depth map 202106
Depth map 202107
Depth map 202108
Depth map 202109
Y (km)
X (km)
Height (m)
5860.0
5859.5
5859.0
5858.5
5858.0
5857.5
5857.0
319.5
320.0
320.5
321.0
3
2
1
0
−1
−2
−3



B Appendix

Script used to make the depth profiles:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Sep  5 14:42:23 2022

@author: lgeessinck
"""

# %%
import netCDF4 as nc
import numpy as np
import matplotlib.pyplot as plt
import requests
from netCDF4 import Dataset
import glob
import imageio
from os.path import exists
from scipy import stats
import pandas as pd
from scipy.stats import binned_statistic_2d
from datetime import date
import datetime
import scipy as sp
import numpy.ma as ma
import matplotlib
from pathlib import Path
import warnings
import csv
import os
from os import path

r_earth = 6371.000
#%%
#%% convert date to number of days since 1-1-2000
def convert_date(dates):
        format_str = '%d-%m-%Y';
        f_date= datetime.datetime.strptime('1-1-2000',format_str);
        l_date= datetime.datetime.strptime(dates,format_str);
        delta = l_date-f_date;
        number_of_days = delta.days;
```

```
        return number_of_days
# %% convert time to number of seconds since begin of day
def convert_time(dates,times):
    format_str = '%H:%M:%S';
    pt = datetime.datetime.strptime(times,format_str);
    seconds = pt.second + pt.minute*60 + pt.hour*3600;
    total_seconds = seconds-3600
    return total_seconds

#%% open adcp adata
def open_data(data,filename):
    fnheading = filename.replace(".nc","_GPS-Gyro.nc")
    print(fnheading)
    if path.exists(fnheading):
        dataheading = Dataset(fnheading)
        heading = dataheading.variables['HEADING']
        heading=heading[:]
        course = dataheading.variables['COURSE']
        course = course[:]
        depth = data.variables['DEPTH'];
        depth = depth[:];
        heading=heading[depth>0]
        course =course[depth>0]
        depth = depth[depth>0];

    lon = data.variables['LONGITUDE'];
    lon = lon[:];
    lat = data.variables['LATITUDE'];
    lat = lat[:];

    lon = lon[lon>0];
    lat = lat[lat>0];
    time = data.variables['TIME'] ;#time since beginning of day
    time = time[:];
    time = time[time>0];
    day = data.variables['DAY']; #time elapsed in days form 00:00 on January 1st 2000
    day = day[:] ;
    day = day[day>0];
    beam1 = data.variables['D1']
    beam1 = beam1[:]
    beam1 = beam1[beam1>0]
    beam2 = data.variables['D2']
    beam2 = beam2[:]
    beam2 = beam2[beam2>0]
    beam3 = data.variables['D3']
    beam3 = beam3[:]
    beam3 = beam3[beam3>0]
    beam4 = data.variables['D4']
    beam4 = beam4[:]
    beam4 = beam4[beam4>0]


    return lon, lat, time, day, beam1, beam2, beam3, beam4, heading,fnheading


# %%open rws data
def open_rws_data(file):
    file = file.to_numpy();
    dates = file[:,0];
    times = file[:,1];
    rws_depth = file[:,3]/100;
    days = []
    for i in dates:
        delta = convert_date(i);
        days.append(delta);
    seconds = []
    for i,j in zip(dates,times):
        delta = convert_time(i,j);
```

```
        delta -= 3600
        seconds.append(delta);

    return days,seconds,rws_depth

#%% goeie tides
def correct_for_tides(days,rws_depth,seconds,time,day,depth):

    indices = []
    for i in range(len(days)):
        real_depth = [];
        depth_goeie = [];
        if days[i] == day[0]:
            indices.append(i);

        if days[i] != day[0]:
            continue
    return indices #each index is one day
# %%
def tides_correction(indices,time,depth):
    real_depth = [];
    x_min = indices[0] #first measurement of the day
    x_max = indices[-1] #last measurement of the day
    values = rws_depth[x_min:x_max];
    points = seconds[x_min:x_max];
    points = np.array(points);

    x = ma.getdata(time);
    f = sp.interpolate.interp1d(points,values,bounds_error=False);
    y = f(x);
    real_depth = depth-y; #subtracting the tide data
    return real_depth

# %%

def moving_average(depth, window_size):
    i = 0
    j = 0
    moving_averages= np.empty(shape=np.shape(np.transpose(depth)));
    for j in range(len(depth[j])):
        for i in range(len(depth[:,i])):
            if depth[i,j] != np.nan:
                window= depth[i-window_size:i+window_size,j-window_size:j+window_size];
                with warnings.catch_warnings():
                  warnings.simplefilter("ignore",category=RuntimeWarning);
                  window_average = np.nanmean(window);
                  moving_averages[j,i] = window_average;
                  i += 1
        j += 1
        print(j)

    return moving_averages
# %%
def convert_lat(lat,lon):
    lat = 110.574*lat;
    return lat
# %%
def convert_lon(lat,lon):
    lon = (np.cos(np.deg2rad(lat))*r_earth * np.pi)/180 * lon;
    return lon

