ERDC/CRREL TR-22-18

US Army Corps
of Engineers®
Engineer Research and
Development Center

ERDC
ENGINEER RESEARCH & DEVELOPMENT CENTER

Subterranean Detection and Monitoring

Automated Ground-Penetrating-Radar Post-Processing Software in R Programming

Samantha N. Cook, Marissa J. Torres, Nathan J. Lamie,
Lee J. Perren, Scott M. Slone, and Bonnie J. Jones

September 2022

Cold Regions Research
and Engineering Laboratory

The US Army Engineer Research and Development Center (ERDC) solves the nation's toughest engineering and environmental challenges. ERDC develops innovative solutions in civil and military engineering, geospatial sciences, water resources, and environmental sciences for the Army, the Department of Defense, civilian agencies, and our nation's public good. Find out more at www.erdc.usace.army.mil.

To search for other technical reports published by ERDC, visit the ERDC online library at https://erdclibrary.on.worldcat.org/discovery.

ERDC/CRREL TR-22-18
September 2022

Automated Ground-Penetrating-Radar Post-Processing Software in R Programming

Samantha N. Cook, Marissa J. Torres, Nathan J. Lamie, Lee J. Perren, Scott M. Slone, and Bonnie J. Jones

US Army Engineer Research and Development Center (ERDC)
Cold Regions Research and Engineering Laboratory (CRREL)
72 Lyme Road
Hanover, NH 03755-1290

Final Report

Prepared for Headquarters, US Army Corps of Engineers
Washington, DC 20314-1000

Under PE 0602146A, Project AT2, "Subterranean Detection and Monitoring," Task SAT202, "Subterranean Threat Assessment through Rapid Sensing (STARS)"

Abstract

Ground-penetrating radar (GPR) is a nondestructive geophysical technique used to create images of the subsurface. A major limitation of GPR is that a subject matter expert (SME) needs to post-process and interpret the data, limiting the technique's use. Post-processing is time-intensive and, for detailed processing, requires proprietary software. The goal of this study is to develop automated GPR post-processing software, compatible with Geophysical Survey Systems, Inc. (GSSI) data, in open-source R programming. This would eliminate the need for an SME to process GPR data, remove proprietary software dependencies, and render GPR more accessible. This study collected GPR profiles by using a GSSI SIR4000 control unit, a 100 MHz antenna, and a Trimble GPS. A standardized method for post-processing data was then established, which includes static data removal, time-zero correction, distance normalization, data filtering, and stacking. These steps were scripted and automated in R programming, excluding data filtering, which was used from an existing package, RGPR. The study compared profiles processed using GSSI software to profiles processed using the R script developed here to ensure comparable functionality and output. While an SME is currently still necessary for interpretations, this script eliminates the need for one to post-process GSSI GPR data.

Contents

Figures and Tables

Figures

Preface

This study was conducted for Headquarters, US Army Corps of Engineers, under PE 0602146A, Project AT2, "Subterranean Detection and Monitoring," Task SAT202, "Subterranean Threat Assessment through Rapid Sensing (STARS)." The technical monitor was William J. McCleave, US Army Engineer Research development Center, Geotechnical and Structures Laboratory (ERDC-GSL).

The work was performed by the Terrestrial and Cryospheric Sciences Branch (Dr. John Weatherly, chief), Engineering Resources Branch (Dr. Melisa Nallar, acting chief), and Signature Physics Branch (Dr. Steven Peckham, acting chief) of the Research and Engineering Division, US Army Engineer Research and Development Center, Cold Regions Research and Engineering Laboratory (ERDC-CRREL). At the time of publication, Dr. Caitlin A. Callaghan was the division chief and Ms. Pamela G. Kinnebrew was the technical director for Military Engineering. The acting deputy director of ERDC-CRREL was Mr. Bryan E. Baker, and the director was Dr. Joseph L. Corriveau.

COL Christian Patterson was commander of ERDC, and Dr. David W. Pittman was the director.

1 Introduction

1.1 Background

Ground-penetrating radar (GPR) is a nondestructive geophysical technique that uses reflected electromagnetic energy to create images of the subsurface. The system works by transmitting electromagnetic waves of a known frequency into the ground in a spherical shape. These waves reflect off subsurface structures with different electrical conduction properties or dielectric relative permittivities located behind, directly below, and in front of the antenna (due to the spherical shape of the signal). A receiver at the surface records two-way travel time (TWTT), or the time it takes for a wave to travel down to an interface and back up to a receiver at the surface, as well as the amplitude, or strength of the signal (Figures 1 and 2; Baker et al. 2007). With this information, changes in physical properties (geologic interfaces and other features), electromagnetic wave velocities, and depths to interfaces and features can be estimated.

Figure 1. *Left*, depiction of ground-penetrating radar (GPR; *orange rectangle*) being towed across the surface and transmitting electromagnetic energy into the subsurface in a spherical shape. *Right*, depiction of GPR imaging a boulder in the subsurface over time as the antenna is towed from position 1 (P1) through position 5 (P5). Note that the boulder is imaged while the antenna is located in front of, directly over, and beyond the boulder.

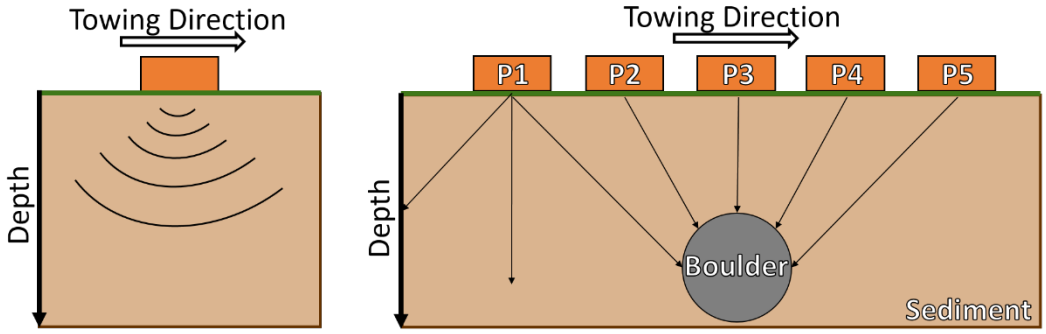


Figure 2. GPR data collection.

