Multi-Data Approach for Subsurface Imaging: Combining Borehole and GPR- Data for Improved Analysis

Pontus Yngvesson

A thesis submitted for the degree of
Master of Science in Engineering Physics
Thursday 15th June, 2023
Abstract

The investigation to understand the subsurface and its features has long been a subject of interest for various fields, including fields such as archaeology and infrastructure projects. However, traditional excavation methods are often costly and time-consuming. In their place, alternative techniques such as borehole drilling, which is itself expensive, and ground-penetrating radar (GPR), which produces a good but distorted image, have gained popularity. Nonetheless, the limitations of each method impede them from meeting the requirements of subsurface exploration. This Master’s thesis introduces an approach combining these two methods to overcome their limitations and enhance their accuracy to understand the subsurface.

This thesis aims to demonstrate the feasibility and effectiveness of integrating borehole drilling and GPR for subsurface exploration. Specifically, the integration of borehole with GPR-profiles will be examined to enhance their practicality and accuracy, meaning that this thesis will investigate the utilization of borehole data to update and adjust GPR-profiles, thereby providing more precise and informative data for further analysis.

The findings of this work indicate that combining borehole drilling and GPR-profiling to improve and update the accuracy of the GPR-profiles is entirely feasible and results in a substantially improved subsurface exploration capability. Further, the outcomes of this thesis suggest that the integrated approach can generate a more precise representation of the underground structure. Ultimately, the proposed integration of borehole drilling and GPR-profiling presents a promising approach to enhance the accuracy and efficiency of subsurface exploration and has the potential to be valuable in a wide range of fields.
Acknowledgement

I would like to express my heartfelt gratitude to several individuals without whom this thesis would not have been possible. First and foremost, I would like to thank Rickard Sjödin and André Lilja for their invaluable help and guidance with regard to MALÅ Vision and programming in the various languages. Their unwavering support throughout this project has been instrumental in ensuring its success. I would also like to express my appreciation to my supervisor, Anders Abrahamsson, for his insightful ideas and valuable feedback, which helped shape and refine my research. I am grateful to Johan Friborg for his advice and guidance in developing the layer-picking algorithm and the Octave script, which formed the basis of my version in Python. I would also like to thank Jaana Gustafsson for her help and support in helping me to understand boreholes and their applications. Finally, I would like to extend my heartfelt thanks Victor Jonsson and all the other people at Guideline Geo who supported me throughout this project. Their help, encouragement, and support were critical in ensuring that this project was completed successfully.
# Contents

Abstract

Acknowledgement

1 Introduction
   1.1 Background .................................................. 1
   1.2 Proposed Approach ............................................. 3
   1.3 Objective ..................................................... 4

2 Theory
   2.1 GPR - Ground Penetrating Radar ............................... 5
   2.2 Boreholes ...................................................... 6

3 Method
   3.1 Borehole importation ........................................... 7
      3.1.1 SND-files .................................................... 7
      3.1.2 COR-files .................................................... 8
   3.2 Positioning ..................................................... 9
   3.3 Visualization of borehole ...................................... 10
   3.4 User information .............................................. 11
   3.5 Combining boreholes with GPR-profiles ......................... 11
      3.5.1 Layer-boundary picking .................................... 11
      3.5.2 GPR-profile adjustment .................................... 14

4 Result
   4.1 Importation of boreholes ....................................... 18
   4.2 Visualization .................................................. 18
   4.3 GPR-profile adjustments ....................................... 19

5 Discussion
   5.1 Future recommendation ......................................... 28
1

Introduction

1.1. Background
Understanding what lies beneath the Earth’s surface is of great importance in various contexts, including infrastructure projects, archaeology, and studying ice thickness.

Infrastructure projects often involve construction and development activities that require a comprehensive knowledge of the underlying geology, soil properties, and groundwater conditions. This is crucial for designing stable foundations, determining the feasibility of tunnels and underground structures, and assessing potential risks such as sinkholes or contaminated soil. By examining the subsurface, engineers, and planners can make informed decisions, minimize uncertainties, and ensure the safety and long-term sustainability of infrastructure projects [1] [2].

In the field of archaeology, the subsurface holds a wealth of historical information. Uncovering buried artifacts, ancient structures, and archaeological features provides insights into past civilizations, their way of life, and cultural evolution. Excavating the subsurface with precision is essential to preserve delicate artifacts and establish the historical context accurately [3].

Another area where subsurface exploration is essential is in understanding ice thickness. Climate change and its impact on ice formations, such as glaciers and ice sheets, have drawn significant attention. Monitoring changes in ice thickness over time is crucial for assessing the health of these icy environments and predicting their contribution to sea-level rise [4] [5].
GPR allows for non-destructive imaging of the subsurface, providing continuous profiles of the geophysical properties along survey lines [6]. But, despite its widespread use and effectiveness in identifying subsurface features, it is important to note that GPR-data is not infallible, as it can only detect changes in permittivity between two materials. It can not determine the exact nature of the materials present beneath the surface. Furthermore, a significant limitation of GPR is the ambiguity of the depth-/y-axis of a GPR-profile. This axis represents the time taken for the electromagnetic pulse to be reflected and returned from the subsurface. Since different materials exhibit varying velocities, the depth in meters can not be determined precisely and necessitates a substantial degree of approximation [7] [8].

