Automated rubblestone boundary detection

Advisory Report

Group 7: Sicco de Jong, Milou Maathuis, Lennart Murk, Jelmer Sonnemans, Nik Verweel

Project RGIC23-07

Commissioner: Luuk van Gorp, Antea Group Nederland

Coach: Nandika Tsendbazar

July 2nd, 2024

Version: Final

Table of Contents

Executive summary 3
1 Introduction 4
1.1 Background 4
1.2 Problem analysis 5
2 Objectives 6
3 Methodology 7
3.1 Data 7
3.2 Approach 8
3.2.1 Pre-processing 9
3.2.2 Thresholding 11
3.2.3 Random forest classifier 11
3.2.4 Gradient boosting classifier 12
3.2.5 Histogram gradient boosting classifier 12
3.2.6 Support vector machine 12
3.2.7 Post-processing 13
4 Findings 14
4.1 Pre-processing 14
4.2 Thresholding 15
4.3 Random forest 17
4.4 Gradient boosting classifier 18
4.5 Histogram gradient boosting classifier 20
4.6 Support vector machine 20
4.7 Post-processing 21
5 Analysis & Insights 22
5.1 Methodologies to determine the boundary 22
5.2 Accuracy and practicality of methodologies 22
6 Conclusion 26
6.1 Methodologies to determine the boundary 26
6.2 Scalability of the methods 26
7 Recommendations 27
7.1 Model recommendations 27
7.2 Computation power limits 27
7.3 Training data 27
7.4 Properties and parameters 28

7.5 Hand annotation 28
8 Bibliography 29
Appendices 31
Appendix I: Data management plan 31
Appendix II: Overview parameters accuracy metrics 31
Appendix III: Workflow creating boundary metrics 33
Appendix IV: Computational specifications 34

Executive summary

The protective rubble stone layer of the dykes of the Oosterscheldekering needs to be monitored for subsidence. Doing this by manual inspection leads to inaccuracies. Therefore, five data-driven methods were explored to automate this process. The data used for this project are point clouds obtained by Antea Group in several phases of flights over the Brouwersdam and Roggenplaat. The point clouds were converted to Digital Surface Models. These were classified into rubble stone and non-rubble stone with thresholding, a random forest model, a gradient boosting model and a histogram based gradient boosting model. A support vector machine based methodology was experimented with, but not further explored due to time constraints. The methods showed various degrees of usefulness, but in the end the gradient boosting model yielded the most promising results. Although the resulting accuracy is not yet sufficient, it is expected that the method will, upon further development, lead to the desired results through the use of more training data and more further parameter refinement through a grid search.

1 Introduction

1.1 Background

The Oosterscheldekering is the biggest storm-surge barrier of the Netherlands, primarily built with the purpose of protecting the Netherlands against storm surges while keeping the Oosterschelde open. Protective dykes surrounding the storm-surge barrier also play a fundamental role in keeping the Netherlands safe and are an important part of the Dutch delta works. Unprotected dykes decay over time due to erosive tidal forces, waves, and subsidence. For this reason, most sea-dykes in the Netherlands are covered in a layer of rubble stone close to the waterline, reducing the amount of erosion.

Over time, the protective layer of rubble stone can sink or collapse due to the decaying forces. Storms and high waves may cause the movement of rubble stones. This can cause the underlying dyke to become once again vulnerable. It is for this reason that Rijkswaterstaat has been tasked with monitoring the state of the rubble stones of dykes surrounding the Oosterscheldekering. Under this directive, some 71 hectares of sea-dykes are monitored through field inspections (Figure 1; Fase 1, Fase 2 and Fase 4).

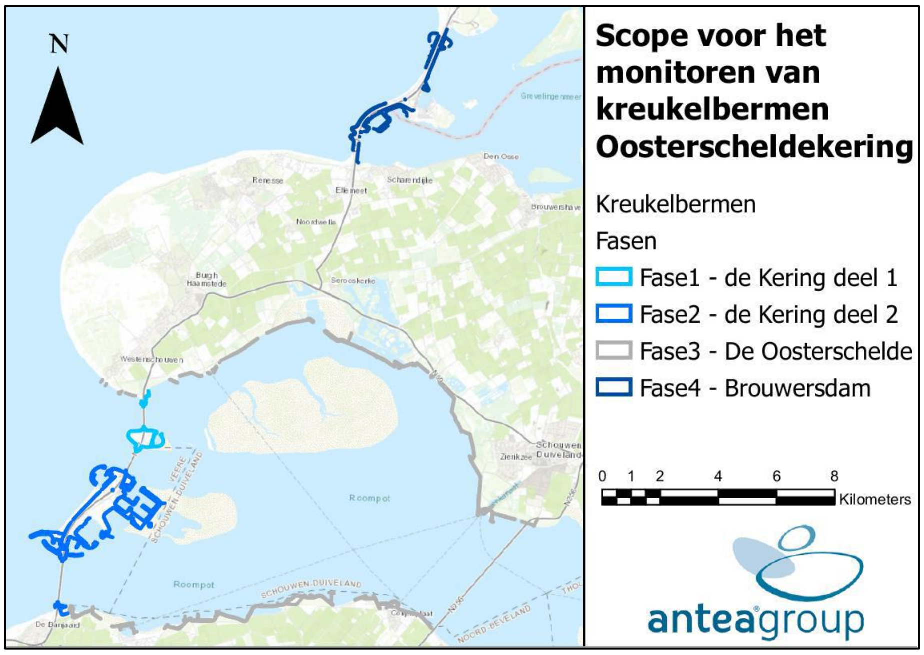

Figure 1 - Map of project area, courtesy of Antea Group. Phase 3 is not included in this project due to a scope change by Antea Group.

Visual inspection, as performed currently by Rijkswaterstaat, provides limited to no information about the continuous processes taking place and only results in the identification of areas where the level of rubble stone is (nearly) too low. An added risk is the potential for inaccuracies using this methodology.

As such, Rijkswaterstaat has commissioned Antea Group to explore the use of repeat collection of digital data and automated systems to monitor erosion and subsidence of rubble stone more closely. This has resulted in the collection of Unmanned Aerial Vehicle (UAV) photogrammetry data and preliminary experimentation with automated rubble stone boundary detection (Antea Group project number 0480486.100). Potential opportunities to perform repeat collections of data in the study area in the future offer the possibility to perform temporal change detection, allowing better insight into the continuous processes of rubble stone movement.

1.2 Problem analysis

Rubble stones protecting sea-dykes near the Oosterscheldekering move due to a multitude of reasons including erosion, subsidence, tidal forces but primarily storms. Excessive movement or subsidence may lead to insufficient protection of the underlying dykes. As such, monitoring of these rubble stones is of significant importance.

Current field monitoring methods to track the movement of rubble stones by Rijkswaterstaat are considered to be at risk of being inaccurate, and do not offer insight into changes happening over time. **A data-driven approach using photogrammetry-derived point clouds offers a potential solution. Preliminary boundary detection methods as developed by Antea Group have proven usable, but accuracy metrics show there is room for improvement.**

Considering global climate trends and rising sea levels, frequent and robust monitoring of the rubble stones protecting the sea-dykes will be increasingly important (Haasnoot et al., 2019). An automated, data-driven approach with repeat data collection should allow for more accurate rubble stone monitoring. Improvements to the algorithm detecting the separating boundary between rubble stone and other parts of the dyke are needed.

Developing an enhanced method to classify rubble stone and dykes should increase the accuracy of boundary detection. Such a method should be easily applicable to future data collection and fit into existing workflows or pipelines.

2 Objectives

The primary objective of this project is to devise a method that will most accurately and effectively find the boundary between rubble stone and the dykes near the Oosterscheldekering. This advisory report is one of two key products as proposed in BeShore project proposal RGIC-23-07. The other product is a set of scripts to classify rubble stone and create a boundary. In this report, the following proposed questions as outlined in the project proposal are answered:

What methodologies can be used to determine the boundary between rubble stone and dykes?

- *Can spectral analysis play a role in the classification of rubble stone?*
- *Can shape uniformity be used to differentiate between rubble stone and dykes?*
- *Can the use of machine learning or deep learning improve the accuracy of boundary detection?*
- *How can misclassifications be avoided?*
- *Are additional datasets required to determine the boundary accurately?*

Which methodology can be both accurate and practical in usage?

- *What are the positives and negatives of different methodologies?*
- *To what extent are different methodologies computationally demanding and complex in usage?*

Which additional steps are needed to make sure the method is scalable?

Considering the purpose of the project, a high level of boundary detection accuracy was deemed to be of great importance. To match the input data accuracy, methodologies should identify the boundary line to within an accuracy of 10 cm in the X and Y dimensions with respect to ground truth data. Misclassification of pixels in the process of the boundary line detection was considered less relevant if the desired accuracy of the boundary line was reached. Where possible, computational demands should be minimized.

3 Methodology

3.1 Data

Input point clouds and polygons for this project have been supplied by Antea Group. A brief description of the methodology used to collect the input data follows. Further details, including processing reports, can be found in Antea Group project number 0480486.100. For a detailed data management plan and data structure, consult Appendix I.

Point clouds

UAV imagery to derive the input point clouds was collected between November 2022 and March 2024 using a DJI Matrice 300 RTK UAV equipped with a DJI Zenmuse P1 camera. Images were collected within a time window of two hours before or after low tide, with a targeted overlap of 80%. Collected images were processed into point clouds using Agisoft MetaShape. Georeferencing point clouds into the WGS84 coordinate reference system (CRS) was done using the Emlid REACH RS2+ RTK GNSS Receiver by measuring coordinates at ground control points (GCPs).

Point clouds were created by Antea Group for each phase of the study area as described in Chapter 1.1. Every chunk is a point cloud resulting from the images taken during one flight. Due to high computational demands of processing all this data and limited computer resources, a subset of the input data was taken as seen in Table 1.

Table 1 - Overview of the used chunks of phase 1: Roggenplaat.

Chunks	Usage
1	Parameter tuning for thresholding and training of machine learning models
2	Methods validated on Digitaal Topografisch Bestand
3	Parameter tuning for thresholding
5	Methods validated on Digitaal Topografisch Bestand
6	Hand annotated, used as validation

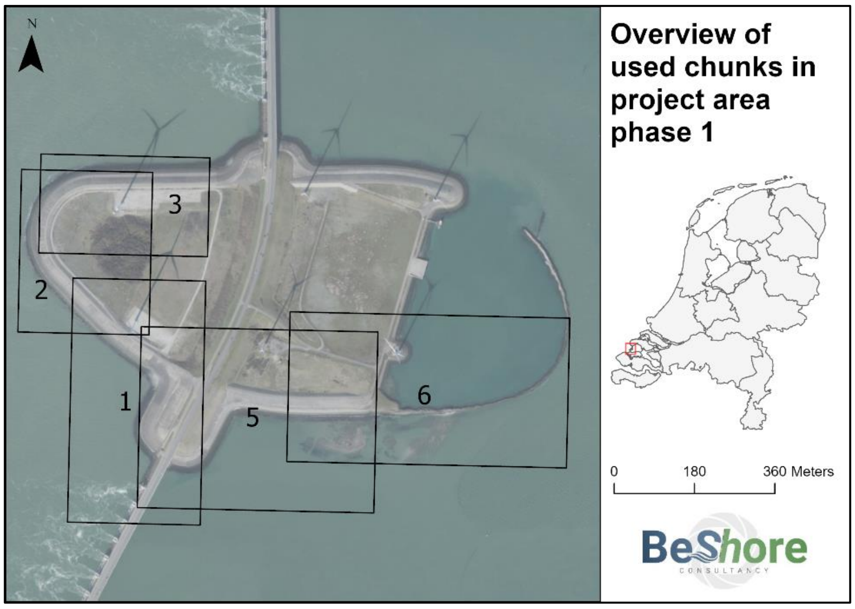

Figure 2 - Graphical overview of chunks used from Phase 1.