GPR is a widely utilized technique that has applications in geological investigations (e.g., Campbell et al. 2017), archaeology (e.g., Ciampoli et al. 2020; Ristić et al. 2020), construction and engineering (e.g., Morcous and Erdogmus 2009; Amran et al. 2017), environmental remediation (e.g., Steelman et al. 2017; Shao et al. 2018), military operations (Abeynayake and Tran 2016), and other industries. A major limitation of using GPR, however, is that a subject matter expert (SME) must post-process and interpret the data, which makes widespread use less likely. Post-processing, defined here as universal modifications made to GPR data after its collection, including static data removal, time-zero correction, distance normalization, data filtering, and stacking, is typically a time-intensive process and, for more detailed processing, requires costly proprietary software. These factors further limit widespread use of GPR.

1.2 Objectives

The goal of this study was to develop automated GPR post-processing software in the open-source R programming language (henceforth "R"). This would eliminate the need for an SME to post-process GPR data, remove dependencies on proprietary GPR data processing programs, and ultimately render GPR a more accessible tool to evaluate the subsurface. To achieve this goal, the objectives were (1) to define a standardized post-processing procedure for Geophysical Survey Systems Inc. (GSSI) GPR data, (2) to collect GPR data to support script development, and (3) to create an automated routine in R that could carry out all defined GPR post-processing steps.

1.3 Approach

To achieve the objectives, the team defined a standardized procedure for GPR file post-processing using GSSI software "RADAN" (GSSI 2017a). Subsequently, previously developed GPR-processing scripts written in R were reviewed to determine if existing scripts could carry out all the required post-processing steps. To meet script-development needs, multiple rounds of fieldwork were conducted to acquire specific GPR profiles, such as profiles with varying lengths of static data, collected with high-accuracy GPS, and with differing stacking values. These files were used in the development of a stand-alone script, written in R, to perform all defined GPR post-processing steps.

Section 2 of this report addresses the defined post-processing steps needed to generate interpretable GPR data. Section 3 highlights the portions of a previously developed script written in R called RGPR (Huber and Hans 2018), which this study used in part. Section 4 provides the methodology and equipment used for multiple data collections in the field, while section 5 details the automated GPR post-processing script developed in R. Section 6 outlines conclusions from script development and provides insights into next steps for future development. The Appendix provides the GPR post-processing script developed (with operating instructions).

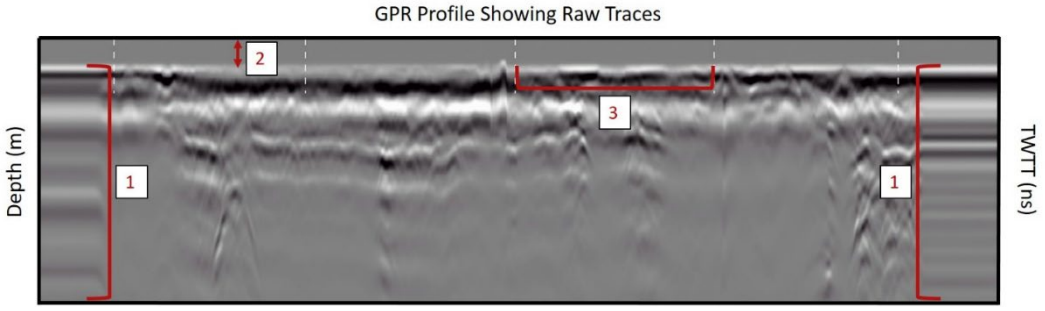
2 GPR Post-Processing

The full manner in which GPR data is post-processed largely depends on the application and the properties of targets of interest. However, several post-processing steps are universal, or required, for all GPR profile interpretations regardless of end-user application. To assist in defining a standardized post-processing methodology, GSSI's proprietary RADAN GPR software was utilized, as only GSSI GPR equipment was used in this project. Profiles processed using RADAN acted as a baseline to compare with profiles processed using this study's R script to ensure comparable functionality and end results. The standardized GPR post-processing procedure defined for this study was performed for the same surveys that were processed in RADAN. This procedure includes the following steps: (1) static data removal, (2) time-zero correction, (3) distance normalization, (4) data filtering, and (5) stacking. These post-processing steps remove insignificant portions of the data (step 1), correct the *y*- (step 2) and *x*-axes (step 3), remove unwanted noise (step 4), and improve signal-to-noise ratios (step 5). The following subsections provide a more detailed description of the post-processing steps and why each is necessary.

2.1 Static Data removal

Static data refers to any data collected while a GPR antenna is stationary. Static data usually occur at the beginning and end of GPR profiles because of delays between when data collection is initiated and the operator begins moving or, conversely, delays between when the operator stops moving and data collection is terminated (Figure 3). Static data can also occur anywhere within a profile if the operator stops moving for a given length of time. Removing the resultant static and insignificant data ensures an accurate calculation of the horizontal distance traveled and combines features surveyed together into a continuous subsurface profile.

Figure 3. A GPR profile collected with a 200 MHz* antenna. The *left* and *right y*-axes depict depth below the surface in meters and TWTT (two-way travel time) in nanoseconds, respectively. The *x*-axis shows the profile with all the raw traces, which includes static data at the beginning and end of the profile (when the antenna was stationary) as well as the time the antenna was towed along the ground surface (between static data portions). The *red-bracketed* sections labeled "1" highlight areas of static data at the beginning and end of the profile. The *red arrow* labeled "2" shows an offset between the ground surface (0 m depth) and the start of the data. The *red-bracketed* section labeled "3" shows an area between two *white-dashed line* user marks. These user marks, or high-resolution GPS data, are used to distance normalize the data and correct the *x*-axis.



2.2 Time-zero correction

Time zero refers to the arrival of the first transmitted pulse to the receiver, which denotes the location of the ground surface. Time zero is typically set arbitrarily during data collection and therefore is not aligned with the actual location of the surface. This results in inaccurate TWTT, depths, and depth-to-feature estimates. The depth estimates in GPR profiles used in this study are made using a single velocity, which results in a level of inaccuracy as reflections are produced only when velocity changes occur. However, these velocity changes are minor and result in small (centimeter-scale) as opposed to large (meter-scale) inaccuracies. Note that multiple offset surveys and variable velocity-migration techniques can provide more-accurate velocity and associated depth estimates, but these techniques are beyond the scope of this development.

Time-zero correction, however, improves depth estimates. When collecting GPR data, the direct wave, or the wave that results from energy traveling between the transmitter and receiver in the air, arrives at the receiver prior to energy propagation into the ground and, therefore, can be an indication of the actual location of the surface (Leach 2019). In a GPR profile,

* For a full list of the spelled-out forms of the units of measure used in this document and their conversions, please refer to *US Government Publishing Office Style Manual*, 31st ed. (Washington, DC: US Government Publishing Office, 2016), 245–252, https://www.govinfo.gov/content/pkg/GPO-STYLEMANUAL-2016/pdf/GPO-STYLEMANUAL-2016.pdf.

the direct wave appears as a flat wavelet at the top of the profile, above which there is a lack of amplitude response caused by the receiver listening for an energy return prior to the transmitter sending a signal. During time-zero correction, the direct wave is automatically removed to accurately depict the ground surface location (see Figure 3; Leach 2019).