Boreholes, on the other hand, provide direct sampling and measurement of geological and environmental parameters, offering high-resolution information at specific locations [2]. It is however very expensive and time-consuming to make and analyze them [9]. Discrepancies between the actual depth of the bedrock and the approximation may lead to delays and additional costs during the excavation process, as illustrated in figure 1.1. In the left figure, we see how the bedrock depth is approximated from the depth in 5 boreholes, and on the right, we can see the problem with that approach where the boundary for the bedrock has been added showing that the bedrock went higher then approximated between borehole 2 and 3.

Therefore, it is important to consider the limitations and uncertainties associated with borehole-data when using them for subsurface characterization.

By combining these two techniques, researchers can obtain a comprehensive understanding of subsurface structures, composition, and variations, contributing to
more accurate interpretations and informed decision-making. Today this is done by combining the two methods in exactly this way, putting the boreholes on a GPR-profile, but if we again look at the right figure of figure 1.1 we can see the limitation of this. We can see that the layer-boundary of the bedrock is not at the same depth as the bedrock depth obtained from the boreholes.

In this master thesis, I explore the integration of borehole-data and GPR-profiles to develop a multi-data approach for subsurface imaging. By combining the strengths of both techniques, I aim to improve the analysis and interpretation of subsurface conditions for all of these fields. This research contributes to the growing body of knowledge in subsurface exploration, demonstrating the value of integrating different data sources and advancing our understanding of what lies beneath the Earth’s surface.

Overall, the investigation of the subsurface in various fields holds immense importance. Through the use of boreholes and GPR, we can delve into the hidden depths, unraveling the secrets of the Earth’s subsurface and unlocking valuable insights that inform infrastructure projects, archaeology, and climate research, among others. By combining these tools and techniques, we can bridge the gap between what is seen and what lies beneath, ultimately leading to more informed decision-making and a deeper understanding of our planet.

1.2. Proposed Approach

In the context of geological investigations, the integration of borehole- and GPR-data can offer valuable insights into subsurface conditions. As illustrated in figure 1.2, the combination of borehole-data with GPR-profiles enables the correlation of geological layers visible on the GPR-data profiles, with material changes identified through borehole sampling [10]. While GPR-data can reveal layer-boundaries, it does not provide information on the nature of the layers and the boundaries themselves such as material, and it also does not give the depth in meters and instead gives it in time for the wave to travel back and forth. However, by examining the corresponding borehole, it becomes possible to discern the boundaries depths in meters, and this can be used to adjust the GPR-profiles to make the layer-boundaries align with the boreholes layers, as can be seen on figure 1.2. This gives a much better and more correct picture of the subsurface.
1.3. Objective

In my master’s thesis, my objective was to develop a method for integrating borehole-data with GPR-profiles in MALÅ Vision, a software commonly used for visualizing GPR-profiles, to provide a more detailed understanding of the subsurface. This was done by importing boreholes, visualise them together with GPR-profiles, and then adjust the profiles, as shown seen on figure 1.2, by using the knowledge that the boreholes give a better view of the layer-boundaries depths.
2

Theory

2.1. GPR - Ground Penetrating Radar

Ground Penetrating Radar, GPR, is a geophysical technique that involves the use of an electromagnetic pulse-emitting device mounted on, for example, a cart. The cart can be pushed along the surface of the ground to investigate the subsurface features of an area. As the electromagnetic pulses travel through the ground at different velocities, depending on the materials they encounter, they reflect back from permittivity boundaries between different materials [11]. These reflections produce wave-like patterns and lines, as illustrated in figure 2.1 [12]. The velocities of some common materials are listed in table A.1 [13].

GPR has gained widespread popularity as a versatile method used in various fields, such as civil engineering, archaeology, and glaciology. It is highly effective in detecting hidden objects and utility systems such as pipes and cables, which are crucial in avoiding potential hazards and disruptions during construction projects. In addition, GPR is highly useful in identifying archaeological features and potential points of interest in archaeological sites.

Overall, GPR is a valuable tool for a wide range of applications, providing non-destructive and accurate subsurface information. Its effectiveness in identifying hidden structures and materials has made it an indispensable tool in many industries.
Figure 2.1: An example of a GPR-profile, where a boundary between two layers of subsurface materials has been marked with a red line. The y-axis shows the time it took for the wave to travel down in the subsurface, get reflected, and then travel back up into the receiver. The x-axis shows the traces, where each trace is one measurement done when the GPR-unit has traveled along the ground.

2.2. Boreholes

Borehole drilling is a widely utilized method for investigating subsurface conditions. This technique involves drilling holes using various types of drills or manual excavation to gather valuable information about the specific location’s subsurface.

In practical terms, the process begins with the drilling of a borehole to a desired depth. As the borehole is drilled, soil- or rock samples are extracted and collected as core samples. These core samples are crucial for analyzing the properties of each subsurface layer. In some cases, sensors are deployed down the borehole to gather additional data, while in others, analysis is performed manually on the extracted cores. By examining the core samples and utilizing specialized techniques, engineers and geologists can determine various properties, such as layer composition, soil characteristics, and even the presence of groundwater.

In summary, borehole drilling plays a vital role in understanding subsurface conditions. Through the acquisition and analysis of core samples, engineers and geologists can obtain essential data for informed decision-making in various civil engineering projects, including road construction, infrastructure development, and geological investigations.
3

Method

3.1. Borehole importation

The first thing that needed to be done was to import borehole files. These come in many different formats, and I chose to implement two of them COR and SND. They are two of the more common types used in Sweden and the rest of the Nordic countries.