Project extent

Boundaries of each chunk (OSK boundaries) used as shown in Table 1 were supplied by Antea Group in the WGS84 CRS.

Ground truth reference

Reference data for model training and validation was found in the Digital Topographical Archive of water infrastructure (DTB) downloaded inline in the script from Publieke Dienstverlening Op de Kaart (PDOK) using the Atom download service. The DTB contains the types of infrastructure in the Netherlands. In the DTB, rubble stone is considered a class 2 object. Class 2 objects should be entered into the DTB with an accuracy of 10 centimetres in the x and y coordinate and 12 centimetres in the z coordinate (Rijkswaterstaat & Ministerie van Infrastructuur en Waterstaat, 2024). Using the images gathered during the flight above chunk 6 and the DTB, a more accurate, hand-annotated validation set has been created. This is described in more detail in Chapter 3.2.1.

3.2 Approach

As the project is explorative in nature, a multitude of different approaches were taken in Python scripts. Often, these approaches were developed in tandem next to each other. Finetuning of specific parameters within each method was also an important part in the approach. The scripts were written in Python. To ensure that all methods started with the same data in the correct CRS, the input data was pre-processed before exploring and experimenting with different methods. Not all methods proved equally useful, and thus were not all equally far developed when initial testing showed limited results. This is further expanded upon in Chapter 4.

The data action model for this project (Figure 3) provides an overview of the steps taken to develop a method that creates a boundary between rubble stone and non-rubble stone. Further detail per step is provided in the remainder of this chapter.

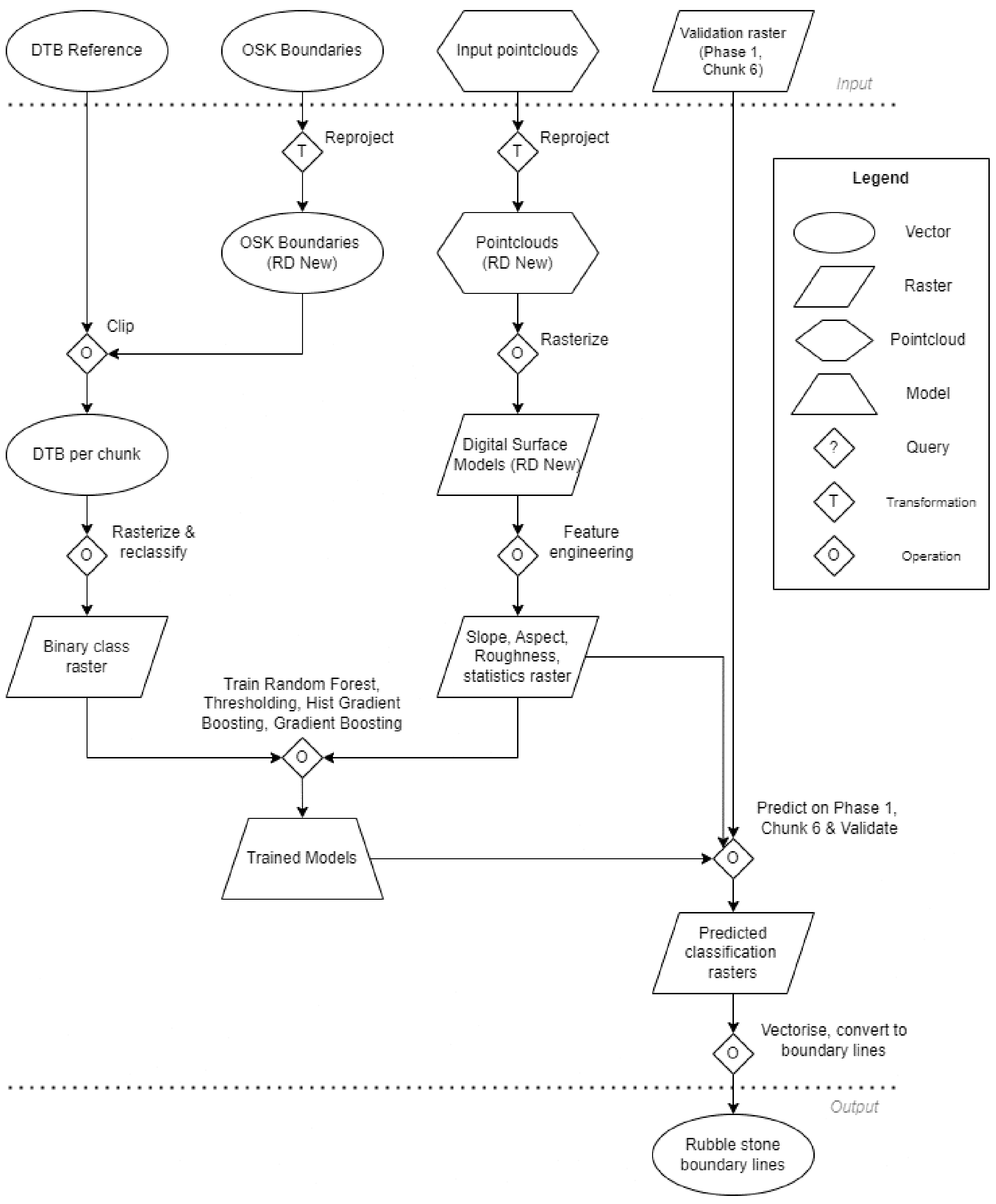
DTB Reference
OSK Boundaries
Input pointclouds
Validation raster (Phase 1, Chunk 6)
Input
Reproject
T
OSK Boundaries (RD New)
Pointclouds (RD New)
Clip
O
Rasterize
Digital Surface Models (RD New)
DTB per chunk
Rasterize & reclassify
Feature engineering
Binary class raster
Slope, Aspect, Roughness, statistics raster
Train Random Forest, Thresholding, Hist Gradient Boosting, Gradient Boosting
Trained Models
Predict on Phase 1, Chunk 6 & Validate
Predicted classification rasters
Vectorise, convert to boundary lines
Output
Rubble stone boundary lines
Legend
Vector
Raster
Pointcloud
Model
?
Query
Transformation
Operation

Figure 3 - Data action model: providing an overview of the steps taken to develop a method that creates a boundary between rubble stone and non-rubble stone.

3.2.1 Pre-processing

Reference data & validation set

To prepare the reference DTB data for algorithm training, the data was clipped per chunk using the OSK boundaries. As OSK boundaries were generated by Antea Group in Agisoft MetaShape, they were in CRS WGS84 and thus needed reprojection into CRS RD New first.

The clipped reference shapefiles were then converted into binary class rasters with a resolution of 5 centimetres. The DTB class “rubblestone” was assigned the value of 1 in the binary class raster, and all other DTB classes were assigned the value 0.

Upon inspection and in consultation with Antea Group, it was discovered that some parts of the DTB have not been updated since 2002. During preliminary research, inconsistencies were found between the DTB and orthomosaic images. This indicates that in some areas the rubble stone is incorrectly classified, further expanded upon Chapter 4.1.

The validation set was created from data from phase 1, chunk 6. This chunk was chosen as it was one of the largest chunks of phase 1 covering a multitude of different classes of land cover. Using an orthomosaic generated in Agisoft MetaShape, the DTB polygons were manually altered to accurately reflect the ground truth. This was done based on visual inspection using ArcGIS.

Due to the observed differences between the DTB and orthomosaic images, the difference between the DTB and the produced validation set was quantified. By overlaying the DTB and hand annotated validation set and calculating distance metrics between the lines, the average inaccuracy of the DTB was found.

Point cloud reprojection & rasterization

The point clouds produced by Antea Group were supplied in the WGS84 CRS, and thus required reprojection to RD New CRS. Due to computational constraints, one point cloud (phase 1, chunk 6) had to split in two as memory overflow errors were encountered. This function is no longer called in the main script, though the function still exists in the script to be called if needed by the user. Inspection in different software including QGIS and CloudCompare confirmed the reprojection was successful. As RD New (EPSG:28992) does not contain height (or 'z') values, the obtained height values are still in the WGS84 CRS. Practically, this means that the heights in the DSM are around 44 meters higher in WGS84 compared to RD New. For this project, calculations are done relative to other resulting data. Therefore, this will have no effect on the project results.

Following this, the point clouds were used to generate a DSM with 5 cm resolution for each chunk using the Whitebox Tools package, utilizing the lidar_tin_gridding function (*parameter='z', returns='all', resolution=0.05, max_triangle_length=1*). The goal of the project was to monitor the movement of rubble stone at 10 centimetres accuracy. For this reason, a spatial resolution of 5 centimetres was chosen to allow classifications within the desired accuracy. Lower spatial resolutions would risk exceeding the 10-centimetre accuracy requirement, whilst higher spatial resolutions would become too computationally demanding.

Initially, the maximum triangle length (i.e. the maximum distance between two points where a line is drawn as part of the triangle generation) was set to 50 cm but upon visual inspection it was discovered that, whilst effective at ignoring water and large areas where no points were present in the point cloud, it left many gaps. By increasing the maximum triangle length to 1 m, far fewer gaps were present in the DSM.

Feature engineering

From the DSM, different feature layers were engineered to ensure that the different machine learning algorithms and thresholding algorithm had sufficient data to be trained on. Based purely on height, classifying the rubble stones would be difficult. As such, DSM-derivative rasters were created. Using the DSM as input, a slope, aspect, and roughness (SAR) raster was engineered. Subsequently, the focal standard deviation (std) was calculated using a 11x11 sliding window for each of the SAR bands and added to the raster. This resulted in a raster containing six bands: slope, aspect, roughness, slope_std, aspect_std and roughness_std that can be used for model training and thresholding.

3.2.2 Thresholding

Thresholding is a method that classifies an image by assigning a value to each pixel based on thresholds set by the user for certain parameters (Shapiro & Stockman, 2001). There are different thresholding methods, for this case a simple method of thresholding was used as it was deemed to be sufficient for this type of case. (Sahoo et al., 1988).

Thresholding was performed based on the standard deviation of the slope, aspect and roughness values in the SAR rasters. A high standard deviation in these rasters means that it is more likely that there is rubble stone (i.e. there is a big standard deviation for slope and aspect when looking at rubble stone). Next to that, the minimum and maximum of roughness was used. Initially, the minimum and maximum values of the slope and aspect were used, but it was decided to not use these in thresholding. This was done as the results showed inaccuracies, and using the standard deviation showed clearer differences between classes.. Exact combinations of thresholding values and accuracy metrics can be found in Appendix II.

Pixels classified as rubble stone by the thresholding algorithm were assigned the value of 1, and pixels classified as no rubblestone were assigned the value of 0. Thus, creating a binary raster.

To get the best thresholding values, a loop in the Python script was used to iterate through a list of different parameters and test which are the most appropriate for chunks 1 and 3. The classification report of these loops gives insight into different statistical results such as precision and accuracy of both rubble stone and non-rubble stone compared to the DTB. As the DTB is not perfectly accurate for rubble stone, further refinement of the thresholding parameters was done by visually comparing the over and under classification to the Actueel_orthoHR layer of the Nationaal Georegister. The Actueel_orthoHR is the most recent nationwide 8cm resolution orthophoto in RGB which is a mosaic of maximum, five different years of imagery (Nationaal Georegister, n.d.).

3.2.3 Random forest classifier

Random Forest is a machine learning method that constructs multiple classification decision trees. Each of these trees predicts a classification, and the most predicted choice will be the resulting classification (Belgiu & Dragut, 2016; Cañete-Sifuentes et al., 2021).