#%% Make depth profile
def depth_profile(lon,lat,depth):

    lon_cat = np.concatenate(lon);
    lat_cat = np.concatenate(lat);
    lon_carth = convert_lon(lat_cat,lon_cat);
    lat_carth = convert_lat(lat_cat,lon_cat);
```

```
    depth = np.concatenate(depth)
    bins = binned_statistic_2d(lon_carth,lat_carth,depth, statistic= 'median', bins=[3462/
    grid_size, 4257/grid_size],range= [[319.060791015625,322.5225830078125],
    [5856.43017578125,5860.68701171875]]);
    avg = (moving_average((bins[0]),45));
    return bins, avg

#%% save data
def save_data(bins,avg):
    depth=bins[0]
    path = '/home/lgeessinck/mscproject/depth_profile/data/depth_profile_{}.csv'.format(date
    )
    np.savetxt(path, depth)
    x=bins[1]
    path = '/home/lgeessinck/mscproject/depth_profile/data/x_profile_{}.csv'.format(date)
    np.savetxt(path, x)
    y=bins[2]
    path = '/home/lgeessinck/mscproject/depth_profile/data/y_profile_{}.csv'.format(date)
    np.savetxt(path, y)
    path = '/home/lgeessinck/mscproject/depth_profile/data/avg_profile_{}.csv'.format(date)
    np.savetxt(path, avg)
#%%
days = []
seconds = []
rws_depth = []

rws = pd.read_csv('/home/lgeessinck/mscproject/waterstanden2.csv', delimiter = ';');
days_temp,seconds_temp,rws_depth_temp = open_rws_data(rws);
days.append(days_temp)
seconds.append(seconds_temp)
rws_depth.append(rws_depth_temp)

rws = pd.read_csv('/home/lgeessinck/mscproject/2019rws.csv', delimiter = ';');
days_temp,seconds_temp,rws_depth_temp = open_rws_data(rws);
days.append(days_temp)
seconds.append(seconds_temp)
rws_depth.append(rws_depth_temp)
#%%
rws = pd.read_csv('/home/lgeessinck/mscproject/2015rws.csv', delimiter = ';');
days_temp,seconds_temp,rws_depth_temp = open_rws_data(rws);
days.append(days_temp)
seconds.append(seconds_temp)
rws_depth.append(rws_depth_temp)
rws = pd.read_csv('/home/lgeessinck/mscproject/2018rws.csv', delimiter = ';');
days_temp,seconds_temp,rws_depth_temp = open_rws_data(rws);
days.append(days_temp)
seconds.append(seconds_temp)
rws_depth.append(rws_depth_temp)
rws = pd.read_csv('/home/lgeessinck/mscproject/20162017rws.csv', delimiter = ';');
days_temp,seconds_temp,rws_depth_temp = open_rws_data(rws);
days.append(days_temp)
seconds.append(seconds_temp)
rws_depth.append(rws_depth_temp)
rws = pd.read_csv('/home/lgeessinck/mscproject/2016rws.csv', delimiter = ';');
days_temp,seconds_temp,rws_depth_temp = open_rws_data(rws);
days.append(days_temp)
seconds.append(seconds_temp)
rws_depth.append(rws_depth_temp)
rws = pd.read_csv('/home/lgeessinck/mscproject/20102012rws.csv', delimiter = ';');
days_temp,seconds_temp,rws_depth_temp = open_rws_data(rws);
days.append(days_temp)
seconds.append(seconds_temp)
rws_depth.append(rws_depth_temp)
rws = pd.read_csv('/home/lgeessinck/mscproject/2013rws.csv', delimiter = ';');
days_temp,seconds_temp,rws_depth_temp = open_rws_data(rws);
days.append(days_temp)
seconds.append(seconds_temp)
rws_depth.append(rws_depth_temp)
```