In MALÅ Vision, I created a new custom class called 'Borehole' specifically designed for TypeScript/JavaScript. This class is responsible for encapsulating all the necessary information from the borehole files. In TypeScript, a script first checks what type of file we have, SND or COR, and depending on the type, the corresponding functions were called.

3.1.1. SND-files

The SND-files contain a lot of information, but I only needed a specific part of the file here. To extract the relevant information, I created a filter function. First, I split the data into two parts; the first one contains the first 15 rows which contains general information about the borehole, while the second one got the rest of the data, including one or more analyses for that specific borehole. The SND-files I used only had one type of analysis, a JB2 analysis, but if many analysis would exist then each new analysis is indicated by a '*' symbol at the start and the end. Each new analysis
also starts with two rows of information about the date and analysis type. We can use this knowledge to filter out the wanted borehole information, but this has not yet been implemented.

From the first 15 rows, I chose to save the coordinates, height, borehole-ID, and tilt of the hole. For the second part, the part with the analysis, it is possible to extract many different types of information from the columns, but I chose to save the depth, stop codes, and depths of the layer-boundaries. The stop codes are codes that describe the type of material encountered during drilling. For example, if the number 41 is in the stop code column for a curtain depth, that means ”large stone drilled through”. In the same way, 80 means ”bedrock”. In the stop code column, comments may also be added, such as the drilling crew if they notice something needed to be added.

3.1.2. COR-files
COR-files comes in two types; one containing boreholes and one containing the coordinates for a GPR-profile. The borehole COR-files are simpler to import then the SND-type since they contain less information. These files are designed to be paired with a GPR-profile and can have one or more boreholes. Each borehole starts with the string 'Core' followed by a running number that indicates which borehole it is. The next row contains a trace-number that indicates the location of the borehole on the corresponding GPR-profile. The file ends with a list of all the layer-boundaries found when drilling, which includes the layer-boundaries, each with their own serial number, and the depth in meters at which they were found.

To process each COR file, we first check if there is more than one value on the first row. If there is, it is not a borehole file but instead contains the coordinates for each trace position on the GPR-profile. If there is only one value, I assume that it is a borehole, and search for rows starting with 'Core'. When one is found, the next row’s value is saved as the position, and the following rows until a new row that starts with 'Core' is found, or the document ends, are saved as the layer-boundaries.

To position the boreholes, we search for the nearest trace-number in the GPR-profile’s COR-file. The reason for this is because not every trace-number do exist. We then place the borehole at the same position using the coordinates that the corresponding GPR-profile trace-number used.
3.2. Positioning

To be able to position the boreholes correctly I converted the coordinates to something that would be possible to plot in MALÅ Vision. Since their GPR-profiles were positioned by converting the coordinates into meters relative to a starting point, I chose to do the same. I did this by converting them from geographic coordinates to Mercator projections. Mercator projection is a cylindrical map projection that preserves the shape of small areas and allows for straight lines of constant true direction. However, it distorts the size and shape of areas at high latitudes, for example making Greenland appear larger than Africa [14]. In my case, this was not a problem.

To convert geographic coordinates to Mercator projection, I used the following equations:

Let $r_e$ be the Earth’s radius in meters and $lat_{ref}$, $lon_{ref}$, and $h_{ref}$ be latitude, longitude, and height over sea level in meters. First, I converted the latitude and longitude to radians;

$$lat_{rad} = \frac{lat_{ref} \pi}{180}$$

$$lon_{rad} = \frac{lon_{ref} \pi}{180}$$

after which I calculated the $x$, $y$, and $z$ coordinates of the reference point:

$$x_{ref} = r_e lon_{rad}$$

$$y_{ref} = r_e \log(tan(\frac{\pi}{4} + \frac{lat_{rad}}{2}))$$

$$z_{ref} = h_{ref}$$

With this done I then converted the geographic coordinates for a given point, where $lat$, $lon$, and $h$ are the points latitude, longitude, and height over sea level, to Mercator
projection by:

\[ \begin{align*}
    \text{lat}_\text{rad} &= \frac{\text{lat} \pi}{180} \\
    \text{lon}_\text{rad} &= \frac{\text{lon} \pi}{180} \\
    x &= r_e \text{lon}_\text{rad} \\
    y &= r_e \log(\tan\left(\frac{\pi}{4} + \frac{\text{lat}_\text{rad}}{2}\right)) \\
    z &= h
\end{align*} \]

Then, the differences in x-, y-, and z-coordinates between the reference point and the given point can be calculated by subtracting the reference point from the point:

\[ \begin{align*}
    \Delta x &= x - x_{\text{ref}} \\
    \Delta y &= y - y_{\text{ref}} \\
    \Delta z &= z - z_{\text{ref}}
\end{align*} \]  \hfill (3.1)

Now, by using \( \Delta x, \Delta y, \) and \( \Delta z \) gotten from equation (3.1) it is possible to position all boreholes correctly, relative to the reference point, which in this case is the first GPR-profiles first trace position, or the first borehole if no GPR-profiles have been added.