Time-zero correction refers to a vertical adjustment of the data to calibrate zero depth with time zero (T_0) on the vertical axis. Time-zero corrections are made by designating the first arrival of a wave at T_0 "zero", or level with the ground surface. Although there are a variety of ways to correct time zero, this study used the "first positive peak" method, in which the first positive peak of the direct wave is positioned at the 0 m depth location to set the position of the ground surface (Yelf 2004; Leach 2019).

2.3 Distance normalization

Distance normalization is a correction of the distance traveled during data collection (x-axis) and is necessary to accurately locate the position of targets of interest along a GPR profile. In practice, GPR operators generate user marks during data collection, which are manual entries in the data that represent a known spatial distance and are generally essential for distance normalization. For example, an operator may enter a user mark in the data every 5 m of distance traveled (see Figure 3). Operators typically collect handheld GPS data in tandem with GPR data to obtain latitude and longitude positions to use for post-processing. These handheld GPS units often have low-accuracies but are used routinely to guide user-mark entry. In post-processing, however, there are typically an unequal number of traces found between user marks because of inconsistencies in data collection towing speed. Distance normalization is a method to interpolate the location of individual traces in the profile such that there are an equal number of traces per unit distance (see Figure 3). This method is commonly known as "rubber-sheeting." The RADAN software distance normalizes data via rubber-sheeting, where the unit distance is defined by manual user marks that correspond to known locations along the length of the profile.

Alternatively, collecting high-accuracy GPS data can eliminate the need to manually enter user marks. For this study, a Trimble GPS with a sampling rate of 1 Hz and a resolution of ±10 cm was used. With GPS data of this

kind, equitemporal (1 Hz) GPS locations replace user marks, and an SME selects the desired unit distance over which the traces in the survey are distance normalized (such as 1 m, 5 m, 10 m, etc.; see section 5.3 for more details). In this study, distance normalization required an attached high-accuracy GPS unit (i.e., manual user marks could not be used due to their low accuracy).

2.4 Data filtering

While there are a variety of data filtering post-processing methods that can reduce noise and enhance the visibility of features of interest within GPR profiles, the creation of unwanted artifacts or the accidental removal of meaningful data can occur if a profile is over filtered. Therefore, this study used a simple band-pass filter, which is a post-processing technique that reduces noise in GPR data by removing all frequencies that fall outside of a specified range (or "band"). This filter alters the profile's amplitude with respect to frequency.

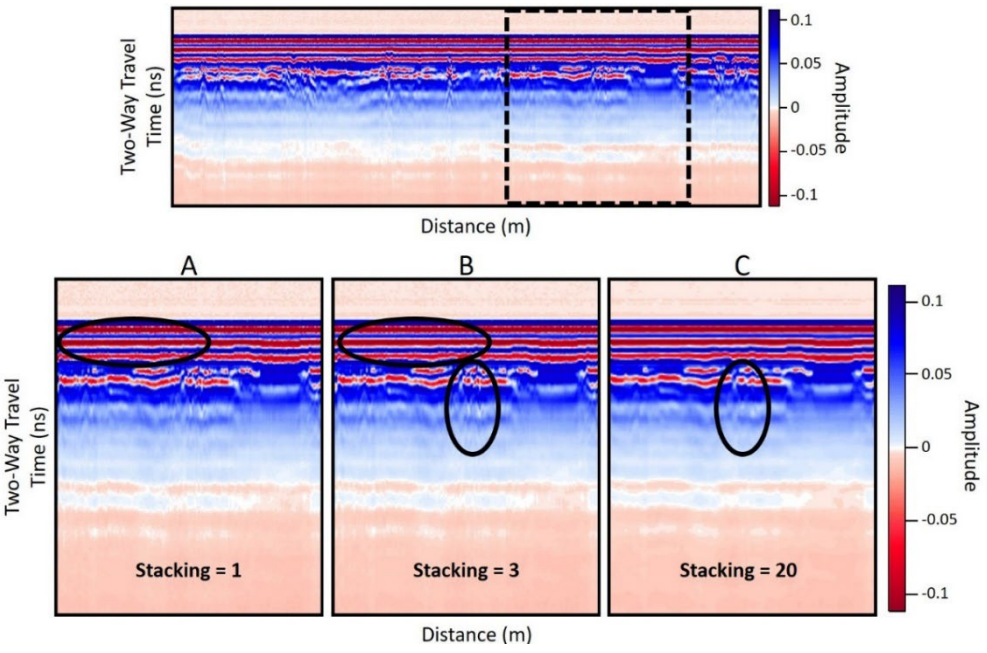
GPR antennas, such as a 100 or 200 MHz antenna, are labeled based on the center frequency collected, but the antenna receives and records data over a larger frequency range. For example, a 200 MHz antenna has a center frequency of 200 MHz but can receive a range of lower and higher frequencies, such as 100 to 300 MHz. Frequencies furthest from the center frequency introduce noise in the data. Removing this noise provides clearer data and improved interpretation. To remove unwanted frequencies, a band-pass filter is used with thresholds determined through a default formula of plus or minus half the center frequency of the antenna, though end users may adjust this range if desired. This threshold frequency formula was selected in this study as it is recommended in the GSSI operation's manual (GSSI 2017b). For example, the frequency thresholds for a 200 MHz antenna would be 100 MHz and 300 MHz, eliminating all other frequencies outside of this band.

2.5 Stacking

Stacking is a horizontal data filter that aids in data visualization as it improves the signal-to-noise ratio and reduces random noise. Stacking applies a lateral moving average to the data to average adjacent traces (GSSI 2017a). Combined adjacent traces are output as a single trace (e.g., traces 3, 4, and 5 are averaged to produce a new trace 4 value; traces 4, 5, and 6

are averaged to produce a new trace 5 value, and so on), but the total number of traces in the profile is preserved. Note, this is different from stacking applications in other programs, such as RADAN, where the total number of traces in the profile is reduced by a factor equal to the stacking value. In this study, stacking is applied as an infinite impulse response (IIR) filter operating in the horizontal direction or as a running average in the horizontal direction within a defined window size. Note, the selected window size of traces to combine enables the filter to output the same number of traces that are input. However, stacking too many traces creates a window that is too large and will potentially filter out high-frequency (i.e., small) targets in the dataset. The IIR horizontal filter process has a $1/n$ influence on the number of traces, where n is the number of traces that are stacked (Figure 4). An n value that is too large will adversely influence the data by oversmoothing it. The negative impact of a running average filter is a reduction in signal amplitude of potentially meaningful data, along with the incoherent noise. However, the horizontal running average does not alter the phase of the signal.