```
rws = pd.read_csv('/home/lgeessinck/mscproject/2014rws.csv', delimiter = ';');
days_temp,seconds_temp,rws_depth_temp = open_rws_data(rws);
days.append(days_temp)
seconds.append(seconds_temp)
rws_depth.append(rws_depth_temp)

days = np.concatenate(days)
seconds = np.concatenate(seconds)
rws_depth = np.concatenate(rws_depth)

rws_depth = np.delete(rws_depth, np.argwhere(rws_depth>2));
# %% open heading data
def convert_beams(lon,lat,beam1,beam2,beam3,beam4,heading):
    x = convert_lon(lat, lon)
    y = convert_lat(lat,lon)

    x_beam1 = x + np.cos(np.deg2rad(70))*beam1/1000*np.sin(np.deg2rad(heading-45))
    y_beam1 = y + np.cos(np.deg2rad(70))*beam1/1000*np.cos(np.deg2rad(heading-45))

    x_beam2 = x + np.cos(np.deg2rad(70))*beam2/1000*np.sin(np.deg2rad(heading + 135))
    y_beam2 = y + np.cos(np.deg2rad(70))*beam2/1000*np.cos(np.deg2rad(heading + 135))

    x_beam3 = x + np.cos(np.deg2rad(70))*beam3/1000*np.sin(np.deg2rad(heading + 45))
    y_beam3 = y + np.cos(np.deg2rad(70))*beam3/1000*np.cos(np.deg2rad(heading + 45))

    x_beam4 = x + np.cos(np.deg2rad(70))*beam4/1000*np.sin(np.deg2rad(heading - 135))
    y_beam4 = y + np.cos(np.deg2rad(70))*beam4/1000*np.cos(np.deg2rad(heading - 135))

    beam1 = np.sin(np.deg2rad(70))*beam1
    beam2 = np.sin(np.deg2rad(70))*beam2
    beam3 = np.sin(np.deg2rad(70))*beam3
    beam4 = np.sin(np.deg2rad(70))*beam4

    beams = np.concatenate([beam1,beam2,beam3,beam4])
    beamsx = np.concatenate([x_beam1, x_beam2, x_beam3, x_beam4])
    beamsy = np.concatenate([y_beam1, y_beam2, y_beam3, y_beam4])


    return beams,beamsx,beamsy

# %%
grid_size=5
datadir = '/home/jvandermolen/data_out/TESO/daily/'
months = ['01','02','03','04','05','06','07','08','09','10','11','12']

years = ['2010','2011','2012','2013', '2014', '2015', '2016', '2017', '2018', '2019', '2020'
    , '2021']

gyro = []
for j in years:
    for i in months:
        date = str(j) + str(i);
        print(date)
        filenames = glob.glob(datadir + '/*/*{}*.nc'.format(date));

        for i, filename in enumerate(filenames):
            if 'Gyro' in filename:
                del filenames[i]
                gyro.append(filename)


        if filenames != [] :
            for i, filename in enumerate(filenames):
                fnheading = filename.replace(".nc","_GPS-Gyro.nc")
                print(fnheading)
                if not os.path.isfile(fnheading):
                    del filenames[i]

```

```
        filenames.sort()
        gyro.sort()
        filenames2 = filenames
        filenames1 = []
        for i, filename in enumerate(filenames2):
            data = Dataset(filename);
            fnheading = filename.replace(".nc","_GPS-Gyro.nc")
            print(fnheading)

            if path.exists(fnheading):
                filenames1.append(filename)

        lon = np.empty(shape=(len(filenames1)), dtype='object');
        lat = np.empty(shape=(len(filenames1)), dtype='object');
        depth = np.empty(shape=(len(filenames1)), dtype='object');
        time = np.empty(shape=(len(filenames1)), dtype='object');
        day = np.empty(shape=(len(filenames1)), dtype='object');
        indices = np.empty(shape=(len(filenames1)),dtype='object');
        real_depth = np.empty(shape=(len(filenames1)),dtype='object');
        beam1 = np.empty(shape=(len(filenames1)), dtype='object');
        beam2 = np.empty(shape=(len(filenames1)), dtype='object');
        beam3 = np.empty(shape=(len(filenames1)), dtype='object');
        beam4 = np.empty(shape=(len(filenames1)), dtype='object');
        real_depth_beam1 = np.empty(shape=(len(filenames1)),dtype='object');
        real_depth_beam2 = np.empty(shape=(len(filenames1)),dtype='object');
        real_depth_beam3 = np.empty(shape=(len(filenames1)),dtype='object');
        real_depth_beam4 = np.empty(shape=(len(filenames1)),dtype='object');
        heading = np.empty(shape=(len(filenames1)),dtype='object');
        course = np.empty(shape=(len(filenames1)),dtype='object');

        for i, filename in enumerate(filenames1):
            print(filename)
            data = Dataset(filename);
            fnheading = filename.replace(".nc","_GPS-Gyro.nc")
            if path.exists(fnheading):
                print(fnheading)

                lon[i], lat[i], time[i], day[i], beam1[i],beam2[i],beam3[i],beam4[i], heading[i],fnheading= open_data(data,filename);

                indices[i] = correct_for_tides(days,rws_depth, seconds, time[i], day[i], beam1[i]);