### 3.3. Visualization of borehole

To visualize the boreholes, a function was created that uses the layer-boundaries depths to make cylinders for each layer and position them at the correct depths. It works by taking a position and an array with all the layer-boundaries from the file. In that position, it then creates a cylinder that starts at a depth of 0 m and ends at the first depth value from the layer-boundaries array. The next cylinder is then created beginning from the last layer-boundary value and ending at the next value, etc. All of these cylinders get an individual color which symbolizes the material between the layer boundaries. Note that the colors are arbitrary, only showing that we have a new layer.

The creation of the layer-arrays, the arrays that contains the layer-boundaries, is done differently for COR- and SND-files.
For the SND-files, the column with the stop codes is used. A script goes through each row of the column, and for each unique value, a layer is made.

When it comes to the COR-files, the layer-boundaries depth is the information given in the file. This makes the list of layers easy to extract, by first finding the start of a new borehole, which is done by finding a row that starts with the string 'Core', then extracting the values from each row until a new row with starting string 'Core' comes. Each value extracted corresponds to the depth of a layer-boundary, and for every such interval between two 'Core' a new borehole is created.

3.4. User information
To make it easy for the user to understand what the borehole and the data connected to it show, I implemented an information “text-box” above each borehole, letting the user choose what information to see. The choices implemented are; layer-thickness, layer-boundary depths, layer-velocities, and comments from the borehole-files. The text-box is created such that it follows the camera in MALÅ Vision, making the text turn around itself to always be pointed toward the user.

3.5. Combining boreholes with GPR-profiles
To adjust the GPR-profiles, a few steps needed to be completed. Initially, I decided to implement these adjustments using Python since I have some previous knowledge of it, and since it is already utilized in MALÅ Vision. Despite that, I was unable to establish communication between the Python-component and the rest of MALÅ Vision. As a result, I chose to divide them into two parts and manually transfer the necessary data between them.

The process of adjusting the GPR-profiles was separated into two main stages, layer-boundary picking, and profile-adjustments.

3.5.1. Layer-boundary picking
The GPR-profiles are adjusted according to layer-boundary depth read from the boreholes. The first step is to locate and pick the same number of layer-boundaries in the profiles as there are in the borehole imported. This is done by letting the user try to identify a layer-boundary and pick it on a figure with the GPR-profile shown. From this point, an algorithm tries to follow it in both directions to the edges.
The position on the figure is saved when the user clicks with the left mouse button and the x- and y-coordinates are then used as the starting point for the algorithm. Pseudocode of this algorithm is shown in Algorithm 1.

Algorithm 1 Layer-boundary follower

```plaintext
function AnalyzeProfile(X, Y, layers, index_max, profile)
    Let X be the user-picked starting value for the length along the profile
    Let Y be the picked starting value for the depth
    Let layers be a matrix where the layer-boundaries Y-value for each trace number are saved
    Set layers[X] ← Y
    while X < length(profile) − 1 do
        Let trace_array be the X-th row in profile
        Let search_window be ± 3 sample depths around Y
        Let sub_array be trace_array[Y − 3 : Y + 3]
        Let index_max ← Index of the maximum absolute value in sub_array
        Set layers[X + 1] ← index_max
        Let X ← X + 1
        Let Y ← index_max
    end while
    Let X be the user-picked starting value for the length along the profile
    Let Y be the picked starting value for the depth
    while X > 0 do
        Let trace_array be the X-th row in profile
        Let search_window be ± 3 sample depths around Y
        Let sub_array be trace_array[Y − 3 : Y + 3]
        Let index_max ← Index of the maximum absolute value in sub_array
        Set layers[X − 1] ← index_max
        Let X ← X − 1
        Let Y ← index_max
    end while
end function
```

If the layer-boundaries inside the GPR-profile are good and easy to follow, then that is that, but they are usually not that good since the subsurface is rarely homogeneous. That is why the user can make adjustments to the layer boundaries, and when the user thinks the boundary-line is good enough they save it to then start with the next layer-boundary, or end boundaries-picking if it was the last boundary.

The user adjusts the layer-boundaries by clicking on the GPR-profile to pick a new position. If a boundary has been picked before, the algorithm runs in both
directions from the new position, double checking if the saved boundary-line from
the last try includes any of the indexes in the next steps interval, then the algorithm
updates only the part of the boundary that has new values and stops. This allows
the user to identify any missed or skipped parts of the boundary by examining the
profile and adding them during a later attempt. If there is a part of a boundary
missing on the GPR-profile, or if the algorithm above can not follow it at all, a
function that records the user’s movements on the figure can also be done. When
the user holds the left mouse button down, the coordinates for the mouse-movements
will be saved as part of the boundary. Upon the user’s interaction of moving the
mouse over specific x-coordinates, all corresponding y-values within the previously
selected boundary will undergo replacement. The user can also click with the right
mouse button, which will then be the starting position for making a straight line
between two points. The user will be asked to click one more time to give the other
end-position of the line, but now with the left button. After that, a straight line will
be drawn between these two positions and added to the layer-boundaries matrix.

The last alternative for the user is to remove the last made layer-boundary to
start over from the beginning on that boundary. After the user is satisfied with the
layer-boundary they can then either click to finish that layer and start on the next
one, or they exit the layer-boundary picking. This can only be done when the number
of boundaries made and saved in the layer-boundary matrix is the same number as
the number of layer-boundaries inside the imported borehole.

To determine whether the user is clicking or holding down the mouse-button, the
program checks the number of x-values generated during the action. If the number
is less than 10, it is considered a click. However, if there are 10 or more values, it is
considered as the mouse button being held down and moved. This is because it is
nearly impossible for a human to click a mouse-button without even slightly moving
it.