The Random Forest was initialised and trained on the SAR raster and the focal statistics, including all bands. Chunk 1 of phase 1 was selected for training. The input data was split randomly into 70% training and 30% test data to give provisional insights into the accuracy metrics, which were used to tune parameters.

Different parameters were experimented with, such as changing the number of trees, the maximum depth, maximum number of features and different class weights using a grid search. A grid search trains and cross-validates models, to find the best parameters within a parameter grid. Exact combinations of parameters and accuracy metrics can be found in Appendix II.

After experimentation, the best parameters were selected, and classification was performed on the validation chunk.

3.2.4 Gradient boosting classifier

Boosting is a machine learning method in which an initial model of a single classification tree (a stump) is fitted to data. After this, a second stump is constructed that aims to accurately predict cases where the first model performs poorly. These two models are assumed to perform better together than either model will on its own. This process is repeated for a user specified number of times - usually in the range of tens to hundreds (Natekin & Knoll, 2013).

Gradient boosting assumes that the possible next model minimizes the overall prediction error when combined with the previous models. To reduce the overall prediction error target outcomes are set. Target outcomes are set based on the gradient of the error that is derived from changes in the prediction. If the change in prediction results in a large drop in error, the next target outcome is set high, if the change in prediction results in no change in error, the next target outcome is set to zero. For the final classification, a weighted sum of the decision stumps is used (Fruend & Schapire, 1997; Krivoguz et al., 2023).

To improve predictions of the gradient boosting model, a grid search was attempted to find good parameters. However, after a runtime of two days the gradient boosting model was not done yet. This made it impossible to do a grid search within the scope of the project. With the factor of time in mind, most parameters were kept default. Three parameters were set to prevent overfitting of the model and increase efficiency. The specific combination of parameters and accuracy metrics can be found in Appendix II.

3.2.5 Histogram gradient boosting classifier

The histogram gradient boosting classifier is a faster version of the gradient boosting classifier. It is a histogram based gradient boosting classification tree which allows it to make use of parallel processing. The histogram gradient boosting classifier generally works better for large data sets where the number of samples is over 10.000, as is the case in this project (Hang et al., 2021).

Like Random Forest, a grid search was performed to find parameters that resulted in good accuracy metrics. Exact combinations of parameters and accuracy metrics can be found in Appendix II.

3.2.6 Support vector machine

Support vector machines transform data, so it fits into a high dimensional space where it becomes linearly separable (Cortes & Vapnik, 1995; Salcedo-Sanz et al., 2014). A weighted raster was calculated using a simple formula of the roughness, and standard deviation of the roughness, slope, and aspect. The weighted raster was then segmented using a k-means clustering algorithm to prepare the data to train the support vector machine. After segmenting the image, the segments were split into unique combinations of standard deviations and ground truth values which are used for training. A grid search was used to see which parameters work well for the classification. The support vector classifier was then fit using the best estimators found by the grid search.

3.2.7 Post-processing

The output of each of the classification methods is a binary raster. Post-processing needed to be done as the desired product was a boundary showing the border between rubble stone non-rubble stone.

First, a function that fills gaps was applied on the binary raster. The gap filling function uses a sliding window of 11x11 pixels to calculate the mean value for each centre pixel. All pixels with a mean value above the threshold of 0.1 get value 1 (rubble stone) and the others get value 0 (non-rubble stone). The output of the gap filling is a binary raster. The filled binary raster was converted to two polygon files. One file containing a polygon representing rubble stone. The other containing a polygon representing non-rubble stone.

A function that makes a buffer of 0.5 meter around the non-rubble stone polygons was applied. Next, a function was applied that subtracts the buffer from the rubble stone polygon. A function that applies a buffer of 0.1 meter was applied next. The output is a multipolygon. In the end a multilinestring is created by taking the boundary of the polygon coming out of the previous function.

Different values were tested for the threshold and buffers. All tested values were put in loops to automate the testing process. All combinations were tried wherein the buffer around the non-rubble stone was bigger than the buffer around the multipolygon. It was determined by visual assessment what values provided the best result. This means the final output line was compared with the validation set in QGIS. The specific combination of parameters and accuracy metrics can be found in Appendix II.

The boundaries created by the script have been used to calculate the distance between the validation set and the classified set. To calculate the distances, the boundary was transformed using ArcGIS from a multipart shape to a single part shape. To remove outliers when calculating distance statistics and improving calculation time, only the largest shapes were used. The distance from the created boundary to the validation set was then calculated, on which the metrics are based. To adjust for extreme outliers, the median has been used instead of the mean. A workflow of this process can be found inAppendix III: Workflow creating boundary metrics Appendix III: Workflow creating boundary metrics.

4 Findings

4.1 Pre-processing

Using the new orthomosaic image generated in Agisoft MetaShape and the resulting hand-drawn validation set, new insights into the inaccuracy of the training data were gained. The difference in land use classes between the DTB and the annotated validation set highlighted inaccuracies averaging around 4 meters, with maximum differences of up to 27 meters in chunk 6 of phase 1 (Figure 4).

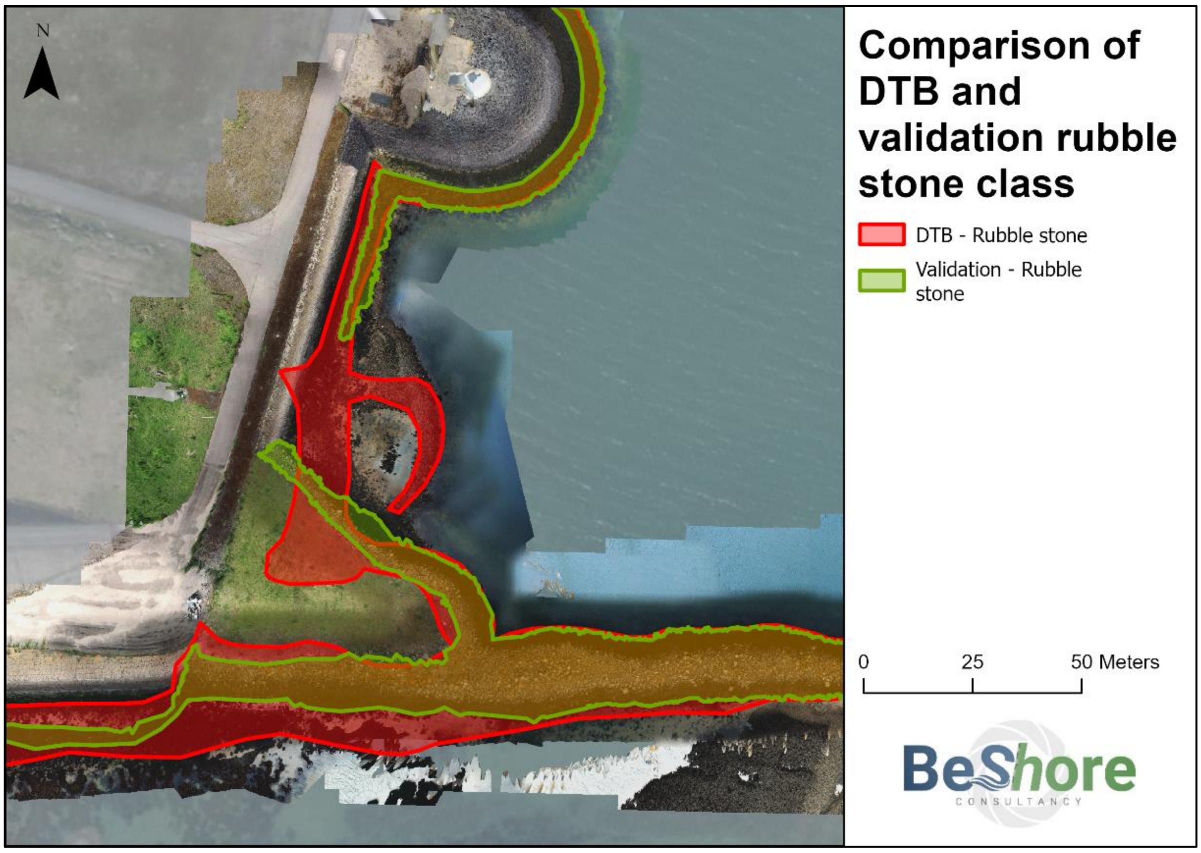
N
Comparison of DTB and validation rubble stone class
🟥 DTB - Rubble stone
🟩 Validation - Rubble stone
0
25
50 Meters
BeShore CONSULTANCY

Figure 4 - Visualisation showing the difference between the DTB and hand-annotated validation set for chunk 6 of phase 1.

The DTB was used as training data for the models, and during the experimentation with parameters often validated on the DTB. When running eventual classifications, the model metrics were found to generally be lower validating on the hand annotated dataset compared to the DTB.

Reprojection of the point clouds was observed to be time consuming and computationally intensive. Reprojecting a single cloud, with the computational specifications as outlined in Appendix IV, took between one to three hours, depending on the size of the cloud.

During rasterization information is lost as only the X, Y & Z is kept. The RGB values and intensity information were not used.

4.2 Thresholding

Thresholding the least time consuming algorithm computationally-wise compared to the other methodologies applied. However, the drawback of this algorithm is the time it takes to find the best values for classifying rubble stone. The best result has been achieved by using a minimum slope_std of 10, a minimum aspect_std of 82 and a minimum roughness_std of 0.033. When validated on phase 1, chunk 6, this results in a precision of 0.60 and recall of 0.68 (Figure 5). There are still misclassifications, but the line dividing rubble stone and grass/setting stone is well defined. When experimenting with different parameters, a different threshold for the aspect standard deviation is found to have the most influence on classification.

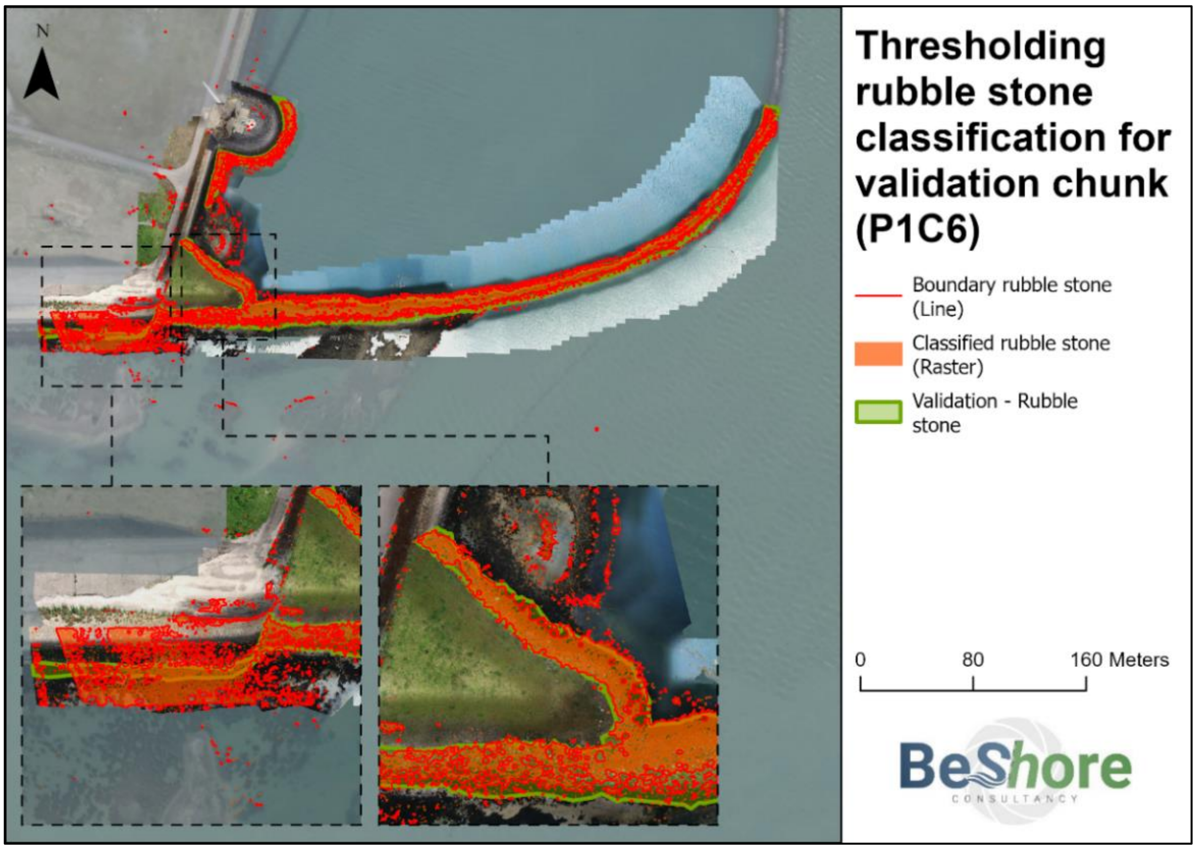
N
Thresholding rubble stone classification for validation chunk (P1C6)
Boundary rubble stone (Line)
Classified rubble stone (Raster)
Validation - Rubble stone
0
80
160 Meters
BeShore CONSULTANCY

Figure 5 - Results of thresholding classification and resulting boundary line for phase 1, chunk 6 compared to validation data set. Areas of interest highlighted.