                real_depth_beam1[i] = tides_correction(indices[i],time[i],beam1[i]);
                real_depth_beam2[i] = tides_correction(indices[i],time[i],beam2[i]);
                real_depth_beam3[i] = tides_correction(indices[i],time[i],beam3[i]);
                real_depth_beam4[i] = tides_correction(indices[i],time[i],beam4[i]);


        lon_cat = np.concatenate(lon);
        lat_cat = np.concatenate(lat);
        heading_cat = np.concatenate(heading);
        lon_cat = np.delete(lon_cat,np.argwhere(heading_cat<0))
        lat_cat = np.delete(lat_cat,np.argwhere(heading_cat<0))


        real_depth_beam1 = np.concatenate(real_depth_beam1);
        real_depth_beam1 = np.delete(real_depth_beam1,np.argwhere(heading_cat<0))
        real_depth_beam2 = np.concatenate(real_depth_beam2);
        real_depth_beam2 = np.delete(real_depth_beam2,np.argwhere(heading_cat<0))
        real_depth_beam3 = np.concatenate(real_depth_beam3);
        real_depth_beam3 = np.delete(real_depth_beam3,np.argwhere(heading_cat<0))
        real_depth_beam4 = np.concatenate(real_depth_beam4);
        real_depth_beam4 = np.delete(real_depth_beam4,np.argwhere(heading_cat<0))
        heading_cat = np.delete(heading_cat, np.argwhere(heading_cat<0));

        beams,beamsx,beamsy = convert_beams(lon_cat,lat_cat,real_depth_beam1, real_depth_beam2,real_depth_beam3,real_depth_beam4,heading_cat)
```

```
            bins = binned_statistic_2d(beamsx,beamsy,beams, statistic= 'median', bins=[3462/grid_size , 4257/grid_size],range= [[319.060791015625,322.5225830078125],[5856.43017578125,5860.68701171875]]);

            avg = (moving_average((bins[0]),45));
            fig=plt.figure(figsize=(3,8))
            plt.pcolormesh(bins[1],bins[2],np.transpose(bins[0]),cmap='terrain')
            plt.colorbar()
            plt.grid()
            plt.xlim(319.5,321)
            plt.title(f'Depth profile {date}')
            plt.savefig(f"/home/lgeessinck/mscproject/analyze/hdtxdepth_profile_plot_{date}5month.png".format(date))
            plt.show()