Figure 4. *Top*, GPR profile collected with a 100 MHz antenna at the Cold Regions Research and Engineering Laboratory (CRREL) in Hanover, New Hampshire, in October 2020. The area outlined by the *dashed line* is shown in the bottom panels. *Bottom*, panels *A*, *B*, and *C* show the same area of the profile above with different stacking values applied. *A* was stacked once, *B* was stacked three times, and *C* was stacked 20 times. The horizontal ovals in *A* and *B* show horizontal banding (noise) that was reduced as a result of the stacking. The vertical ovals in *B* and *C* show hyperbolas that were present in *B* but inadvertently removed in *C* as a result of overstacking and oversmoothing of the data.



The number of traces selected to stack is typically based on the observed noise of the profile. In this study, the number of stacked traces was set to a default value of three, though the end user can select a different value if required for their processing needs. Section 5.5 discusses the reasoning behind this default value.

3 Existing Post-Processing

Prior to automating the defined GPR post-processing steps, a literature review was conducted as part of this project to determine if other GPR-processing software in R existed and, if so, whether any portion of it could be leveraged for this study. Ultimately, R was selected because personnel had expertise in this language and because of the existence of one publicly accessible package called RGPR. RGPR is an open-source GPR visualization and processing software package written in R and located on GitHub (Huber and Hans 2018). RGPR allows users to load, plot, visualize, and process GPR data. RGPR script development was based around Sensors & Software Inc. GPR data, though it has since been updated to incorporate capabilities for other GPR file types, including GSSI's proprietary DZT file type. RGPR's data loading, plotting, comparison (subtracting one GPR profile from another), and filtering capabilities were used in this study. Four of the study's five defined post-processing steps were developed independently from RGPR, including static data removal, time-zero correction, distance normalization, and stacking. Two of these five steps, time-zero correction and data filtering, exist in RGPR. While the RGPR data filtering was used in this study, the time-zero correction was not as it did not fit within the overall developed workflow for this project. The other three steps, static data removal, distance normalization, and stacking, are not available in RGPR.

4 GPR Data Collection

Fieldwork was conducted at the Cold Regions Research and Engineering Laboratory (CRREL) in Hanover, New Hampshire, to acquire GPR datasets that would support the development of the automated post-processing software (Figure 5). This area was selected because of its ease of access as well as the presence of known near-surface infrastructure (Figure 6). Knowing the location of the near-surface infrastructure ensured the post-processing script did not alter GPR-imaging of the subsurface features (i.e., the expression of features did not change significantly within the profiles when processed through the script). GSSI GPR equipment was used for this fieldwork, including a SIR4000 control unit and 100 MHz, 200 MHz, and 400 MHz antennas. Additionally, simultaneous GPS data were collected using a Trimble SPS852 and a GA810 Global Navigation Satellite System (GNSS) rover antenna (Figure 7). In total, 201 unique profiles were collected in 2020 for this effort.

Figure 5. Location of the CRREL field site. The *yellow circle* in the *inset* denotes the location of the laboratory. The *red rectangle* highlights the general area of data collection.

Figure 6. *Left*, subsurface infrastructure map of the parking lot data collection area shown in Fig. 5. The black grid represents several lanes of data collection. The enlarged image in the middle depicts subsurface infrastructure that crosses the data collection lanes perpendicularly. The image on the *right* shows GPR data collected at this location, in which a *red arrow* indicates the infrastructure crossed (a concrete culvert).

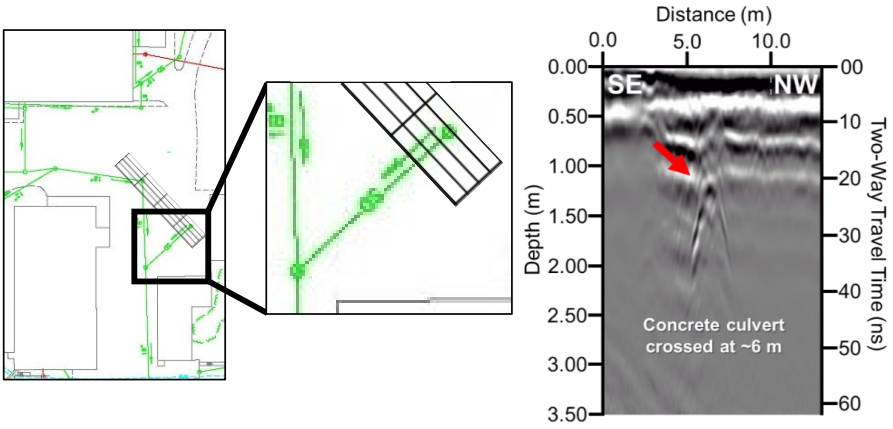
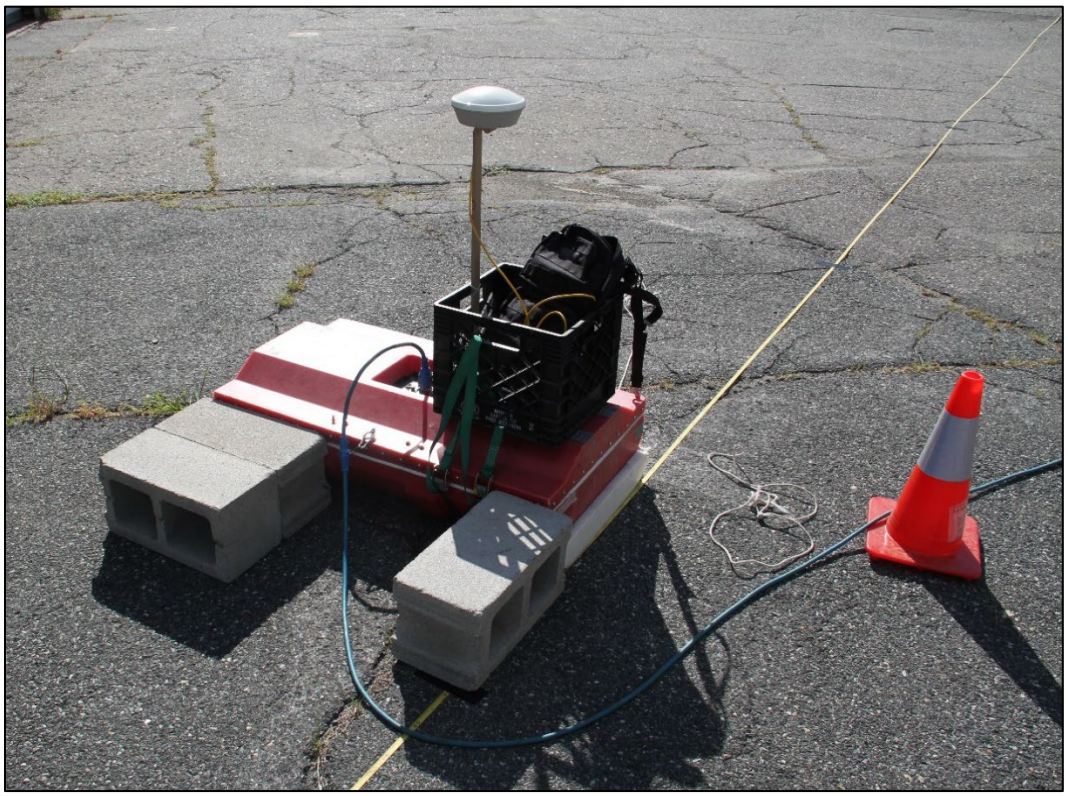