The boundary created around the classified raster shows a median deviation from the validation raster of 2.5 meters.

When using the abovementioned values for thresholding on phase 1 chunk 2, the following metrics are found: a precision of 0.61 and recall of 0.40 (Figure 6). The boundary created around the classified raster showed a median deviation from the DTB raster of 2.7 meters.

For phase 1 chunk 5, thresholding resulted in a rubble stone precision of 0.55 and a rubble stone recall of 0.18 (Figure 7) The boundary created around the classified raster showed a median deviation from the DTB raster of 2.5 meters.

These metrics are based on the DTB as a validation, so the precision and recall might be higher, but as of yet there is no validation set for this chunk.

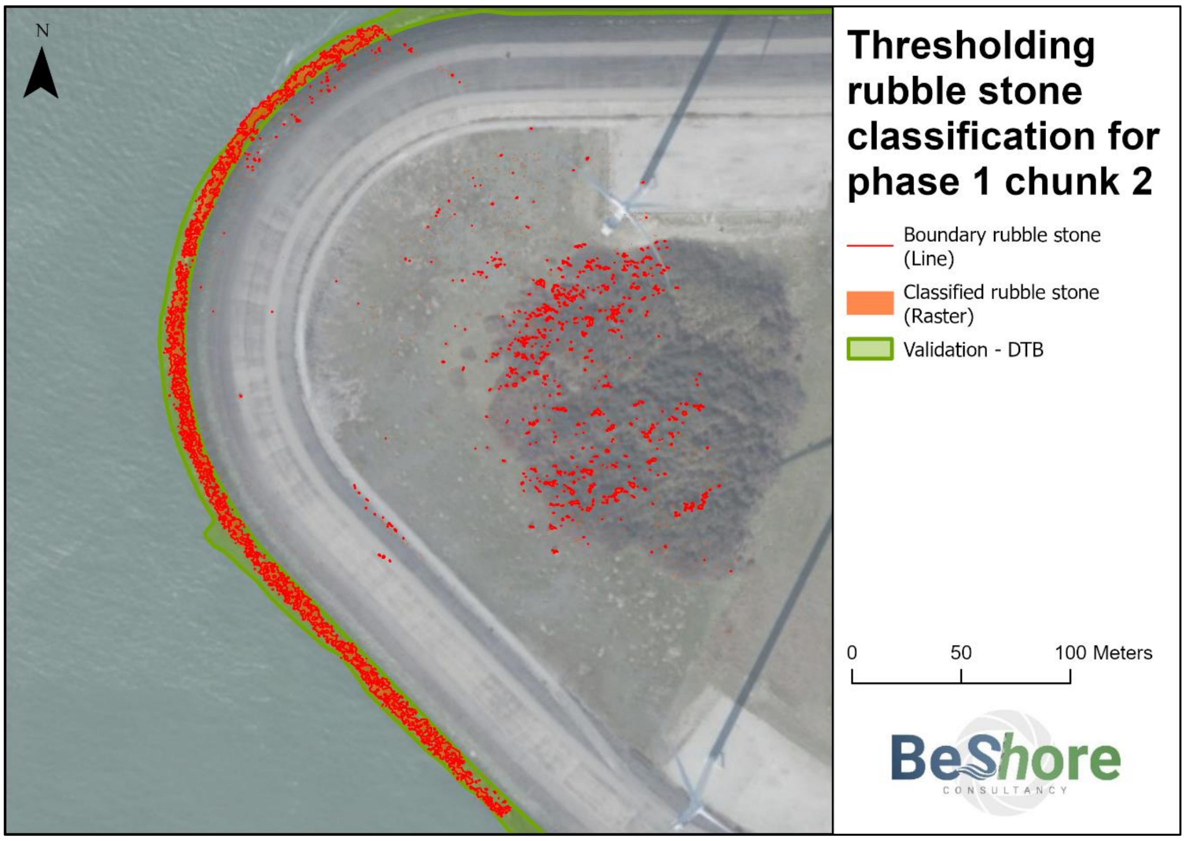
N
Thresholding rubble stone classification for phase 1 chunk 2
Boundary rubble stone (Line)
Classified rubble stone (Raster)
Validation - DTB
0
50
100 Meters
BeShore CONSULTANCY

Figure 6 - Results of thresholding classification and resulting boundary line for phase 1, chunk 2 compared to DTB.

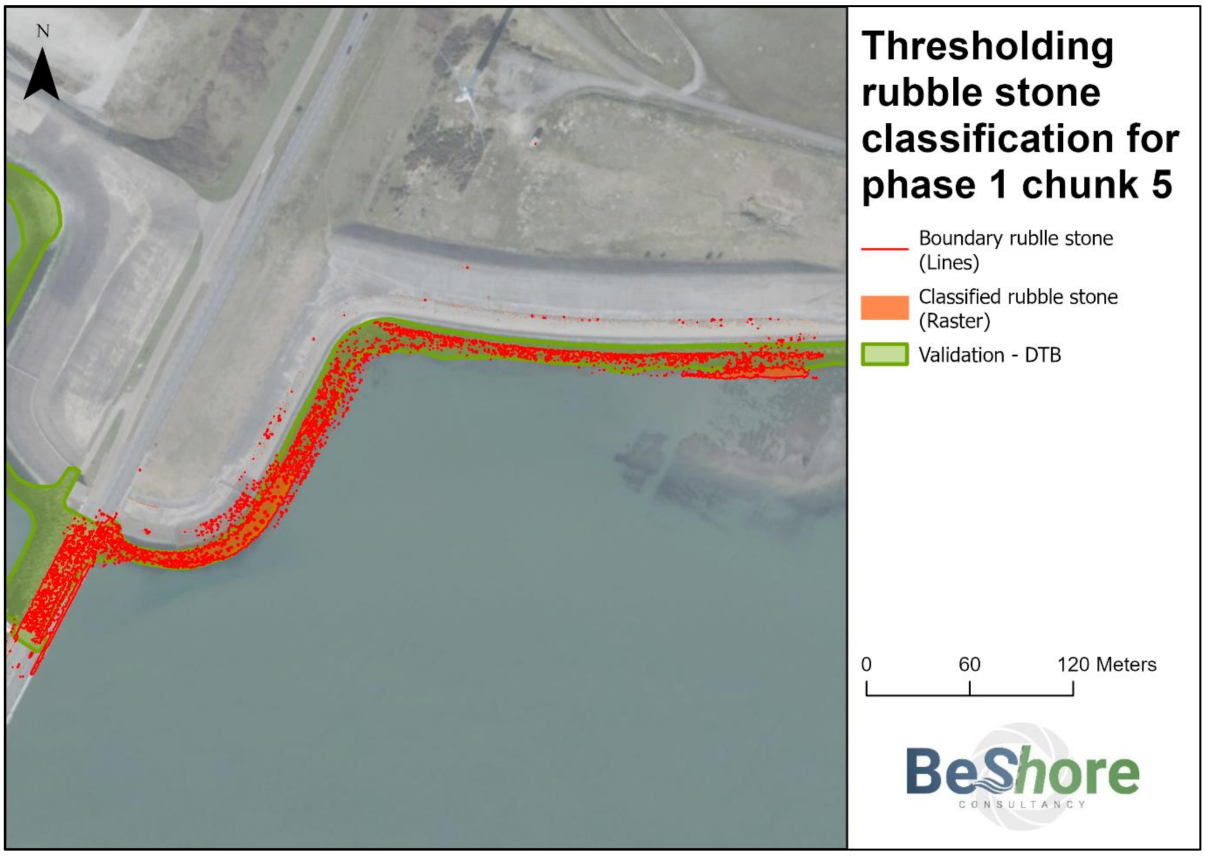
N
Thresholding rubble stone classification for phase 1 chunk 5
Boundary rublle stone (Lines)
Classified rubble stone (Raster)
Validation - DTB
0
60
120 Meters
BeShore CONSULTANCY

Figure 7 - Results of thresholding classification and resulting boundary line for phase 1, chunk 5 compared to DTB.

4.3 Random forest

Based on the parameter experimentation as described in Appendix II, the best parameters were found to be 100 trees, balanced weights and maximum features ‘sqrt’. When validated on phase 1, chunk 6, this results in a rubble stone precision of 0.22 and rubble stone recall of 0.68, signifying a lot of overclassification of rubblestone. This resulted in the correct classification of most rubble stones, but also non-rubble stone pixels being classified as rubblestone, as can be seen in Figure 8.

The boundary created around the classified raster showed a median deviation from the validation raster of 9.3 meters.