            sand_waves = np.transpose(bins[0])-avg

            depth=bins[0]
            pathd = '/home/lgeessinck/mscproject/depth_profile/data/4beams/hdtxdepth_profile_{}5.csv'.format(date)
            np.savetxt(pathd, depth)
            x=bins[1]
            pathx = '/home/lgeessinck/mscproject/depth_profile/data/4beams/hdtxx_profile_{}5.csv'.format(date)
            np.savetxt(pathx, x)
            y=bins[2]
            pathy = '/home/lgeessinck/mscproject/depth_profile/data/4beams/hdtxy_profile_{}5.csv'.format(date)
            np.savetxt(pathy, y)
            patha = '/home/lgeessinck/mscproject/depth_profile/data/4beams/hdtxavg_profile_{}5.csv'.format(date)
            np.savetxt(patha, avg)
            paths = '/home/lgeessinck/mscproject/depth_profile/data/4beams/hdtxsand_waves_{}5.csv'.format(date)
            np.savetxt(paths, sand_waves)
```

Script used for analysis:

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Jan  7 13:46:55 2023

@author: lgeessinck
"""
from os import path
import numpy as np
from shapely.geometry import asShape
from shapely.geometry import Polygon
from shapely.geometry import Point
import matplotlib.pyplot as plt
from scipy.signal import find_peaks

months = ['01','02','03', '04', '05', '06', '07','08', '09', '10', '11', '12']
years = ['2010','2011','2012','2013','2014','2015','2016','2017','2018','2019','2020','2021']
dates = []
count = []
mask_out = np.zeros((851,692))
for j in years:
    for i in months:
        date = str(j) + str(i);
x=np.loadtxt('/home/lgeessinck/mscproject/depth_profile/data/4beams/hdtxx_profile_{}5.csv'.format(date))
y=np.loadtxt('/home/lgeessinck/mscproject/depth_profile/data/4beams/hdtxy_profile_{}5.csv'.format(date))
```

```
#%% define polygons
coord = [[320,5857.50],[320.65,5857.6],[320.5,5857],[320.1,5857.05]]
polygon1 = Polygon(coord)
coord.append(coord[0])
xs1, ys1 = zip(*coord) #create lists of x and y values

coord = [[320,5857.6],[320.7,5857.65],[320.5,5858.05],[320,5858.1]]
polygon2 = Polygon(coord)
coord.append(coord[0])
xs2, ys2 = zip(*coord) #create lists of x and y values

coord =
    [[320,5859.2],[320.2,5859.15],[320.25,5859.4],[320.5,5860],[320.35,5860],[320.15,5859.7]]

polygon3 = Polygon(coord)
coord.append(coord[0])
xs3, ys3 = zip(*coord) #create lists of x and y values

plt.figure(figsize=(4,8))
plt.pcolormesh(x,y,mask_out,cmap= 'nipy_spectral')

cbar= plt.colorbar()
cbar.set_label('Number of months passed through grid cell', rotation=270)
plt.grid()
plt.xlim(319.5,321)
plt.ylim(5856.75,5860.25)
#plt.plot(xs1,ys1)
#plt.plot(xs2,ys2,'r')
#plt.plot(xs3,ys3,'black')
plt.xlabel('X (km)')
plt.ylabel('Y (km)')

plt.show()

#%% look for points wheteher they are inside the polygon or not
def check_polygon(polygon):
    polygon_out = np.zeros((851,692))
    polygon_out = polygon_out*np.nan
    for i in range(len(x)):
        for j in range(len(y)):
            point = Point(x[i],y[j])
            print(point)
            test = polygon.contains(point)
            print(test)
            if test == True:
                polygon_out[j,i] = 1
    return polygon_out

#%% make polygons

polygon1_out = check_polygon(polygon1)
polygon2_out = check_polygon(polygon2)
polygon3_out = check_polygon(polygon3)
#%%count number of NaNs
months = ['01','02','03', '04', '05', '06', '07','08', '09', '10', '11', '12']
years = ['2010','2011','2012','2013','2014','2015','2016','2017','2018','2019','2020','2021'
    ]
dates = []
count = []
mask_out = np.zeros((851,692))
for j in years:
    for i in months:
        date = str(j) + str(i);
        if path.exists(('/home/lgeessinck/mscproject/depth_profile/data/4beams/
    hdtxsand_waves_{}5smoothed.csv'.format(date))):
            sand_waves = np.loadtxt('/home/lgeessinck/mscproject/depth_profile/data/4beams/
    hdtxsand_waves_{}5smoothed.csv'.format(date))
```

```
            dates.append(date)
            sand_waves=sand_waves*mask90
            count_temp = np.count_nonzero(~np.isnan(sand_waves))
            count.append(count_temp)

#%%generate all monhts
datums = [] #maanden die wel goed zijn gegaan
for i in dates:
    print(i)
    datum= float(i)
    datums.append(datum)
alle_maanden = []
all_dates = []
for j in years: # alle maanden
    for i in months:
        date = str(j) + str(i);
        dates = float(date)
        all_dates.append(dates)
        alle_maanden.append(date)
#%% add months
def add_months(value):
    index = np.where(np.isin(all_dates,datums))
#test = np.stack(transy_all, axis=0)
    all_value = np.zeros((144))
    all_value[:] = np.NaN

    count = []
    for i,maand in enumerate(all_value):
        print(i)
        print(maand)
        count_temp = np.count_nonzero(~np.isnan(maand))
        count.append(count_temp)