Figure 7. A typical GPR and GPS setup during field collections. This data collection occurred in August 2020 at CRREL. In front of the two cinder blocks, the red-orange rectangular 100 MHz antenna has a Trimble SPS852 and GA810 Global Navigation Satellite System (GNSS) rover antenna secured in a backpack placed within a milk crate. The crate is held in place using straps that attach to the antenna. The blue cable attached to the antenna connects the SIR4000 control unit (not pictured).

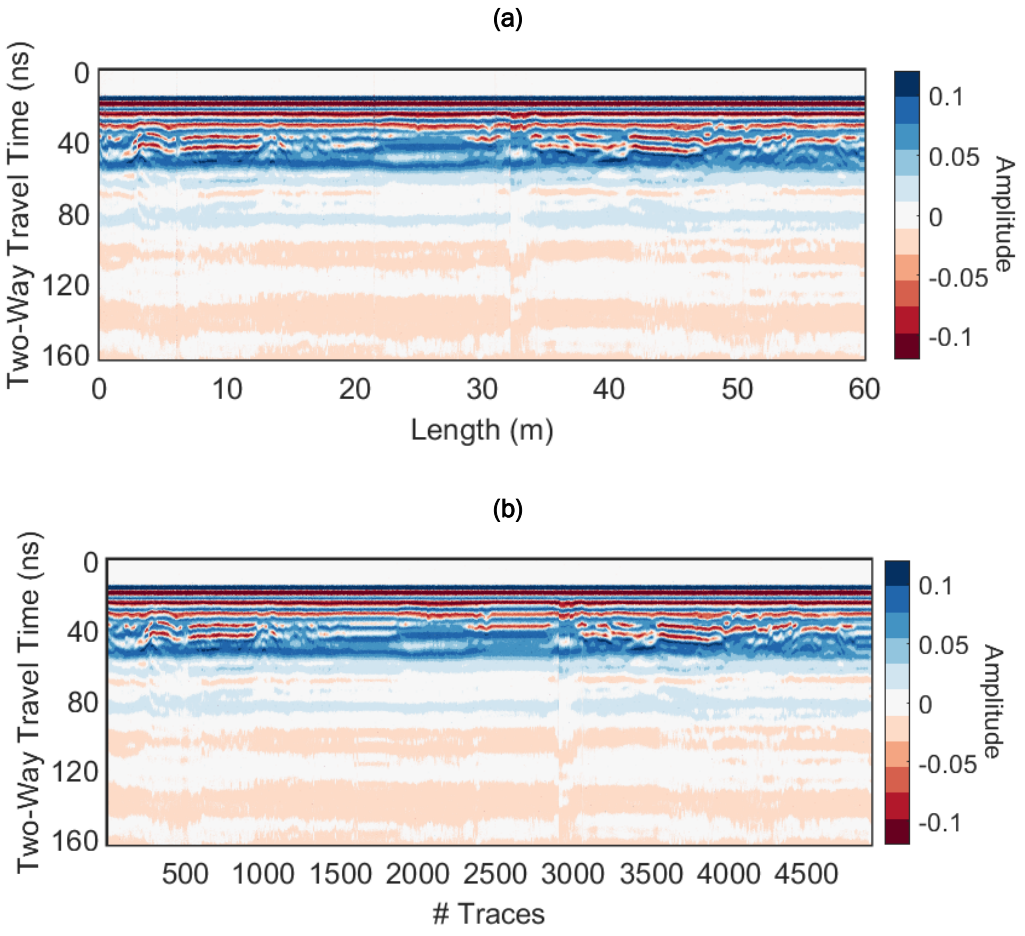


To develop a static data-removal script, profiles were collected that contained intentional static data at the beginning and end of profiles, as well as in the middle of the profile. For time-zero correction, data were collected that intentionally had time zero set at a poor location (and therefore needed to be corrected). To support distance-normalization development, profiles of various lengths and towing speeds were collected; and for data filtering, multifrequency data using all three antennas were collected. To develop a stacking script, both unstacked data and data that were stacked 4 and 16 times were collected to compare with data that were stacked 4 and 16 times during post-processing (as opposed to stacking applied concurrent with data collection).

5 GPR Post-Processing in R

In this section, the methods used in R to develop the five standardized GPR post-processing steps are presented and described. The data from a sample GPR profile (Figure 8) has been processed through each of the five steps, and the respective changes to the profile are presented throughout this section. This profile was collected on 20 October 2020 at CRREL using the equipment setup shown in Figure 7. The total distance traveled during data collection was 60 m, and the profile is composed of 4,915 traces.

Figure 8. Example of a raw GPR profile collected on 20 October 2020 at CRREL by using a 100 MHz GSSI antenna. The *y*-axis depicts the TWTT in nanoseconds. In *a*, the *x*-axis represents the total distance traveled according to GPS data; in *b*, the *x*-axis represents the total number of traces in the raw (unprocessed) profile, where static data are more pronounced at the beginning, middle, and end of the profile (when the antenna was stationary). The amplitude, or strength of the signal, is depicted by the *red*, *white*, and *blue* color ramp.