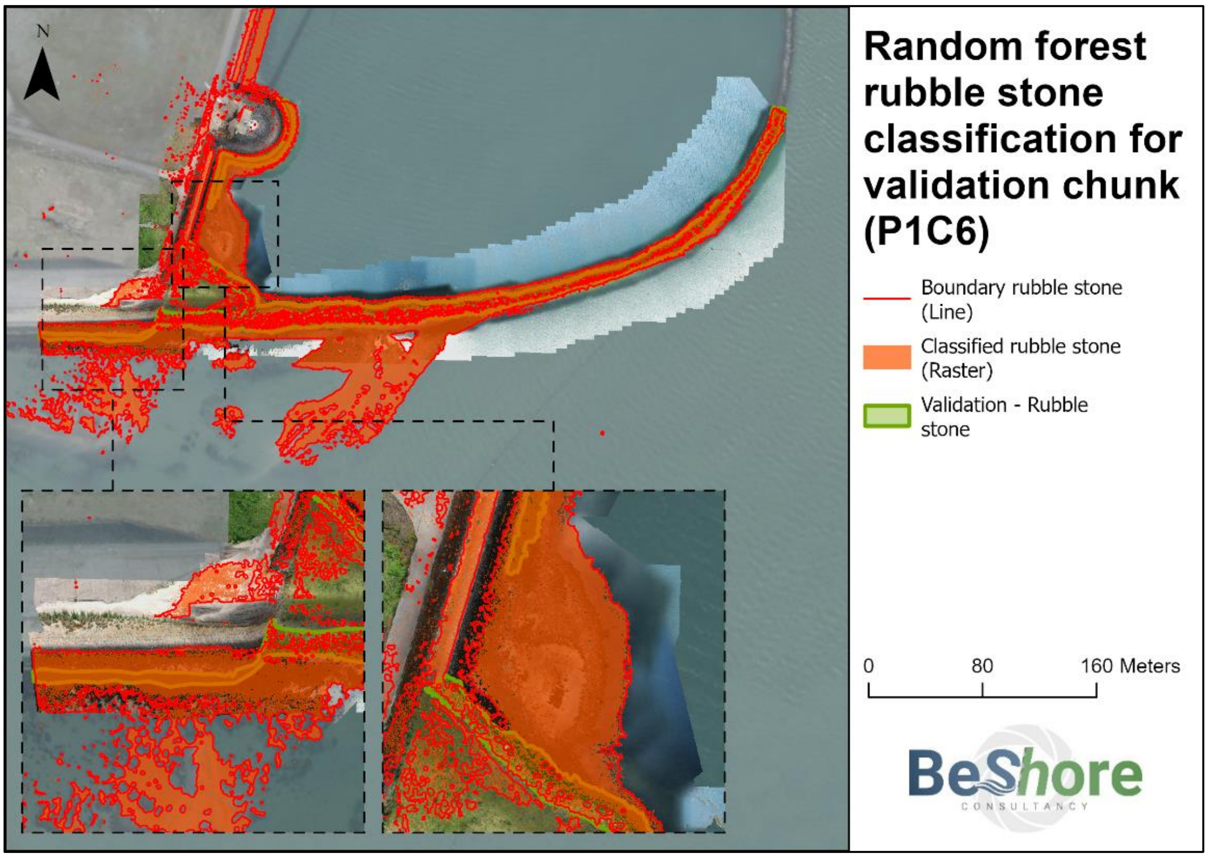
N
Random forest rubble stone classification for validation chunk (P1C6)
Boundary rubble stone (Line)
Classified rubble stone (Raster)
Validation - Rubble stone
0
80
160 Meters
BeShore CONSULTANCY

Figure 8 - Results of random forest classification and resulting boundary line for phase 1, chunk 6 compared to validation data set. Areas of interest highlighted.

4.4 Gradient boosting classifier

Initial trials for the gradient boosting classifiers were ran with the DTB as validation data, these results were deemed insufficiently validated for the case study. The metrics for two of these trials can be found in Appendix II. A gradient boosting classifier was run on a parameter grid with single values for three parameters which can be seen in Appendix II.

For the model with the parameters as outlined in Appendix II, the runtime is quite high for this method, but in the end, it produced a recall of 0.85 and a precision of 0.59 (Figure 9). From the image it can be derived that most misclassifications occur near the boundary with the sea, which is less important for this case.

The boundary created around the classified raster showed a median deviation from the validation raster of 2.9 meters.

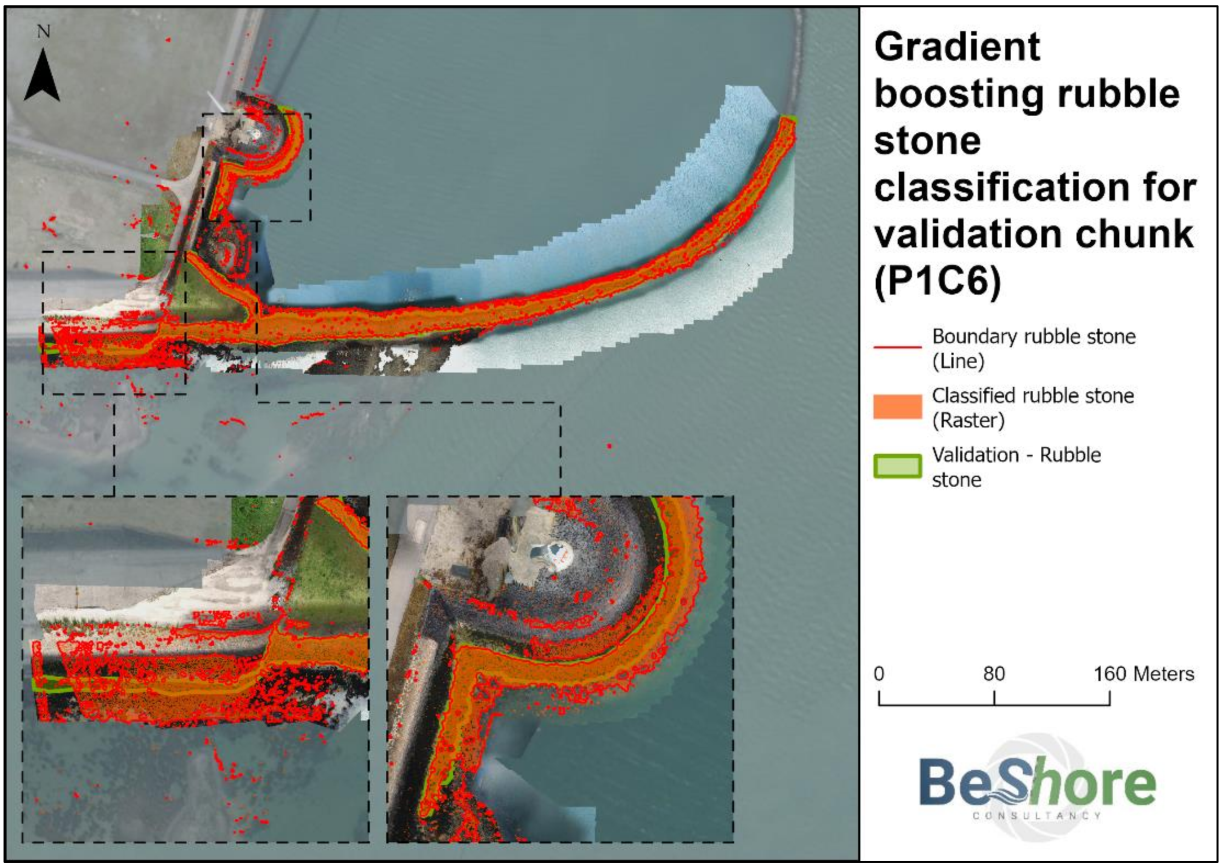
N
Gradient boosting rubble stone classification for validation chunk (P1C6)
Boundary rubble stone (Line)
Classified rubble stone (Raster)
Validation - Rubble stone
0
80
160 Meters
BeShore CONSULTANCY

Figure 9 - Results of gradient boosting classification and resulting boundary line for phase 1, chunk 6 compared to validation data set. Areas of interest highlighted.

Due to the promising nature of this methodology, the model was also used to predict on phase 1 chunks 2 and 5, like thresholding.

Classification of phase 1 chunk 2 resulted in a rubble stone precision and recall of 0.47 and 0.47 respectively (Figure 10). For phase 1 chunk 5, the rubble stone precision was 0.55 and the rubble stone recall was 0.23 (Figure 11).

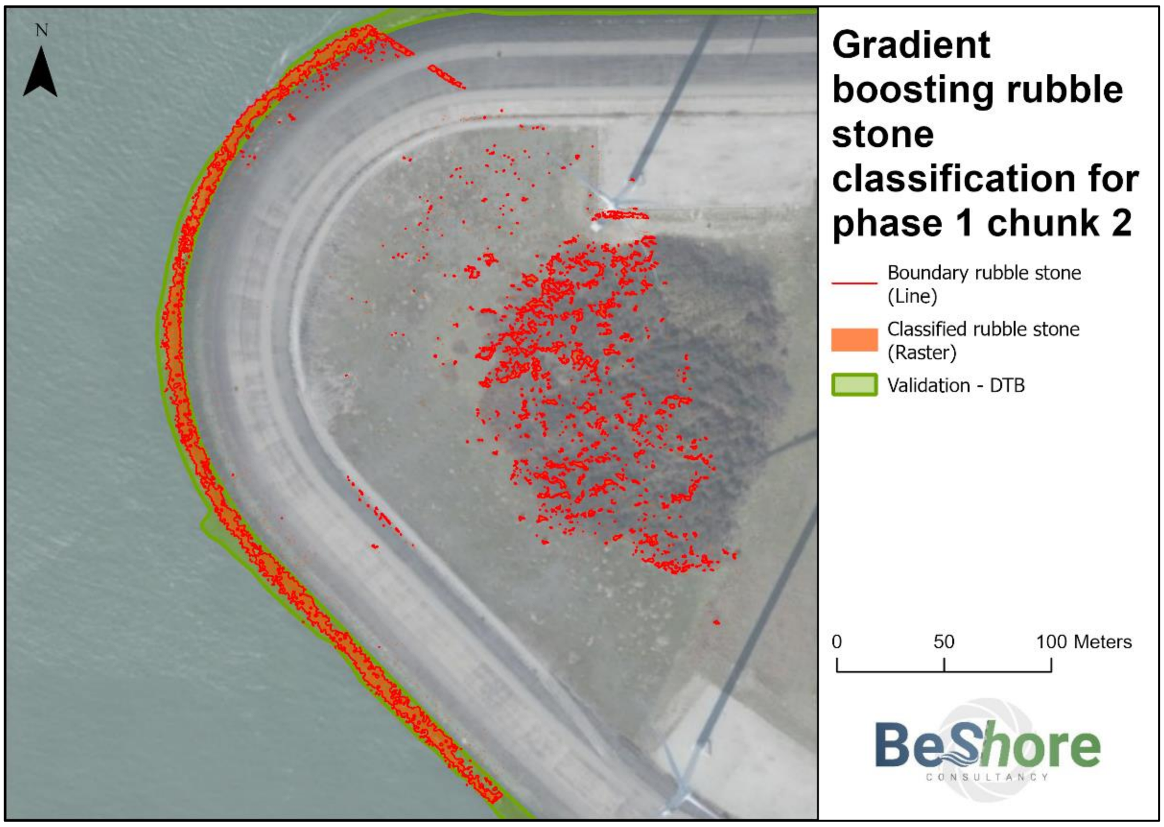
N
Gradient boosting rubble stone classification for phase 1 chunk 2
Boundary rubble stone (Line)
Classified rubble stone (Raster)
Validation - DTB
0
50
100 Meters
BeShore CONSULTANCY

Figure 10 - Results of gradient boosting classification and resulting boundary line for phase 1, chunk 2 compared to DTB.