    for i in index:
        for j in range(len(value)):
            all_value[i] = value
            j += 1
    return all_value


#%% make mask
months = ['01','02','03', '04', '05', '06', '07','08', '09', '10', '11', '12']
years = ['2010','2011','2012','2013','2014','2015','2016','2017','2018','2019','2020','2021'
    ]
dates = []
all_mask = []
mask_out = np.zeros((851,692))
for j in years:
    for i in months:
        date = str(j) + str(i);
        if path.exists(('/home/lgeessinck/mscproject/depth_profile/data/4beams/
    hdtxsand_waves_{}5.csv'.format(date))):
            sand_waves = np.loadtxt('/home/lgeessinck/mscproject/depth_profile/data/4beams/
    hdtxsand_waves_{}5.csv'.format(date))
            count_temp = np.count_nonzero(~np.isnan(sand_waves))
            if count_temp>80000:
                indices = (np.argwhere(np.isnan(sand_waves)))
                mask = np.ones((851,692))
                print(np.shape(sand_waves))
                for m,n in indices:
               #     print(m,n)
                    mask[m,n] = np.nan
                mask_temp = mask
                all_mask.append(mask_temp)
                for u in range(len(mask[0])):
                    for v in range(len(mask[:,0])):
                        #print(u)
                        if mask_temp[v,u] ==1:
```

```
                            mask_out[v,u] = mask_out[v,u] + 1
                i =+ 1
 #%%
mask85 = np.ones((np.shape(mask_out)))
for u in range(len(mask[0])):
    for v in range(len(mask[:,0])):
        if mask_out[v,u] < (0.85*mask_out.max()):
            mask85[v,u] = np.nan
        if mask_out[v,u] == (0.85*mask_out.max()):
            mask85[v,u] = 1
        print(mask85[v,u])

mask100 = np.ones((np.shape(mask_out)))
for u in range(len(mask[0])):
    for v in range(len(mask[:,0])):
        if mask_out[v,u] < (1*mask_out.max()):
            mask100[v,u] = np.nan
        if mask_out[v,u] == (1*mask_out.max()):
            mask100[v,u] = 1
        print(mask100[v,u])

mask90 = np.ones((np.shape(mask_out)))
for u in range(len(mask[0])):
    for v in range(len(mask[:,0])):
        if mask_out[v,u] < (0.90*mask_out.max()):
            mask90[v,u] = np.nan
        if mask_out[v,u] == (0.90*mask_out.max()):
            mask90[v,u] = 1
        print(mask90[v,u])


#%% calculate standarddeviation per polygon
def standard_deviation(polygon1):
    std=[]
    dates=[]
    count=[]
    for j in years:
        for k in months:
            date = str(j) + str(k)
            if path.exists(('/home/lgeessinck/mscproject/depth_profile/data/4beams/hdtxsand_waves_{}5smoothed.csv'.format(date))
       ):
                sand_waves = np.loadtxt('/home/lgeessinck/mscproject/depth_profile/data/4beams/hdtxsand_waves_{}5smoothed.csv'.format(date))
                count_temp = np.count_nonzero(~np.isnan(sand_waves))
                sand_waves=sand_waves*mask90
                if count_temp>70000:


                    count.append(count_temp)
                    print(date)
                    dates.append(date)

                    area1= polygon1*sand_waves
                    std_temp = np.nanstd(area1)
                    print(std_temp)
                    std.append(std_temp)
                    plt.pcolormesh(area1)
    return std,dates
#%%
def getEquidistantPoints(p1, p2, parts):
     eq = list(zip(np.linspace(p1[0], p2[0], parts+1),np.linspace(p1[1], p2[1], parts+1)))
     eq = np.array(eq)
     xline = eq[:,0]
     yline = eq[:,1]
     return xline,yline
# %%
def make_transect2(xline,yline,sand_waves):
```

```

    delta_x = (x.max() - x.min()) / len(sand_waves[0])
    trans_x = (xline-x.min()) / delta_x
    trans_x = trans_x.astype(int)

    delta_y = (y.max() - y.min()) / len(sand_waves)
    trans_y = (yline-y.min()) / delta_y
    trans_y = trans_y.astype(int)

    ytransect = ()
    for i,j in zip(trans_y,trans_x):

        wave = sand_waves[i,j]
        ytransect = np.append(ytransect,wave)
        xtransect = np.sqrt((xline-xline.min())*(xline-xline.min())+(yline-yline.min())*(yline-yline.min()))
    return ytransect,xtransect
#%%
# Initialize an empty list to store moving averages
def moving_average1D(arr,window_size):
    moving_averages = []
    i = 0
    # Loop through the array to consider
    # every window of size 3
    while i < len(arr) - window_size + 1:

        # Store elements from i to i+window_size
        # in list to get the current window
        window = arr[i-window_size : i + window_size]

        # Calculate the average of current window
        window_average = round(sum(window) / window_size, 2)

        # Store the average of current
        # window in moving average list
        moving_averages.append(window_average)

        # Shift window to right by one position