Once the user has picked layer-boundaries that match the number of boundaries
in the imported borehole and pressed the ‘E’ key when prompted, the layer boundary-
picking process is completed and exited.

In certain cases, layers in the subsurface can overlap, with one layer extending
into another. In such cases, the layer boundary corresponding to the upper layer is
given priority, and the lower boundary is not allowed to cross it. Instead, the lower
boundary is set to the same depth as the upper boundary at that location.
3.5.2. GPR-profile adjustment

My goal with the adjustments was to adjust each layer between the layer-boundaries to align the GPR-boundaries with the borehole-boundaries. I looked up a number of methods and ultimately decided to narrow my focus on two particular ones that showed the greatest potential for achieving the desired outcome.

The first one is a method called Seam Carving [15]. It is a method that calculates a weight/density for each pixel on the figure, using a couple of different algorithms, mainly; gradient magnitude, entropy, visual saliency, and eye-gaze movement. Depending on the situation, a suitable algorithm is chosen. From the weights with the lowest values, seams that go through the entire picture are created. These seams are then ranked by their total weight, where low-weight seams have the lowest importance to the content of the figure. A new figure is then generated by removing seams with the lowest total weight, where the number of seams removed matches the number of pixels by which the picture is reduced. The objects in the figure with the highest weight, i.e. the ones that give the viewer the most information, stay without getting deformed. This is visually described below in figure 3.1 [16].
Chapter 3. Method

(a) Original picture that will be reduced along the x-axis.

(b) Show the parts of the picture with the highest values.

(c) The seams that have the lowest total sum is marked with red.

(d) Updated picture where the seams have been removed.

Figure 3.1: Shows how seam carving works, where each pixel on a figure gets a value, and then a seam is created that finds the path with the lower total sum. These are then removed, reducing the size without removing the parts of the picture with the highest value for the understanding of the picture.

The other method is a more basic one I made myself, and it is the one I chose to implement and use. It takes the original GPR-profile, the matrix with the picked layer-boundaries, and the borehole with its trace-number, i.e. position along the GPR-profile. A percentage difference between each layer-boundary on the GPR-profiles and the boundaries in the borehole is calculated by finding the difference in the number of y-values. This is done by using equation (3.2)

\[ \text{procDiff} = \frac{\text{nrSampGPR} - \text{nrSampBorehole}}{\text{nrSampBorehole}} \] (3.2)

The term \( \text{procDiff} \) refers to the percentage difference between the number of sample values obtained from the layer-boundaries in the GPR-trace, denoted as \( \text{nrSampGPR} \), and the borehole boundaries, represented by \( \text{nrSampBorehole} \). This is
done for each layer to know how much each GPR-profiles boundary has to be adjusted to make the borehole and GPR layer-boundary match up at the borehole position.

After the percentage difference is calculated, each trace in the GPR-profile is updated. The number of sample values, values along the y-axis, in each layer, is extracted and then multiplied by the percentage difference. The resulting value is the number of samples that need to be removed or added, \( \text{sampDiff} \).

To then make new layers, a copy of the old layer called \( \text{newLayer} \), is created. Secondly, the sign of \( \text{sampDiff} \) is checked to determine if samples will be added or removed. If it is negative then samples need to be removed, done by looping through and removing \( \text{sampDiff} \) number of samples from the \( \text{newLayer} \) with an equal distance between each removed sample. A loop that does \( \text{sampDiff} \) iterations with iteration variable \( i \) is used in equation (3.3) to extract the index, rounded to the closest integer, of the rows then get removed from \( \text{newLayer} \).

\[
\text{index} = \left\lfloor \frac{\text{len(newLayer)}}{|\text{sampDiff}| + 1}i \right\rfloor - 1
\] (3.3)

If the sign of \( \text{sampDiff} \) is positive, samples need to be added. A check is done to see if the number of samples that need to be added is more than the number inside \( \text{newLayer} \) from the beginning. If that is true then the maximum number of samples added will be \( \text{maxSamp} \), and the first loop will be equal to \( \text{newLayer}-1 \). A loop will add that number of values until the total number added is equal to \( \text{sampDiff} \). The positioning of the new sample values is done similarly to when values are removed, i.e. they are positioned by getting an index, dividing the size of the array by the number of values that will be added, and then looping through, starting from the last position. The value for each of these new values is the mean of the two values next to it. See equation (3.4), in which \( i \) is the iteration number that goes from 0 to the maximum number of samples possible to add each loop, and \( j \) is the iteration number for when \( \text{newLayer} \) is smaller than \( \text{sampDiff} \).

\[
\text{index} = \left\lfloor \frac{\text{len(newLayer)}}{\text{maxSamp} + 1} \cdot (j+i) \right\rfloor
\] (3.4)

To not change the total amount of sample values in each trace, the last step of the updating is to double-check that the total number of samples before and after is the same. This needs to be done because their may be a rounding error removing or adding a value extra when creating \( \text{sampDiff} \). If not, fill in/remove values to/from
the last layer so they match each other.

The velocity is approximated from the start to be 100 m/µs in MALÅ Vision, since this is close to the mean velocity in a normal grounds structure, [13]. This is of course not always right, and that is why an approximation for the velocity in each layer is also made. The percentage difference, \( \text{procDiff} \), is multiplied by the mean velocity, 100 m/µs, which gives a value, \( \text{layerVelocity} \), that is closer to the true value for the velocity in that layer.