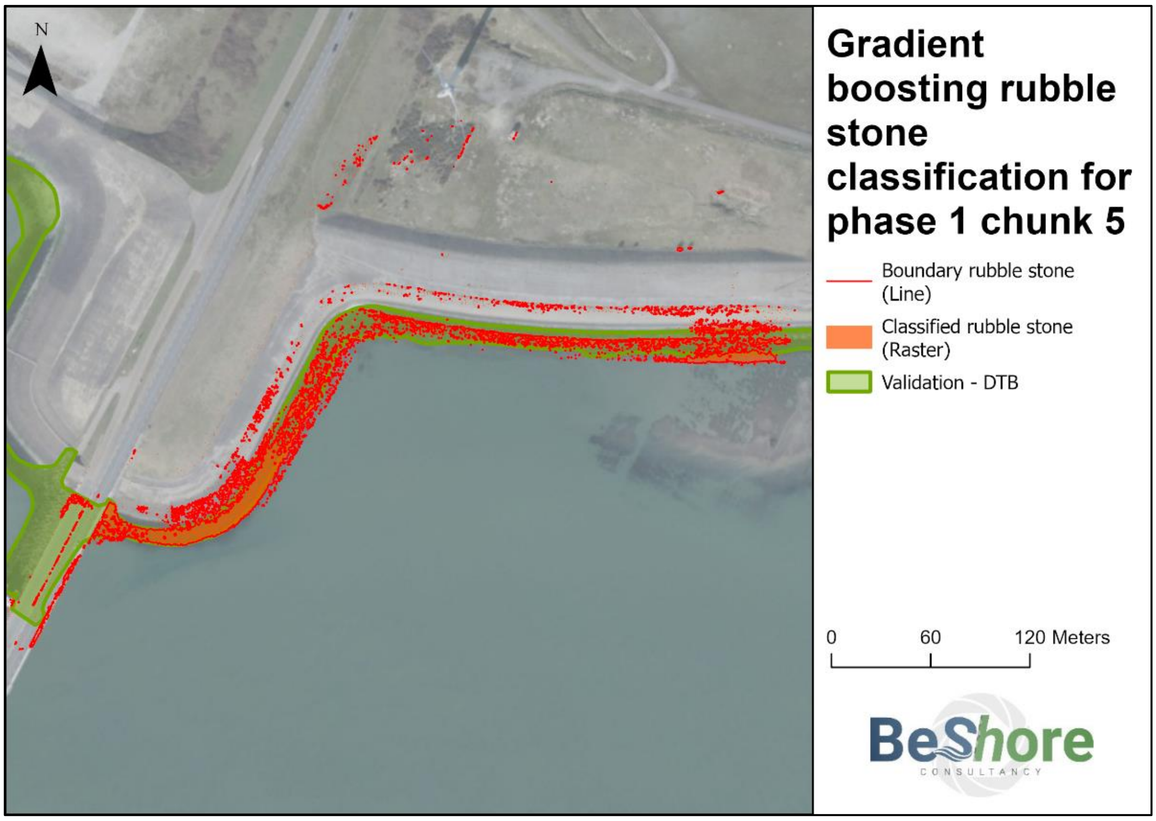
N
Gradient boosting rubble stone classification for phase 1 chunk 5
Boundary rubble stone (Line)
Classified rubble stone (Raster)
Validation - DTB
0
60
120 Meters
BeShore CONSULTANCY

Figure 11 - Results of gradient boosting classification and resulting boundary line for phase 1, chunk 5 compared to DTB.

4.5 Histogram gradient boosting classifier

Similarly to the gradient boosting classifier, the first models were validated using the DTB of chunk 6, the results of which can be seen in Figure 12. The results of the models validated on the DTB have been highlighted in the table in Appendix II. The best estimators found during grid search on the validation raster were 0.1 for l2_regularization, 0.01 for learning_rate, 3 for max_depth, 0.3 for max_features, and 75 for max_iter.

These parameters resulted in a precision of 0.31 and a recall of 1.00 of rubble stone on the validation raster. The boundary created around the classified raster shows a median deviation from the validation raster of 15.7 meters.

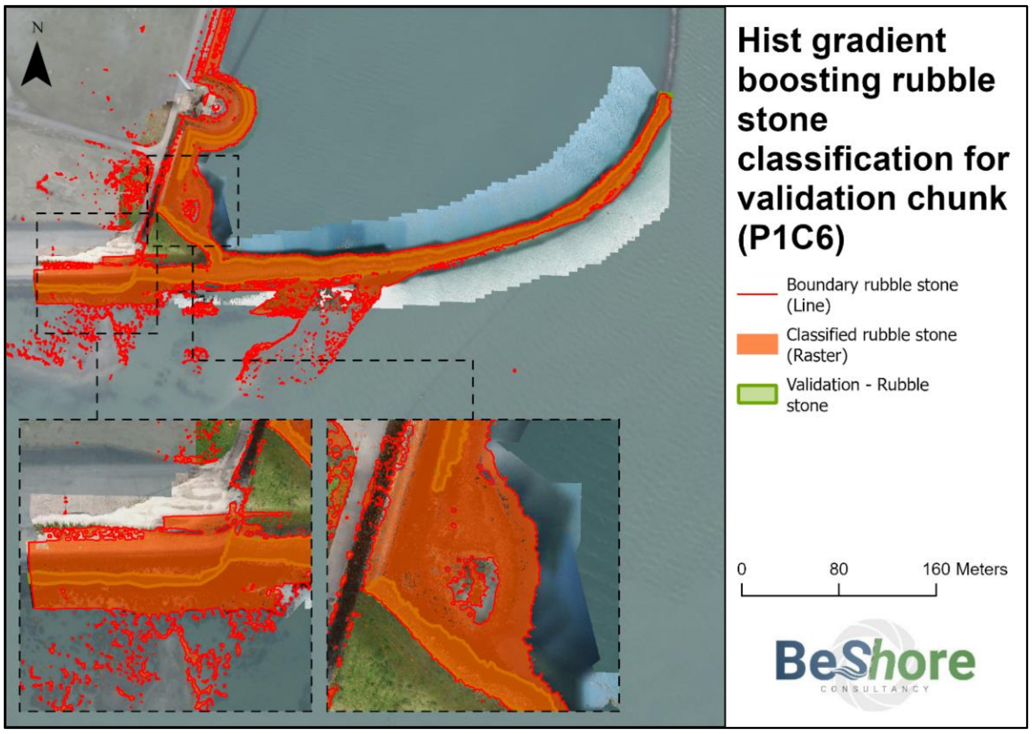
N
Hist gradient boosting rubble stone classification for validation chunk (P1C6)
Boundary rubble stone (Line)
Classified rubble stone (Raster)
Validation - Rubble stone
0
80
160 Meters
BeShore CONSULTANCY

Figure 12 - Results of histogram gradient boosting classification and resulting boundary line for phase 1, chunk 6 compared to validation data set. Areas of interest highlighted.

4.6 Support vector machine

Due to complexity and time constraints, the support vector machine was tested only on one chunk with the accuracies printed based on the test segments. One of the main hurdles for this approach was retrieving individual pixel values from the segments for a raster product. The time it would have taken to resolve this issue would either fall outside the scope of this project or would necessitate sacrificing the quality of other, more promising methods.

4.7 Post-processing

Two threshold values were evaluated for the filling of the gaps. The value 0.1 provided the best results. Three buffer values were tried for the buffer around the non-rubble stone, and four buffer values were tried for the buffer around the multipolygon. Respectively 0.5 meter, and 0.1 meter yielded the best line, as can be seen in figure 13.

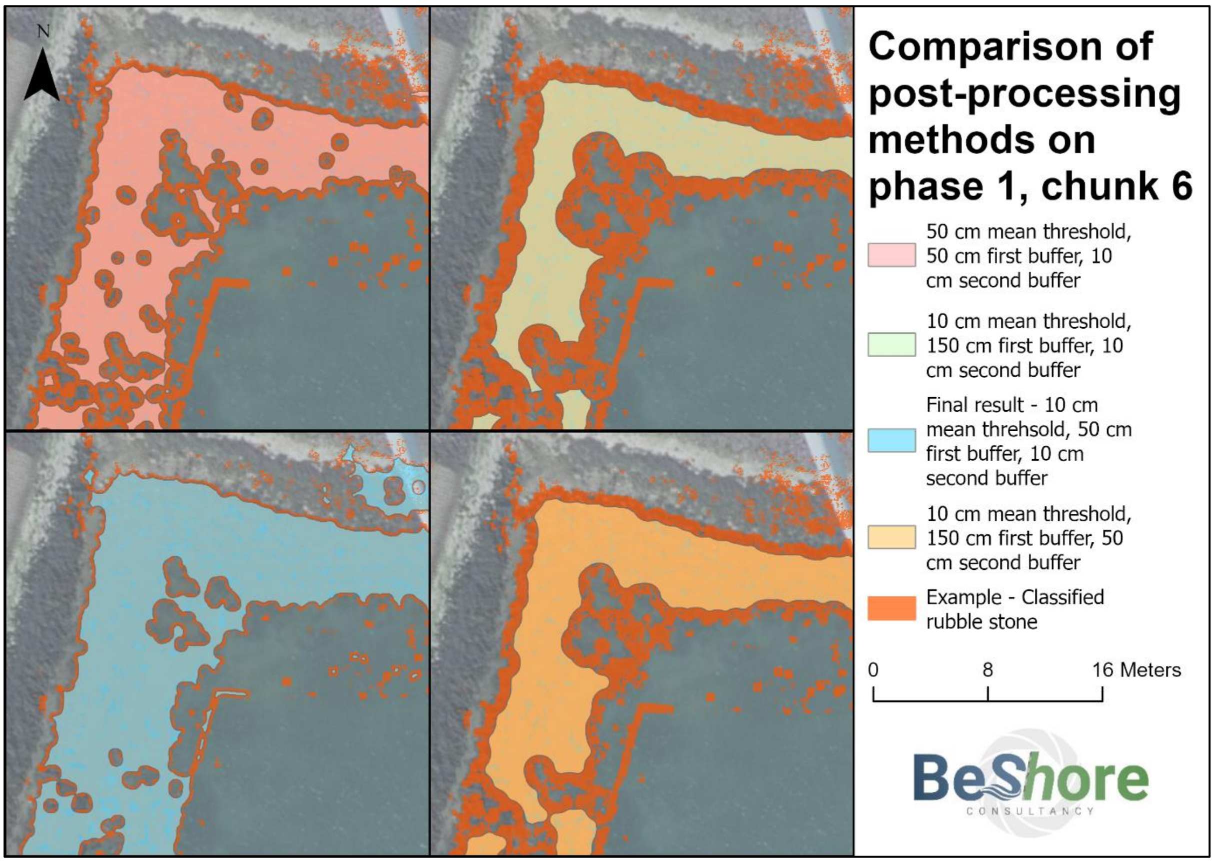
N
Comparison of post-processing methods on phase 1, chunk 6
50 cm mean threshold, 50 cm first buffer, 10 cm second buffer
10 cm mean threshold, 150 cm first buffer, 10 cm second buffer
Final result - 10 cm mean threhsold, 50 cm first buffer, 10 cm second buffer
10 cm mean threshold, 150 cm first buffer, 50 cm second buffer
Example - Classified rubble stone
0
8
16 Meters
BeShore CONSULTANCY

Figure 13 - Visualisation showing the comparison between different post-processing methods to convert classified rasters to vectors ahead of boundary line creation. Shown on a boosting classification from early in the project.

5 Analysis & Insights

5.1 Methodologies to determine the boundary

In this study, multiple approaches were evaluated to determine the boundary between rubble stones and dykes. All methods can be useful for classification, but some methods are better to use in the case of this study. In Chapter 5.2 a more in-depth analysis is given per method used.

Using additional data from the point clouds can be useful to determine the boundary more accurately. Exploring the possibility of spectral analysis using the RGB values in the point clouds for the differentiation between rubble stone and dykes turned out to not be possible due to limits in processing power and time. Using spectral values would have been useful since there was trouble with classification on the boundary between rubble stone and other types of stone/asphalt on the dyke. This misclassification can be attributed to similarities in roughness between these surfaces. Through the inclusion of multispectral data, misclassifications can potentially be decreased due to the differing spectral profiles of rubble stone and mastic asphalt, grass and kelp. A difficulty that comes with using RGB data is that some of the rubble stone is covered with kelp making it hard to see the boundary (Al-Najjar et al., 2019; Sohl et al., 2024).

For most of the machine learning methods there was a problem with overclassification. This can be seen in the accuracy metrics and the classification images. The trained machine learning models, do not improve the accuracy of the boundary detection between rubble stone and dykes, when comparing it to the thresholding method. It is expected that the use of more training data will allow for more robust models that lead to better classification performance. Another issue was that the machine learning models were all trained on the DTB, which is found to be inaccurate. Since object-based classification was not tested in the end, there can be no further analysis of this method here.