        i += 1
        x = np.array(moving_averages)


    return x
#%%
def moving_average(depth, window_size):
    i = 0
    j = 0
    moving_averages= np.empty(shape=np.shape(np.transpose(depth)));
    for j in range(len(depth[j])):
        for i in range(len(depth[:,i])):
            if depth[i,j] != np.nan:
                window= depth[i-window_size:i+window_size,j-window_size:j+window_size];
                with warnings.catch_warnings():
                  warnings.simplefilter("ignore",category=RuntimeWarning);
                  window_average = np.nanmean(window);
                  moving_averages[j,i] = window_average;
                  i += 1
        j += 1
        print(j)

    return moving_averages
#%%
def running_mean(data, window_size):
   # depth_mean= data.mean(axis=0)
  #  arr = np.array(depth_mean)
    arr = data
    i = 0
    moving_averages = []
```

```
    for i in range(len(arr)):
        window = arr[i - window_size : i + window_size]
        window_average = np.nanmean(window)
        moving_averages.append(window_average)
        i += 1

    return moving_averages
#%%
def compute_wavelength(start1,end1):
    for j in years:
        for k in months:
            date = str(j) + str(k)
            if path.exists(('/home/lgeessinck/mscproject/depth_profile/data/4beams/hdtxsand_waves_{}5.csv'.format(date))
       ):
                sand_waves = np.loadtxt('/home/lgeessinck/mscproject/depth_profile/data/4beams/hdtxsand_waves_{}5smoothed.csv'.format(date))

                x = np.loadtxt('/home/lgeessinck/mscproject/depth_profile/data/4beams/hdtxx_profile_2021085.csv')
                y = np.loadtxt('/home/lgeessinck/mscproject/depth_profile/data/4beams/hdtxy_profile_2021085.csv')
                count_temp = np.count_nonzero(~np.isnan(sand_waves))
                sand_waves=sand_waves*mask90
                if count_temp>70000:
                    xline, yline = list(getEquidistantPoints(start1,end1,100))
                    transy2,transx1 = make_transect2(xline,yline,sand_waves)
                    transy1 = transy2 * -1
                 #    transx.append(transx1)
                  #  transy.append(transy1)
                    moving = running_mean(transy1,2)
                    moving = np.array(moving)
                    peak_movex = find_peaks(moving,distance=10)[0]
                    peak_movey = moving[peak_movex]
                    peak_movex = peak_movex/100*transx1.max()

                    trough_movex = find_peaks(-moving,distance=10)[0]

                    trough_movey = moving[trough_movex]
                    trough_movex = trough_movex/100*transx1.max()

                    diff = np.diff(peak_movex)
                    diff_trough = np.diff(trough_movex)
                    mean = np.mean(diff)
                    std = np.std(diff)
                    mean_wl.append(mean)
                    std_wl.append(std)
                    dates.append(date)
                    moving_all.append(moving)
                    asym=[]
                    print(date)
                    for i,u in zip(range(len(diff)-1),range(len(diff_trough)-1)):
                            if peak_movex[i] > trough_movex[i]:
                                AS = (trough_movex[u+1]-peak_movex[i])/(peak_movex[i+1]-peak_movex[i])
                                if AS > 0 and AS <1:
                                    asym.append(AS)
                            if trough_movex[i]>peak_movex[i]:
                                AS = (trough_movex[u]-peak_movex[i])/(peak_movex[i+1]-peak_movex[i])
                                print(AS)
                                if AS > 0 and AS <1:
                                    asym.append(AS)
                    mean_AS=np.mean(asym)
                    num_peaks.append(len(diff))
                    std_AS=np.std(asym)
                    asym_mean.append(mean_AS)
```

```
                    asym_std.append(std_AS)
                    print(mean_AS)
                    print(std_AS)
    return moving,transx1,peak_movex,peak_movey,trough_movex,trough_movey

#%%
mean_wl=[]
dates=[]
num_peaks=[]
std_wl = []
asym_mean=[]
asym_std=[]
moving_all=[]
start = (320.4,5857.9)
asym=[]
end1 = (start[0]+0.3*np.cos(0.45) , start[1]+0.3*np.sin(0.45))
start1 = (start[0]-0.4*np.cos(0.45) , start[1]-0.4*np.sin(0.45))

start2=(320.4,5857.3)

end2 = (start2[0]+0.3*np.cos(0.8) , start2[1]+0.3*np.sin(0.8))
start2 = (start2[0]-0.4*np.cos(0.8) , start2[1]-0.4*np.sin(0.8))
xline2,yline2 = list(getEquidistantPoints(start2,end2,100))

start3=(320.2,5859.5)
end3 = (start3[0]+0.5*np.cos(1) , start3[1]+0.5*np.sin(1))
start3 = (start3[0]-0.3*np.cos(1) , start3[1]-0.3*np.sin(1))
xline3,yline3 = list(getEquidistantPoints(start3,end3,100))