A modification was added that uses the same equations as above. It removes all samples above the first layer-boundary. This is done mainly to remove the first “trash” data gotten from the wave that travels straight from the sender to the receiver, but it also removes the trash data gotten when the GPR-unit is not close to the ground such as with drones. It checks the number of samples that is in the first layer, removes them all, and then adds the same amount to the last layer, to not change the total number of samples.

To easier understand how all this is done, see algorithm 2

---

Algorithm 2 Profile Adjuster

```plaintext
procedure ProfileAdjustment(GPRProfile, boreholeBoundaries)

    Calculate procDiff using equation (3.2)
    for layer in GPRProfile do
        Let sampDiff be the difference in the number of samples for each layer
        if sampDiff < 0 then
            Remove sampDiff samples from layer according to equation (3.3)
        else
            Add sampDiff samples to layer according to equation (3.4)
        end if
    end for

    Calculate layerVelocity by multiplying 100 m/µs with procDiff
    Remove samples above the first layer boundary and add an equal number of samples to last layer

    return Adjusted GPRProfile

end procedure
```
4

Result

4.1. Importation of boreholes
In the already implemented importation tab in MALÅ Vision, given to me by Guideline Geo, it is now possible to import boreholes of the type COR and SND. They get successfully read and saved into the new class created called ”Borehole”, where the most important information about them is saved. A working filter also distinguishes between borehole COR-files and the GPR-profiles COR-files.

4.2. Visualization
It is now possible for the user to add the boreholes to the already shown GPR-profiles, and they are correctly positioned, no matter if it is by using coordinates gotten from the SND-file or if using the trace number gotten from the COR file and position them at the same trace numbers on the GPR-profile. Each layer of the boreholes is positioned at the correct depth too.

In figure 4.1 a GPR-profile that is taken with a drone over the ice in Sörfors, can be seen together with an imported borehole, in this case, done with an ice drill. So what we see is the thickness of the ice, but above the ice is an unknown height in meters.

Above each borehole information is shown that gives the user information about the boreholes layer thickness, the layer-boundary depths, the approximated velocity
in each layer, and any comment read in the SND-file. The user chose what information to view by clicking different buttons.

4.3. GPR-profile adjustments

I was able to make a working adjuster of GPR-profiles. In a Python-figure it lets the user choose layer-boundaries, which are then used to follow that boundary to both ends creating boundaries, figure 4.2.

If the boundary-line does not follow the path the user wants it to, the user clicks again. This is either done by clicking on a new position where the line should go and then letting the algorithm try to correct it, or the user clicks and holds the mouse button and draw the path the line should go, and then that will be the new and updated boundary line. On figure 4.3 the user has at the right arrow clicked, letting the automatic layer-follower algorithm adjust the boundary-line, and in the left circle the user has drawn a new line by itself. It is also possible for the user to remove the last placed boundary-line, either because there are too many picked lines or because the user wants to start over with the last line.

The steps above are true no matter the number of layer-boundaries needed to be identified. However, the number of layer-boundaries has to match between the number the user has picked and the number in the imported borehole, and if they do not then a message tells what number that is now picked, and how many the user should adjust the number by. When the number of layers in the borehole is the same as the number marked by the user, an automatic script adjusts the layer to make
Figure 4.2: In figure (a) the user clicks on the position of the arrow and in figure (b) we can see the result; a line that follows the boundary.

The boundaries align between the boreholes and the GPR-profile.

It can also completely remove the first layer, i.e. the layer above the first boundary line. The first layer is in most cases the return waves that travel straight from the transmitter to the receiver. But in some cases, for example when the data is taken with a flying drone, then the layer down to the ground is air and as a consequence useless, only making the subsurface information harder to understand. By removing
Figure 4.3: The line is not perfect after the first click, so the user clicks to update the line at the position of the arrow and then tries to correct the line inside the circle by drawing by hand. The resulting boundary-line can be seen in figure (b), and as can be seen, the user did not have a steady hand when drawing by hand.

that layer, a smooth ground level is made easier to read, as can be seen on figure 4.4. The gradient color change near the borehole is a visual representation telling us that the closer we are to the borehole the more sure we can be that the adjusted profiles show the truth. It is only a visual aid, not something calculated.
Figure 4.4: An original GPR-profile can be seen in figure (a), and after a borehole and the boundary-lines have been marked by the user we get figure (b). The resulting adjusted GPR-profile where the layer above the ground-boundary has been removed is shown in figure (c).
When trying to adjust the thickness of the ice seen on figure 4.4, the velocity in the ice layer was calculated to be 141 m/µs. This is within the interval of the true velocity which is 78-157 m/µs [13].

The resulting adjusted GPR-profiles, without the colors, are then possible to add to MALÅ Vision, visualizing the boreholes together with the updated profiles. In figure 4.5 we can see the updated version of figure 4.1, where the GPR-profiles first layer has been removed and the other layers have been adjusted to align with the depth of the borehole.