5.2 Accuracy and practicality of methodologies

The classification methods are evaluated using the precision and recall of the rubble stone classified on the hand annotated validation raster of phase 1, chunk 6. As seen in Table 2, the random forest and histogram based gradient boosting get low precision scores while getting high recall scores which means that a lot of non-rubble stone is classified as rubble stone. The gradient boosting has a higher precision score while maintaining a high recall score as well. The thresholding method has a competitive precision and recall score. It has a slightly better precision than the gradient boosting method but a lower recall score.

Table 2 – Precision, recall, and distance to boundary scores for rubble stone for each of the methods validated on the hand annotated validation set of chunk 6.

Accuracy scores	**Thresholding**	**Random forest**	**Gradient boosting**	**Histogram gradient boosting**
Precision	0.60	0.22	0.59	0.31
Recall	0.68	0.68	0.85	1.00
Median distance to boundary	2.5 meters	9.3 meters	2.9 meters	15.7 meters

The distance to boundary is an important evaluation metric for the methods, since the movement of the boundary is what must be assessed accurately.

For phase 1 chunk 2, the accuracy statistics can be found in Table 3 for thresholding and gradient boosting. These accuracy scores are lower than for chunk 6, as these are validated on the less accurate DTB.

Table 3 - Precision, recall, and distance to boundary scores for rubble stone for each of the methods validated on the Digitaal Topografisch Bestand (DTB) of chunk 2

Accuracy scores	Thresholding	Gradient boosting
Precision	0.61	0.47
Recall	0.40	0.47
Median distance to boundary	2.7 meters	3.7 meters

Table 4 - Precision, recall, and distance to boundary scores for rubble stone for each of the methods validated on the Digitaal Topografisch Bestand (DTB) of chunk 5

Accuracy scores	Thresholding	Gradient boosting
Precision	0.55	0.55
Recall	0.18	0.23
Median distance to boundary	2.5 meters	3.4 meters

Edge effect on the edges of the chunks is a problem for accurate results when calculating on individual chunks. Edge effects could cause inaccuracies in accuracy metrics, due to the point clouds being warped. These edge effects can be clearly seen in the southwest corner when classifying on chunk 6 (Figures 5, 8, 9 & 12). When classifying on mosaics of multiple chunks, these edge effects are limited, resulting in more accurate classifications. Due to the overlapping nature of the chunks, the mean of the overlap can be calculated during the creation of a DSM mosaic that will likely decrease the influence of the edge-effect.

Currently, the machine learning models are overfitting, which is likely due to them being trained on a limited amount of data (chunk 1) because of computational power limitations.

The rest of this chapter is a more detailed description of the benefits and drawbacks for each of the evaluated methodologies.

Thresholding

While thresholding is a relatively simple approach, it requires time to select parameters and experiment with said parameters. There are multiple factors at play, of which it may not be directly clear what the influence is on the classification. Different parts of dykes have different characteristics, which makes it more difficult to implement a good threshold classifier for the whole of the Oosterscheldekering. However, the approach is computationally efficient and easy to implement. This way, experimenting with different parameters can be done in a simple way using a loop. There is no ready to use thresholding function but creating a thresholding function is straightforward.

An important note to add about the thresholding is that, although the results are relatively good, in this project the standard deviation of aspect was used. This was done to diminish the problem of having a large difference in aspect when looking north. When classifying aspect, all values between 0-22.5 degrees and 337.5 – 360 degrees are classified as being north. The standard deviation calculated over the aspect is in this case not trustworthy. A solution is to use the cosine (northing) and sine (easting) of the aspect, to translate circular data into linear data. This way, two variables from the aspect both ranging from -1 to 1 can be combined to get a non-distorted aspect measurement, where (1,1) equals NE and (-1, -1) equals SW. These two variables can be implemented in the thresholding. This approach is common in ecological modelling, such as generalised linear modelling (MacLeod et al., 2008) and should be effective. However, this has not been tested by BeShore.

Random forest

The implementation of a random forest algorithm is straightforward due to existing functions and packages. However, computationally it is less efficient than thresholding and the histogram gradient boosting, but more efficient than regular gradient boosting. Despite this, model training can take a significant amount of time depending on the amount of input data and parameters to be experimented with.

During experimentation, initial accuracy metrics based on the randomly split test data (from the same chunk as the training data) are promising, with rubble stone precision and recall seeing a trade-off but averaging around 0.65. However, when validating these scores dropped, particularly the rubble stone precision.

Generally, it was observed that random forest tends to overestimate. This might be remedied by training with more data or more detailed hyperparameter tuning.

Gradient boosting

Gradient boosting produces good accuracy scores. The algorithm is slow with because of the limitations in computational power during the project. This made experimenting with parameters difficult.

The main drawback of the gradient boosting method is its runtime, this is because gradient boosting does not allow for parallel processing. As mentioned above, this made it hard to experiment with different parameters and it was not possible to complete a proper grid search in time. The results with the chosen parameters were promising.

Histogram based gradient boosting

The histogram based gradient boosting classifier (HGBC) has a low precision and thus classifies a lot of non-rubble stone as rubble stone. While thresholding, random forest and gradient boosting suffer from the same misclassifications, it is a more prevalent issue in the classification made by the HGBC.

Histogram based gradient boosting is significantly faster than the other tested machine learning algorithms. The HGBC allows for parallel processing which explains the increased efficiency compared to regular gradient boosting. The implementation of the model is straightforward. There are ready to use packages and it is very similar to the implementation of the random forest and regular gradient boosting.

Support vector machine

The support vector machine has not been validated for rubble stone detection due to complex implementation and computational limitations. The support vector machine is significantly more complex to run than the random forest and boosting algorithms. The support vector machine has potential but requires segmentation and more computational power than was available during the project. In short, the support vector machine had too many drawbacks to be considered for the scope of this project.

Post-processing

The output of the classifications contains a lot of loose rubble stone patches and gaps in the rubble stone patches, which makes it hard to determine a clear boundary line. It makes sense to clear the small rubble stone patches, because the rubble stones should be present in large patches only.

The function that fills gaps in the rubble stone fraction was made to already make the raster simpler. The function does not fill all the gaps. Trying different window sizes might solve this.

Applying the two buffers simplifies the classification polygon by erasing small rubble stone patches, but not all small patches are being erased. Additionally, the buffers make the gaps in the bigger rubble stone polygons bigger. An advantage is that buffer values could be varied to find the best line. More values for the buffer sizes could be tried to find the best possible line.

Creating an accurate boundary based on the classified raster is important since it can have a significant effect on the accuracy of the final result.

6 Conclusion

6.1 Methodologies to determine the boundary

This research started off with searching for methods that have potential to improve the accuracy of boundary detection. The following methods have been tested:

- Thresholding
- Random forest
- Gradient boosting
- Histogram gradient boosting
- Support vector machine

Gradient boosting is the method with one of the highest accuracy scores, but has a long runtime which makes it a more impractical method. The threshold method is simple to implement, but it is difficult to find the best parameters. Thresholding got the highest precision scores, and gradient boosting the highest recall scores. Random forest and histogram gradient boosting did not show equally high accuracies as the other methods. The support vector machine turned out to be a complex method due to segmentation which was necessary to make it computationally feasible.

Spectral analysis has potential to contribute to the classification. The question about whether shape uniformity can be used has not been answered in this research, as due to time constraints object based classification or deep learning methods were not explored. Machine learning methods require accurate input data in order to get good results. Currently, the predicted boundary does not fall within the outlined accuracy requirements. To improve boundary accuracy, ground truth training data sets should be improved to a higher quality, and if needed additional datasets containing extra variables could be used for model training.

6.2 Scalability of the methods

Thresholding is a difficult to scale method. When trying to classify multiple areas, the range of values to classify all rubble stone will become larger, because characteristics differ over different areas. This will cause under- or overclassification, depending on the area and range. The machine learning methods are easy to scale. To ensure scalability, the machine learning algorithms must be trained on accurate and representative data of all areas.

7 Recommendations

7.1 Model recommendations

The most promising model investigated in this study is the gradient boosting model, because of a relative high precision (Chapter 5.2, Table 2). This precision was achieved despite not being able to completely run the model with a grid search. The advice is to do a more intensive grid search to find parameters that yield a higher precision.

Based on preliminary research by Antea Group, the thresholding method was investigated further. This method results in a relatively high precision as well (Chapter 5.2, Table 2, Table 3, Table 4). There are still ways to improve this method, as discussed in Chapter 7.4. The final advice is to use gradient boosting. Despite this method showing slightly worse results compared to thresholding, it is scalable and has not yet been tested extensively. With increased quantity and quality of training data, and parameter tuning through a grid search, gradient boosting will likely produce higher accuracy results.

Random forest and histogram gradient boosting showed a lower precision than thresholding and gradient boosting. The advice is to prioritise gradient boosting over other machine learning methods due to the promising results it shows.

Due to the complexity of validating a model trained on segments, the support vector machine method was not further developed during this project. This method could be further developed in further research as it shows potential for the purpose of land cover classification (Petropoulos et al., 2012; Robertson & King, 2011).

Further investigation into the use of deep learning techniques for the classification of rubble stone is recommended as it has shown potential in other research (Vali et al., 2020; Zhang et al., 2020).

7.2 Computation power limits

The main obstacle in this research was the limitation in computational power as seen in Appendix IV. Fitting the models was time intensive which lead to difficulties experimenting with parameters. In combination with time limitations, decisions had to be made on what was investigated and to what extent. The best way to deal with this problem is to use a fast computer or working server.

7.3 Training data

To start with, it is necessary to increase the quantity and quality of training data for the machine learning models. This will likely result in an increase in the validation accuracy scores, as it helps overcome the problem of overfitting. Functions created in this research, like mosaic_rasters can be used to create a larger amount of training data.

Furthermore, it is recommended to use higher quality training data. The use of the DTB to train the models is debatable because, as previously stated, the DTB is known to contain inconsistencies. This study recommends training the models on more recent, high-quality datasets instead (and creating newer datasets if needed), since these should be more accurate.

Based on the analysis in Chapter 5, training the machine learning models on RGB data is advised. Using the differences in colours and spectral uniformity, could improve the accuracy of the classification.

7.4 Properties and parameters

The standard deviations of slope, aspect and roughness are used for classification. Other variables can be tried, such as spectral values and intensity. For the focal statistics calculation, only one kernel size is tried. It can be interesting to experiment with different kernel sizes to see how this influences the classification.

Experimenting more with the model parameters, for example by extensive grid searches, shows promise to further increase the accuracy of the models and thus the accuracy of the classification boundary. The higher the computational power, the easier this process is.

For using aspect or the standard deviation from aspect in thresholding, it is useful to use the cosine (northing) and sine (easting) instead of just aspect. This way the distortion of the values can be prevented, and the thresholding can be improved.

7.5 Hand annotation

The option to create the rubble stone boundaries by hand annotation according to a clear protocol should still be kept in mind as a backup option. Hand annotation should be done based on the orthophotos created from the point clouds. This means that it should be done for each inspection cycle, and that it is not scalable to different target areas.