#moving1,transx1,peaksx1,peaksy1,troughsx1,troughsy1=compute_wavelength(start1,end1)
#moving2,transx2,peaksx2,peaksy2,troughsx2,troughsy2=compute_wavelength(start2,end2)
moving3,transx3,peaksx3,peaksy3,troughsx3,troughsy3=compute_wavelength(start3,end3)

all_asym_mean=add_months(asym_mean)
all_asym_std=add_months(asym_std)
all_num_peaks=add_months(num_peaks)

all_mean = add_months(mean_wl)
all_std = add_months(std_wl)
#%%
fig,ax=plt.subplots()
#plt.plot(alle_maanden,all_asym_mean,c='orange')
plt.xticks(np.arange(0, len(alle_maanden)+1, 12),rotation=70)
plt.grid()
#plt.plot(alle_maanden,all_asym_std)
ax.errorbar(alle_maanden,all_asym_mean,all_asym_std,c='r',ecolor='pink')
ax.set_xlabel("Time (months)")
ax.set_ylabel("Mean and standard deviation",color="red",fontsize=14)


# twin object for two different y-axis on the sample plot
ax2=ax.twinx()
# make a plot with different y-axis using second axis object
ax2.plot(alle_maanden,all_num_peaks,c='blue')
ax2.set_ylabel("Number of sand waves",color="blue",fontsize=14)
ax2.set_ylim(-6,8)

plt.title('Transect 3 Asymmetry')
plt.show()
#%%
fig,ax=plt.subplots()
#plt.plot(alle_maanden,all_mean,c='orange')
plt.xticks(np.arange(0, len(alle_maanden)+1, 12),rotation=70)
plt.grid()
#plt.plot(alle_maanden,all_asym_std)
ax.errorbar(alle_maanden,all_mean,all_std,c='r',ecolor='pink')
ax.set_xlabel("Time (months)")
ax.set_ylabel("Mean and standard deviation (km)",color="red",fontsize=14)
```

```


# twin object for two different y-axis on the sample plot
ax2=ax.twinx()
# make a plot with different y-axis using second axis object
ax2.plot(alle_maanden,all_num_peaks,c='blue')
ax2.set_ylabel("Number of sand waves",color="blue",fontsize=14)
ax2.set_ylim(-6,8)

plt.title('Transect 3 Wavelength')
plt.show()
#%%
start = (320.4,5857.9)
plt.figure(figsize=(3,8))
plt.pcolormesh(x,y,-sand_waves*mask90,cmap= 'terrain',vmin=-3, vmax= 3)
plt.colorbar()
plt.grid()
plt.xlim(319.5,321)
plt.ylim(5856.5,5860.5)
plt.xlabel('X (km)')
plt.ylabel('Y (km)')
plt.title(f'Depth profile {date}')
plt.plot(xline,yline)
plt.plot(xline2,yline2)
plt.plot(xline3,yline3)
#%%
std1,dates = standard_deviation(polygon1_out)
std2,dates = standard_deviation(polygon2_out)
std3,dates = standard_deviation(polygon3_out)

datums = [] #maanden die wel goed zijn gegaan
for i in dates:
    print(i)
    datum= float(i)
    datums.append(datum)
alle_maanden = []
all_dates = []
for j in years: # alle maanden
    for i in months:
        date = str(j) + str(i);
        dates = float(date)
        all_dates.append(dates)
        alle_maanden.append(date)

all_std1=add_months(std1)
all_std2=add_months(std2)
all_std3=add_months(std3)



#%%plot number of nans
all_count = add_months(count)
plt.plot(alle_maanden,all_count)
plt.xticks(np.arange(0, len(alle_maanden)+1, 12),rotation = 90)
plt.grid()
plt.axhline(70000,c='r')
plt.xlabel('Time (months)')
plt.ylabel('Number of filled grid cells')

#%% plot masks
plt.figure(figsize=(4,8))
plt.pcolormesh(x,y,mask100)

plt.grid()
plt.xlim(319.5,321)
plt.ylim(5856.75,5860.25)

plt.xlabel('X (km)')
```

```
plt.ylabel('Y (km)')
plt.title('Mask100')

plt.show()
#%%plt std
plt.plot(alle_maanden,all_std3*np.sqrt(2), label = '3', c= 'black')
plt.plot(alle_maanden,all_std2*np.sqrt(2), label = '2', c = 'r')
plt.plot(alle_maanden,all_std1*np.sqrt(2), label = '1', c= 'blue')
plt.xticks(np.arange(0, len(alle_maanden)+1, 12),rotation = 90)
plt.grid()
plt.legend()
plt.xlabel('Time (months)')
plt.ylabel('Height (m)')
plt.title('Amplitude of sand waves in area 1, 2 and 3')

#%%
def add_months2D(value):
    index = np.where(np.isin(all_dates,datums))
#test = np.stack(transy_all, axis=0)
    all_value = np.zeros((144,101))
    all_value[:] = np.NaN

    count = []
    for i,maand in enumerate(all_value):
        print(i)
        print(maand)
        count_temp = np.count_nonzero(~np.isnan(maand))
        count.append(count_temp)

    for i in index:
        for j in range(len(count)):
            all_value[i] = value
            j += 1
    return all_value

#%%

all_moving3= add_months2D(moving_all)

plt.figure()
plt.pcolormesh(transx3,alle_maanden,all_moving3,cmap='terrain',vmin=-3,vmax=3)
plt.yticks(np.arange(0, len(alle_maanden)+1, 12))
plt.colorbar()
plt.ylabel('Time (months)')
plt.xlabel('Distance (km)')
plt.title('Transect 3')
plt.grid()

#%%
as1= np.nanmean(all_asym_mean)
as1std=np.nanstd(all_asym_mean)

wl1= np.nanmean(all_mean)
wl1std=np.nanstd(all_mean)

std1=np.array(std1)
std2=np.array(std2)
std3=np.array(std3)
std1=std1*np.sqrt(2)
std2=std2*np.sqrt(2)
std3=std3*np.sqrt(2)
amp1=np.nanmean(std1)
amp1std=np.nanstd(std1)

amp2=np.nanmean(std2)
amp2std=np.nanstd(std2)

amp3=np.nanmean(std3)
```

```
amp3std=np.nanstd(std3)
```