The GPS and GPR units used in this study had different sampling rates. The GPS unit recorded location data at a rate of 1 Hz (e.g., 1 sample per second), and the GPR antenna emitted electromagnetic pulses at a frequency of 24 Hz (e.g., 24 traces or scans per second). With RGPR, GPS coordinates are interpolated to every trace in the GPR profile upon reading the profile into R. However, for the purposes of this post-processing script, GPR traces and their corresponding GPS coordinates were sampled at the lower sampling rate (1 Hz). At this sampling rate, the velocity (meters per second), or towing speed, of the GPR antenna was more readily computed.

5.1 Static data removal in R

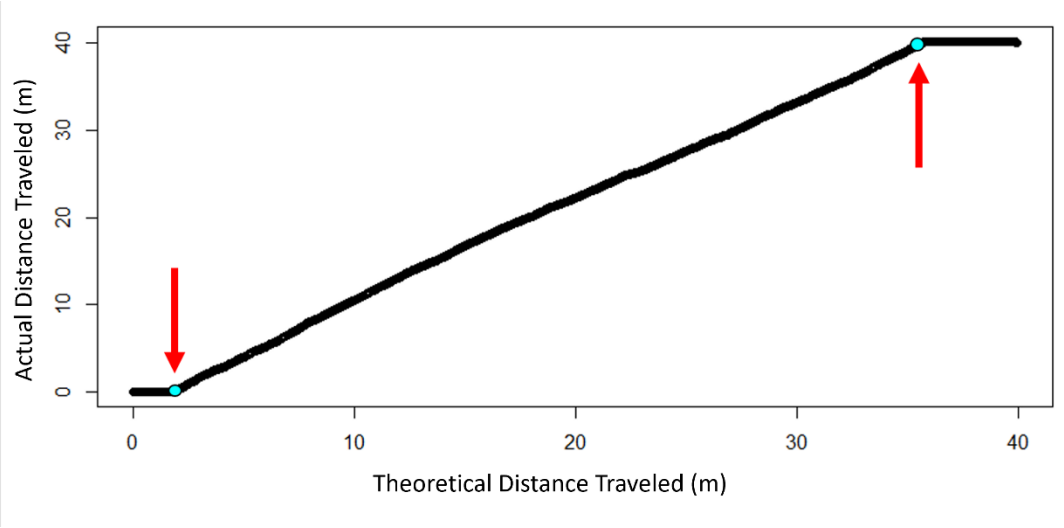
Static data removal is the first step in the overall post-processing workflow. The removal of static data relies on the high-accuracy, high-resolution GPS coordinates (longitude and latitude in decimal degrees) that correspond to each trace in the GPR profile rather than on the GPR traces themselves (which is how static data are removed in GSSI's RADAN). The difference between standing still and acceleration (starting) or deceleration (ending), which correspond to static versus kinetic data collection, can be distinguished from the GPS coordinates.

The difference between static data and kinetic data can be determined by investigating the relationship between the theoretical distance traveled and the actual distance traveled. The theoretical distance traveled considers the total linear distance traveled during data collection (the Haversine distance [distance on a sphere] between the first GPS coordinate and the last GPS coordinate) and assumes equal spacing (Δx) between traces. The actual distance traveled considers the recorded GPS coordinates of each trace along the profile (Δy), accounting for the physical movement of the GPR unit.

When the GPS and GPR units are active and stationary (i.e., static), the GPS records repeat coordinates and the GPR antenna records repeat traces of a single location. Similarly, the distance to each trace from the first position at that location remains at or near zero since multiple GPS coordinates of the same or similar location (within ±10 cm) are recorded. At this same location, the theoretical distance to each trace from the first position increases as though the units were moving. When the GPS and GPR units are in motion, both the actual and theoretical distances to each

trace from the first position are increasing. Comparing the theoretical distance to actual distance resembles a line of near-zero slope for static data and a linear slope for kinetic data. Figure 9 illustrates this relationship. The change in slope (i.e., the derivative) of the theoretical distance versus the actual distance between each trace reveals whether the GPR antenna is in motion versus stationary.

Figure 9. Relationship between the total theoretical distance traveled, considering start and end points only (*x*-axis), and the actual distance traveled, considering the distance between each trace in the profile (*y*-axis). The linear relationships between these distances indicate kinetic data collection. Flat sections of zero slope indicate static data collection. The *cyan circles* and *red arrows* indicate start and stop locations of kinetic data.

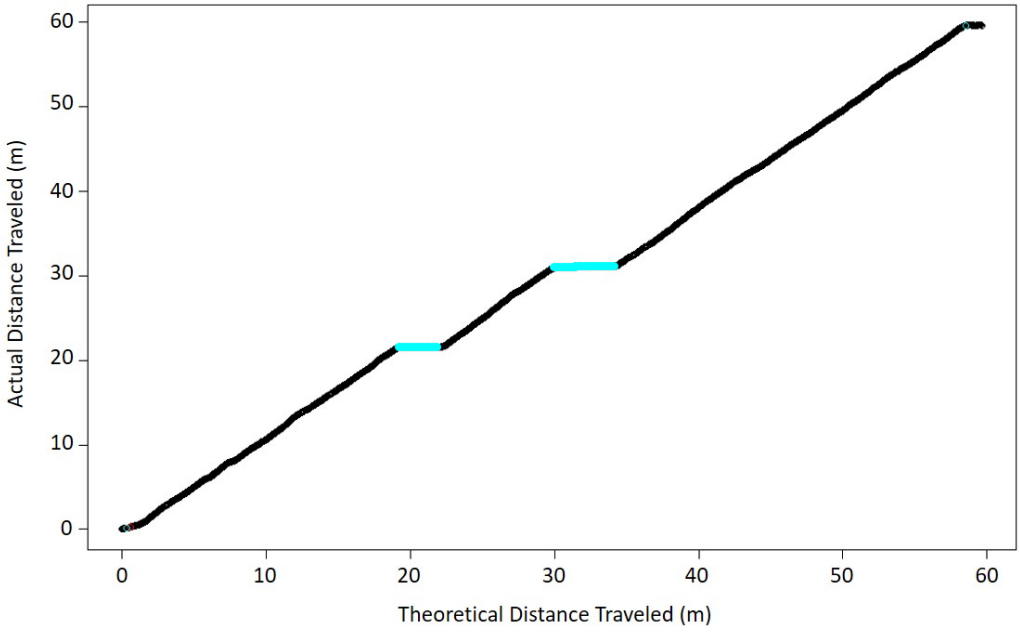


Static data was identified in the profile by evaluating the ratio of difference in actual distance between each trace (Δy) and difference in theoretical distance between each trace (Δx). This ratio, $\frac{\Delta y}{\Delta x}$, is the slope of the line in Figure 9. Where the slope value was at or below a certain threshold, the corresponding trace number was identified as static data. The threshold was defined relative to the average slope of the distance traveled (Equation 1), specifically as one standard deviation (`std`) less than the mean slope. The `which()` command in R returned the indices where this logical expression was met. The corresponding data (amplitude response, GPS coordinates, etc.) associated with static traces were then removed from the profile.

$$\text{which}\left(\frac{\Delta y}{\Delta x} \leq \text{mean}\left(\frac{\Delta y}{\Delta x}\right) - \text{std}\left(\frac{\Delta y}{\Delta x}\right)\right) \quad (1)$$

The relative threshold value was successful at identifying static data in the middle of a profile as well. If, for any reason, a GPR operator stops towing the GPR unit in the middle of the profile, this method will identify and remove that static data. Figure 10 provides an example of pauses during data collection that were identified by the static data removal process (cyan points).