8 Bibliography

Al-Najjar, H. a. H., Kalantar, B., Pradhan, B., Saeidi, V., Halin, A. A., Ueda, N., & Mansor, S. (2019). Land Cover Classification from fused DSM and UAV Images Using Convolutional Neural Networks. *Remote Sensing*, *11*(12), 1461. https://doi.org/10.3390/rs11121461

Belgiu, M., & Drăguţ, L. (2016). Random forest in remote sensing: A review of applications and future directions. *ISPRS Journal Of Photogrammetry And Remote Sensing*, *114*, 24–31. https://doi.org/10.1016/j.isprsjprs.2016.01.011

Cañete-Sifuentes, L., Monroy, R., & Medina-Pérez, M. A. (2021). A Review and Experimental Comparison of Multivariate Decision Trees. *Ciudad López Mateos: Sotirios Goudos*.doi:10.1109/ACCESS.2021.3102239

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors*). Annals Of Statistics*, 28(2). https://doi.org/10.1214/aos/1016218223

Fruend, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 119-139. doi:https://doi.org/10.1006/jcss.1997.1504

Haasnoot, M., Diermanse, F., Kwadijk, J., De Winter, R., & Winter, G. (2019). *Strategieën voor adaptatie aan hoge en versnelde zeespiegelstijging: Een verkenning* (No. 11203724–004). Deltares. Retrieved June 19, 2024, from https://publications.deltares.nl/11203724_004.pdf

Hang, H., Huang, T., Cai, Y., Yang, H., & Lin, Z. (2021). Gradient Boosted Binary Histogram Ensemble for Large-scale Regression. *arXiv* (Cornell University). https://doi.org/10.48550/arxiv.2106.01986

Krivoguz, D., Chernyi, S. G., Zinchenko, E., Silkin, A., & Zinchenko, A. (2023). Using Landsat-5 for Accurate Historical LULC Classification: A Comparison of Machine Learning Models. Data. doi:10.3390/data8090138

MacLeod, C. D., Mandleberg, L., Schweder, C., Bannon, S. M., & Pierce, G. J. (2008). A comparison of approaches for modelling the occurrence of marine animals. *Hydrobiologia*, *612*(1), 21–32. https://doi.org/10.1007/s10750-008-9491-0

Nationaal georegister. (n.d.). https://www.nationaalgeoregister.nl/geonetwork/srv/dut/catalog.search#/metadata/a301ddc7-c26f-42d8-b367-509ae5ae47d0

Natekin, A., & Knoll, A. C. (2013). Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, 7. https://doi.org/10.3389/fnbot.2013.00021

Petropoulos, G. P., Kalaitzidis, C., & Vadrevu, K. P. (2012). Support vector machines and object-based classification for obtaining land-use/cover cartography from Hyperion hyperspectral imagery. *Computers & Geosciences*, *41*, 99–107. https://doi.org/10.1016/j.cageo.2011.08.019

Rijkswaterstaat & Ministerie van Infrastructuur en Waterstaat. (2024). *ProductSpecificaties DTB*.

Robertson, L. D., & King, D. J. (2011). Comparison of pixel- and object-based classification in land cover change mapping. *International Journal of Remote Sensing*, *32*(6), 1505–1529. https://doi.org/10.1080/01431160903571791

Sahoo, P., Soltani, S., & Wong, A. (1988). A survey of thresholding techniques. *Computer Vision, Graphics, And Image Processing*, *41*(2), 233–260. https://doi.org/10.1016/0734-189x(88)90022-9

Salcedo-Sanz, S., Rojo-Álvarez, J., Martínez-Ramón, M., & Camps-Valls, G. (2014). Support vector machines in engineering: an overview. *WIREs Data Mining and Knowledge Discovery*, 4(3), 234-267. doi:https://doi.org/10.1002/widm.1125

Shapiro, L. G., & Stockman, G. C. (2001). *Computer Vision*. Pearson.

Sohl, M. A., Mahmood, S. A., & Rasheed, M. U. (2024). Comparative performance of four machine learning models for land cover classification in a low-cost UAV ultra-high-resolution RGB-only orthomosaic. *Earth Science Informatics*. https://doi.org/10.1007/s12145-024-01318-2

Vali, A., Comai, S., & Matteucci, M. (2020). Deep learning for land use and land cover classification based on hyperspectral and multispectral earth observation data: A review. *Remote Sensing*, *12*(15), 2495. https://doi.org/10.3390/rs12152495

Zhang, X., Han, L., Han, L., & Zhu, L. (2020). How well do Deep Learning-Based methods for land cover classification and object detection perform on high resolution remote sensing imagery? *Remote Sensing*, *12*(3), 417. https://doi.org/10.3390/rs12030417

Appendices

Appendix I: Data management plan

Refer to the attachment BeShore Data Management Plan.pdf

Appendix II: Overview parameters accuracy metrics

Thresholding:

Table 5 - Overview of parameters and respective accuracy metrics from thresholding model experimentation.

Results in grey were experimented with on the DTB ground truth data. Accuracy metrics thus do not represent metrics on the eventual validation set, and thus final accuracy results might be lower than metrics displayed in this table*.*

Parameters	Rubble stone precision	Rubble stone recall
Down_slope = 0.007, Up_slope = 0.33, Down_roughness = 3, Up_roughness = 26	0.31	0.76
Aspect_std = 75, Roughness_std = 0.033, Slope_std = 10, height = 53	0.59	0.77
Aspect_std = 80, Roughness_std = 0.033, Slope_std = 5	0.6378	0.4606
Aspect_std = 95, Roughness_std = 0.033, Slope_std = 10	0.63	0.49
Aspect_std = 82, Roughness_std = 0.033, Slope_std = 10	0.60	0.68

Random forest:

Table 6 - Overview of parameters and respective accuracy metrics from Random Forest model experimentation.

These parameters were experimented with on the DTB ground truth data. Accuracy metrics thus do not represent metrics on the eventual validation set, and thus final accuracy results might be lower than metrics displayed in this table*.*

Parameters	Rubble stone precision	Rubble stone recall
Trees = 100, Weights = {0:2, 1:8}, subset of chunk 1, first 400 thousand pixels	0.59	0.72
Trees = 100, subset of chunk 1, first 400 thousand pixels	0.74	0.59
Trees = 100, subset of chunk 1, first 800 thousand pixels	0.73	0.59
Gridsearch with hypertuning parameters = {‘n_estimators’: [50, 100, 150], ‘max_depth’: [5, 10, 15], ‘max_features’: [‘sqrt’, ‘log2’, None]}, subset of chunk 1, first 200 thousand pixels	0.75	0.58
Final version (validated on validation set instead of DTB): Trees = 100, features = ‘sqrt’, weight = ‘balanced’, all data	0.22	0.68

Gradient boosting:

Table 7 - Overview of parameters and respective accuracy metrics from gradient boosting model experimentation.

All gradient boosting was done with the parameter grid of n_estimators = 50, max_depth = 5, max_features = 'sqrt' (square root of number of features)

Parameters	Rubble stone precision	Rubble stone recall
Tested on DTB		
Subset of chunk	0.6881	0.3913
Whole of chunk	0.6884	0.3870
Tested on validation raster		
Whole of chunk (best result)	0.59	0.85

Histogram gradient boosting

Table 8 - Overview of parameters and respective accuracy metrics from histogram gradient boosting model experimentation.

During all models the following parameters were used early_stopping='auto', and class_weight='balanced'. The grey highlighted part of the table was validated on the DTB Chunk 6 of phase 1.

Parameters	Rubble stone precision	Rubble stone recall
l2_regularization: 0.1, learning_rate: 0.2, max_depth: 10	0.38	0.77
l2_regularization: 0.01, learning_rate: 0.5, max_depth: 3	0.39	0.77
learning_rate: 0.2, max_depth: 5	0.38	0.77
l2_regularization: 0.1, learning_rate: 0.5, max_depth: 5, max_features: 0.3	0.18	0.97
l2_regularization: 0.1, learning_rate: 0.1, max_depth: 3, max_features: 0.5	0.26	0.99
l2_regularization: 0.1, learning_rate: 0.1, max_depth: 3, max_features: 0.3, max_iter: 75}	0.29	0.99
Best estimator: l2_regularization: 0.1, learning_rate: 0.01, max_depth: 3, max_features: 0.3, max_iter: 75}	0.31	1

Post-processing

Table 9 - Overview of parameters from boundary line experimentation and tried values.

thres_value in fill_gaps [-]	buffer_distance in buffer_geojson [m]	buffer_distance in buffer_geojson_multi [m]
0.1	0.5	0.1
0.5	1.0	0.5
	1.5	1.0
	2.0	1.5

Appendix III: Workflow creating boundary metrics

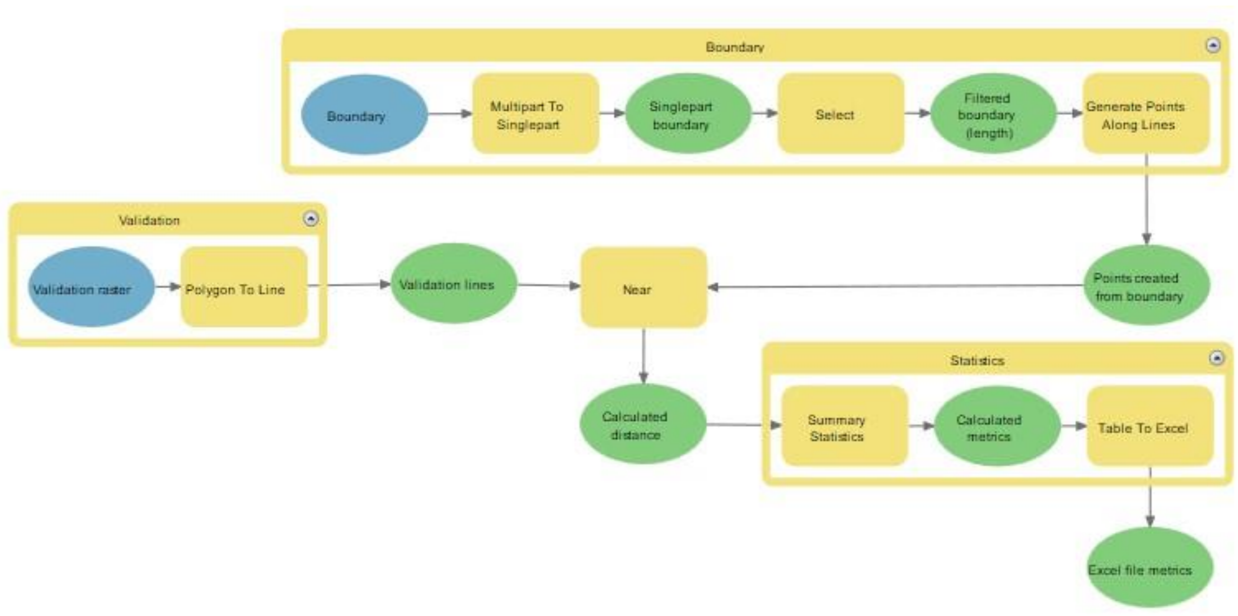

Figure 14 - Workflow for creating boundary metrics using ArcGIS

Appendix IV: Computational specifications

Listed below are the laptop specifications for each of the project members

Jelmer Sonnemans

Processor Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz

Installed RAM 16.0 GB (15.8 GB usable)

NVIDIA Quadro P2000 GPU 12 GB

Nik Verweel

Processor 11th gen Intel(R) Core(TM) i7-11370H CPU @3.30 GHz

Installed RAM 16.0 GB

GPU NVIDIA GeForce RTX 3050 Laptop

Lennart Murk

Processor AMD Ryzen 7 5800H with Radeon Graphics, 3.20 GHz, 8 Core(s), 16 Logical Processor(s)

Installed Physical Memory (RAM) 32,0 GB

NVIDIA GeForce RTX 3060 Laptop GPU

Milou Maathuis

Processor 13th Gen Intel(R) Core(TM) i7-1355U 1.70 GHz

Installed RAM 16.0 GB

GPU Intel(R) Iris(R) Xe Graphics

Sicco de Jong

Processor: 13th Gen Intel(R) Core(TM) i7-1355U 1.70 GHz

Installed RAM 16.0 GB

GPU Intel(R) Iris(R) Xe Graphics