![Figure 4.5: The updated version of figure 4.1 GPR-profile, showing how the first layer is removed and the rest of the profiles layers is adjusted to align with the borehole.](image)

On figure 4.6 we can see the flowchart for working with the resulting codes from this project, showing how the data have to manually transferred between MALÅ Vision and the Python code.
Figure 4.6: Flowchart over the workflow for the resulting codes when adjusting a profile. Start by importing a borehole in MALÅ Vision to get the boundary depth, then follow the chart.
The project has yielded positive results overall, although understanding the workings of MALÅ Vision presented a challenge. This difficulty arose from the project’s requirement to employ multiple programming languages, including Python, Typescript, Javascript, and the Javascript package Three.js. Originally, the intention was to execute the entire project using Python and interface it entirely within MALÅ Vision. However, attempts to establish communication between the various languages proved time-consuming, and the version of MALÅ Vision available did not support a 2D-view, which was an essential component of the project. As a result, the approach was modified, and only importation and visualization were implemented using Typescript and Three.js while the 2D-view was moved to a Python figure window, thereby eliminating the need for communication between Python and other languages used in MALÅ Vision. This approach also obviated the need for implementing a 2D-viewing setup by myself in MALÅ Vision, which is already available in later versions of it.

Subsequently, starting over from scratch and developing a new class called ‘Borehole’ in Typescript greatly simplified the process of creating and positioning the boreholes using Three.js. However, generating information windows proved to be more difficult than expected, as they did not scale properly when zooming in and out, and positioning multiple windows was a challenge. As a workaround, the windows were modified to display one piece of information at a time, and the user could switch between information types.
Once this aspect was completed, the import of the two borehole types, COR and SND, worked flawlessly, with proper positioning and depth boundaries. Above each borehole, users could access information about the hole, as depicted in figure 5.1. This figure showcases the boreholes and comments contained in the imported SND-files.

In the implementation of the borehole visualization component in MALÅ Vision, the only part that required manual input was the velocity information for each layer. This was because as mentioned earlier communication between the Python component and other parts of MALÅ Vision could not be established. As a result, the velocity values were manually transferred from the Python code after being calculated within that environment. Similarly, the borehole-layers depth, the GPR-profile information, and the borehole position/trace number were manually added in Python after they were extracted in MALÅ Vision. However, in the future, all of this information should be provided directly through MALÅ Vision.

The interface with the user was developed using a Python figure window, which proved to be a relatively quick process. Taking only a couple of weeks to create a
basic boundary-follower that allows the user to click on a position in the 2D-view of
the profiles, record the position, translate it into a trace and sample depth, and create
a line. However, there were some challenges with recording the user’s movements
on the figure when the mouse button was held down and pausing the code until the
user let go of the mouse button. The solution was to create a so-called ‘brake point’,
a point that forces the code to stop at a location until the user clicks continue, in
the code at the position where the coordinate path over the figure was saved into a
variable. While this may not be the best solution, it allowed for all the important
parts of the script to be tried, and the code had to be run in ‘debug’ mode. The
basic boundary-follower script was initially provided by Guideline Geo as an Octave
script but was translated into Python for this project. The script works by finding
the largest value on the next trace position within an interval of ±3 samples and
repeating this process for each subsequent trace until reaching the right edge. While
the focus was not on improving the script, but rather on making it as useful as
possible for the user, several features were added. Features such as the ability to
update the boundary by clicking again, drawing lines by holding down the mouse
button, creating and removing multiple layers, and having the boundary-follower go
in both directions until hitting the edge, or until the older version of line has a value
on one of the samples within an interval. The majority of the time spent on this
part of the project was dedicated to finding and fixing bugs, as well as adding new
functions to make it easier for users to update the lines.

I began working on the GPR-profile adjustment script, leveraging a boundary-
follower that proved to be effective. While researching different methods to accom-
plish this, I quickly ruled out several approaches that would not be suitable for our
needs. After considering various options, I decided to explore the Seam carving
method [16], see figure 3.1, which appeared to be both feasible and valuable. How-
ever, I later realized it could be problematic for some layers, particularly those with
intricate details at certain levels. Seam carving works by finding a seam of pixels
with the lowest value and removing them, but this approach could disproportionately
alter a layer’s structure, giving a false picture of the subsurface.

Given this concern, I developed my own method that evenly adds or removes
values throughout the depth of the layer. Initially, I tried to count the number of
samples that differentiated between the borehole and the GPR-layer but encountered
issues when the number of values that needed to be removed exceeded the number
in the layer at another trace number. Furthermore, this approach did not necessarily yield more accurate results, as the discrepancy between GPR- and borehole-layers is primarily due to differences in material velocities. To improve accuracy, I implemented a function that calculated the percentage error in each layer, which was then multiplied by the number of samples to determine the number needed to be added or removed. This worked great and removed many of my problems. I encountered some issues initially when rounding caused the number needed to be added or removed to be off by one, resulting in the script crashing. To address this, I ensured that the number of samples in each trace remained consistent by rounding down the number that needed to be removed and adding the difference to the bottom of the trace. This does not compromise the quality of the results since the number of samples in each layer is typically small, ranging from 1 to around 4, and profiles usually contain a much larger number of samples then that.

As I continued working on the script, I had an idea to use the percentage error to calculate the velocity of the layer. This information could provide valuable insights into the composition of the layer, as the velocities of many materials are well-known, table A.1. Calculating the velocity is straightforward once the percentage error is known, as the standard velocity for GPR-profiles is 100 m/µs, which is close to the mean velocity in typical ground conditions.

In certain scenarios, layers can overlap, for instance, when the bedrock protrudes above the surface. Initially, this posed some challenges, but I devised a solution where the upper layer takes precedence over the lower one. This approach proved to be highly effective, allowing me to make necessary adjustments to profiles affected by this issue, and it has consistently yielded satisfactory results.