Figure 10. Relationship between the total theoretical distance traveled, considering start and end points only (*x*-axis), and the actual distance traveled, considering the distance between each trace in the profile (*y*-axis). Linear relationships between these distances indicate kinetic data collection. Flat sections of zero slope indicate static data collection. Sections of static data in the middle of the profile are highlighted in *cyan*.

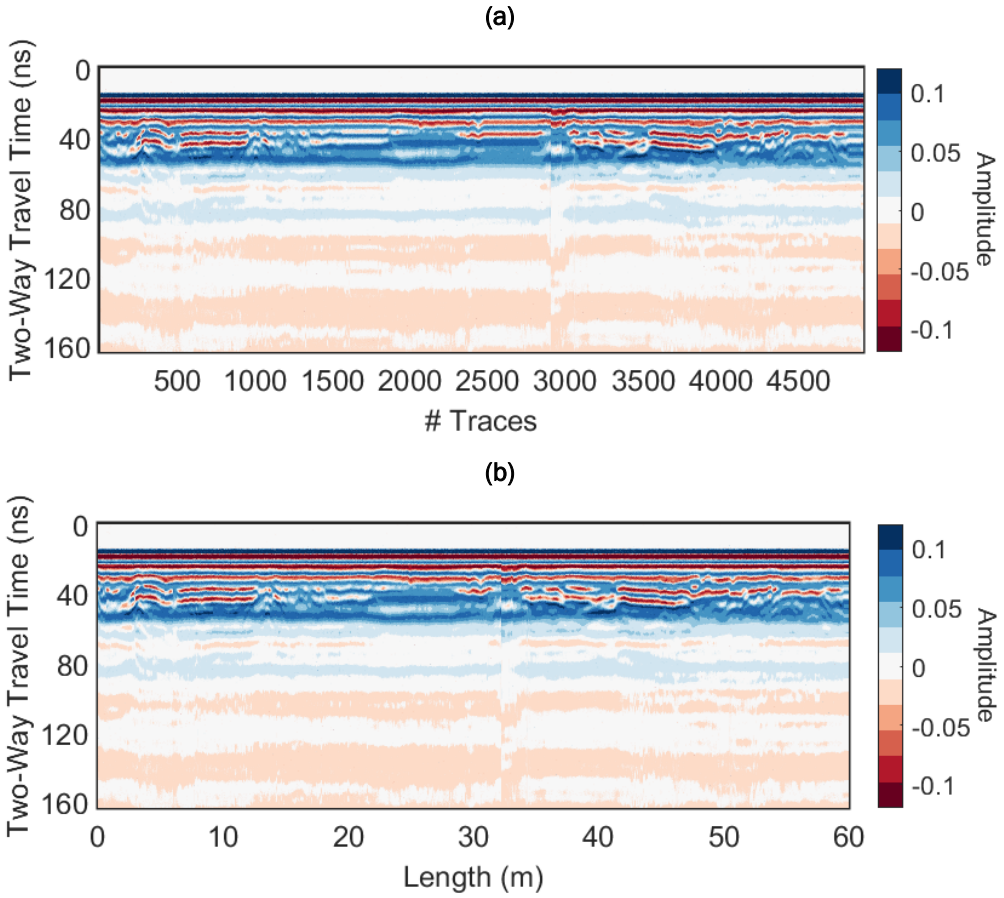


This method of identifying static data is most successful for profiles collected at moderate walking or towing speeds of the GPS and GPR units. Profiles collected at a slow or very slow pace lower the relative threshold value such that more traces are identified as static data throughout the length of the profile. Generally, this effect is not too much of a concern for post-processing of GPR data since thousands of traces are recorded and only a small portion of the data (10%–20%) would be removed at this step.

After the detection and removal of the traces associated with static data collection, the total number of traces and their corresponding GPS coordinates in the profile were reset to reflect the shorter profile. Traces were renumbered in sequential ascending order starting from one to the total

number of traces in the edited profile. The actual distance traveled was updated with the remaining traces and written to the GPR profile data. The edited profile (Figure 11) was then passed to the next post-processing step in the workflow.

Figure 11. Example of the static data removal post-processing step on (*a*) the sample GPR profile detailed at the beginning of this section (duplicate of Fig. 8*b*); and (*b*) the GPR profile after static data removal at the beginning, middle, and end of profile.

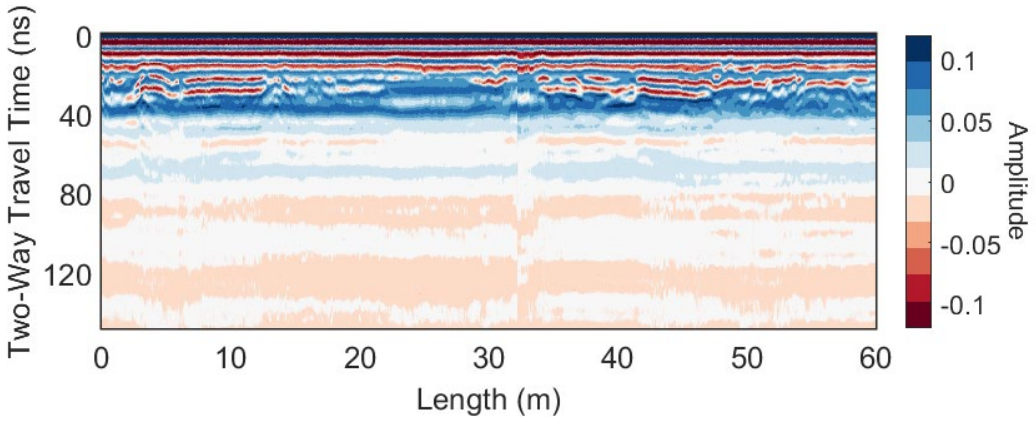


5.2 Time-zero correction in R

Time-zero correction is the second step in the overall post-processing workflow. As mentioned previously, time-zero correction is required to calibrate the depth and TWTT *y*-axes for improved depth-to-feature estimates. Essentially, this step removes the white space along the top of the profile. The "first positive peak method" identifies the location of the maximum positive amplitude value along each trace. Recall that the amplitude values in the GPR profile (red–blue color ramp) represents the energy that