5.1. Future recommendation

5.1.1. The next step

The results from this project will with a little bit of more work have great usability in a wide range of applications. The first and most important step will be to move the user interface to be totally in MALÅ Vision instead of a Python figure window. Doing this will be a big step to make my results more user-friendly. To do this the flowchart seen on figure 4.6 will be exchanged for the much better flowchart figure 5.2, which shows my original plan for the workflow.
Figure 5.2: Flowchart that shows how my original plan for the workflow was, and how I recommend that it looks in the future.

In addition, it would be beneficial to include the ability for the user to click on a profile and display that position’s trace information, such as the depth of each layer-boundary and velocity. This would make the information from other parts of the adjusted GPR-profile accessible, giving new and valuable knowledge previously unknown.

The SND-files I used contained only one type of analysis per file, whereas in reality, most files contain multiple types of analysis within the same file. Therefore, it would be beneficial in the future to develop a filtering mechanism that can extract the specific analysis of interest by identifying the relevant types within the SND file.

As of now, the adjustment algorithm only works when using one borehole, and
this should be changed to be able to use multiple boreholes. A method to do this could be to adjust each trace according to the closest borehole and some kind of linear steps could be implemented so that the adjustments align with each other and no hiccups at the midpoint between them.

5.1.2. Future expansions
Now let’s discuss potential future expansions for this project. One possibility when dealing with multiple GPR-profiles is to interpolate between the picked layers and generate 3D-volumes for each layer. This would give the user the possibility to only show, for example, the ice on a river, or only the bedrock. Another similar approach is to isolate each layer from the GPR-profile and hide all other information. These techniques could also prove useful in certain applications, such as when the user wants to focus on one layer only.

Developing an AI capable of identifying boundaries would be highly advantageous. This would significantly reduce the time required for the user to manually pick each line, and the results would be more consistent regardless of the user’s skill level. An AI trained using an appropriate method would likely produce superior results. But also to improve the semi-automatic layer-picking presented in this thesis could be improved a lot since it will always be hard to make a 100% foolproof AI.
References


Appendix
Table A.1: GPR wave velocities in some common materials and soils.

<table>
<thead>
<tr>
<th>Material</th>
<th>Relative dielectric $[\varepsilon_r]$</th>
<th>Velocity $[m/\mu s]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Air</td>
<td>1</td>
<td>300</td>
</tr>
<tr>
<td>Water</td>
<td>81</td>
<td>33</td>
</tr>
<tr>
<td>Polar snow</td>
<td>1.4-3</td>
<td>194-252</td>
</tr>
<tr>
<td>Polar ice</td>
<td>3-3.15</td>
<td>168</td>
</tr>
<tr>
<td>Pure ice</td>
<td>3.2</td>
<td>167</td>
</tr>
<tr>
<td>Freshwater lake ice</td>
<td>4</td>
<td>150</td>
</tr>
<tr>
<td>Sea ice</td>
<td>2.5-8</td>
<td>78-157</td>
</tr>
<tr>
<td>Permafrost</td>
<td>1-8</td>
<td>106-300</td>
</tr>
<tr>
<td>Coastal sand (dry)</td>
<td>10</td>
<td>95</td>
</tr>
<tr>
<td>Sand (dry)</td>
<td>3-6</td>
<td>120-170</td>
</tr>
<tr>
<td>Sand (wet)</td>
<td>25-30</td>
<td>55-60</td>
</tr>
<tr>
<td>Silt (wet)</td>
<td>10</td>
<td>95</td>
</tr>
<tr>
<td>Clay (wet)</td>
<td>8-15</td>
<td>86-110</td>
</tr>
<tr>
<td>Clay soil (dry)</td>
<td>3</td>
<td>173</td>
</tr>
<tr>
<td>Marsh</td>
<td>12</td>
<td>86</td>
</tr>
<tr>
<td>Agricultural land</td>
<td>15</td>
<td>77</td>
</tr>
<tr>
<td>Pastoral land</td>
<td>13</td>
<td>83</td>
</tr>
<tr>
<td>Average soils</td>
<td>16</td>
<td>75</td>
</tr>
<tr>
<td>Granite</td>
<td>5-8</td>
<td>106-120</td>
</tr>
<tr>
<td>Limestone</td>
<td>7-9</td>
<td>100-113</td>
</tr>
<tr>
<td>Dolomite</td>
<td>6.8-8</td>
<td>106-115</td>
</tr>
<tr>
<td>Basalt (wet)</td>
<td>8</td>
<td>106</td>
</tr>
<tr>
<td>Shale (wet)</td>
<td>7</td>
<td>113</td>
</tr>
<tr>
<td>Sandstone (wet)</td>
<td>6</td>
<td>112</td>
</tr>
<tr>
<td>Coal</td>
<td>4-5</td>
<td>134-150</td>
</tr>
<tr>
<td>Quartz</td>
<td>4.3</td>
<td>145</td>
</tr>
<tr>
<td>Concrete</td>
<td>6-30</td>
<td>55-112</td>
</tr>
<tr>
<td>Asphalt</td>
<td>3-5</td>
<td>134-173</td>
</tr>
<tr>
<td>PVC, Epoxy, Polyesters</td>
<td>3</td>
<td>173</td>
</tr>
</tbody>
</table>