travels between the transmitter and receiver in the air and subsurface. As that energy travels through the air before reaching the subsurface, the first positive value returned to the receiver provides an estimate for the location of the surface. The TWTT associated with the median location of first maximum amplitude along the profile is used as the value by which to shift the data vertically. For example, if the median maximum amplitude TWTT is located at 16.33 ns, then all traces are shifted vertically (up) by –16.33 ns. To shift the traces, all rows in the data up to the median TWTT value are removed. Once shifted, the *y*-axis values are reordered so that 0 m depth and 0 ns TWTT are aligned with the ground surface (Figure 12).

Figure 12. GPR profile shown in Fig. 11 with time-zero correction applied (compare to Fig. 13*b*).



5.3 Distance normalization in R

Distance normalization is the third post-processing step in the overall workflow. As previously mentioned, distance normalization involves correcting, or "normalizing," the horizontal scale of a GPR profile such that there is an equal number of traces per unit distance. In traditional methods, GSSI's RADAN software uses a rubber-sheeting method to accomplish this task. The concept of rubber-sheeting is complex and is generally associated with correcting spatial map projections and other georeferencing adjustments, commonly used in geographic information system (GIS) products (Esri 2016).

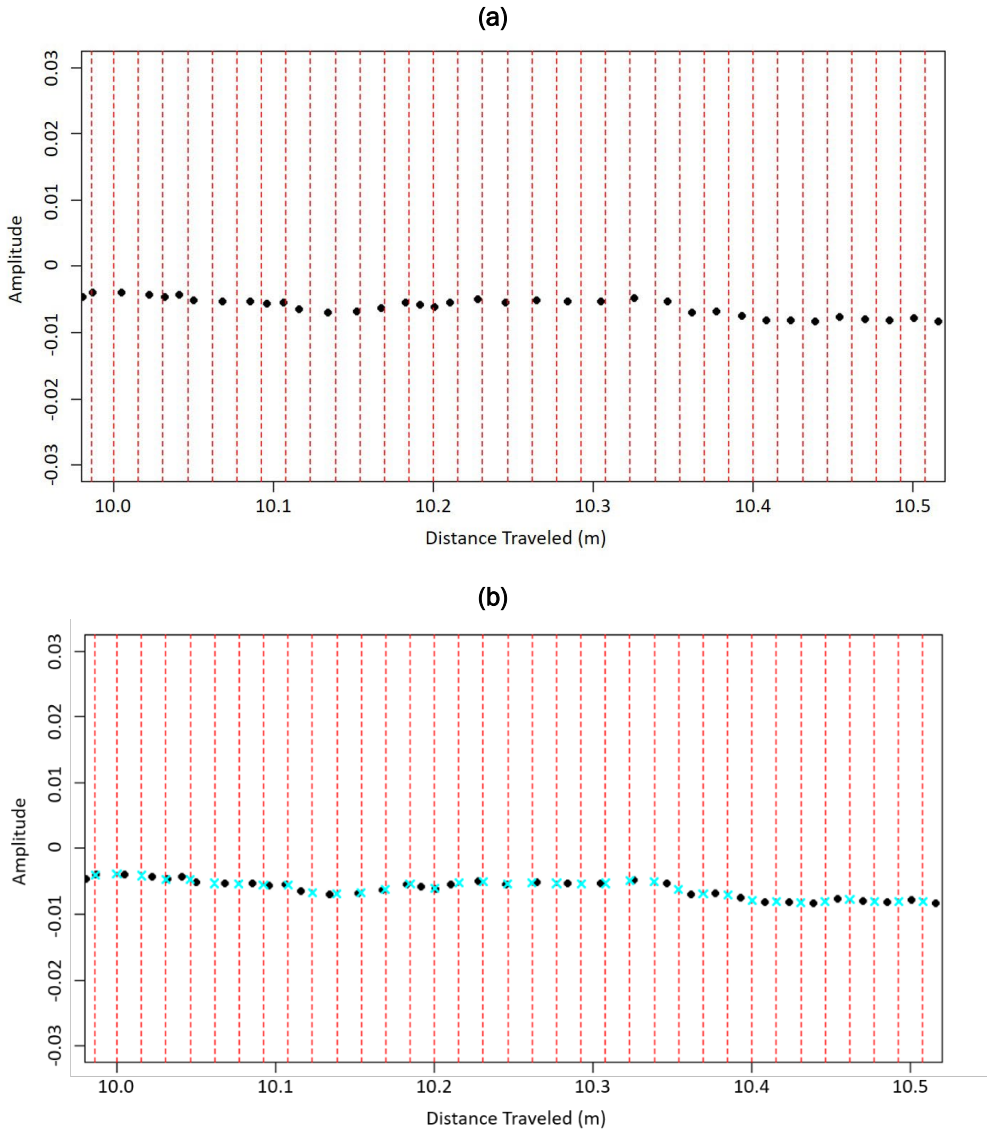
For GPR applications, there is limited guidance and knowledge about how the individual traces and their corresponding GPS coordinates and amplitude returns are treated or manipulated during a rubber-sheeting transformation, though Al-Nuaimy (1999) developed an algorithm to apply

rubber-sheeting to GPR processing. The workflow was designed to use manual user marks made during profile collection, providing an approximate unit distance traveled at each mark. Within each approximate unit distance, traces in between were either locally compressed (averaged) or stretched (interpolated) to meet the target number of traces per unit distance. However, using this rigid method, the total number of traces in the distance-normalized profile changed due to local compressing and stretching to meet the fixed total number of traces per unit distance.

The methods proposed in this study use the high-accuracy GPS coordinates associated with the traces rather than user marks, providing a more accurate estimate of unit distance traveled. Rather than normalizing the number of traces per unit distance, the method normalizes the physical distance between each trace along the length of the profile. As such, GPS coordinates were sampled at 24 Hz as opposed to 1 Hz in this processing step. The higher sampling rate was used to determine the exact trace numbers associated with a unit distance along a GPR profile.

The difference between actual trace positions (Δy) following static data removal is not expected to be equal due to deviations in towing speed during the profile collection process and error in the GPS location (±10 cm). The purpose of distance normalization is essentially to translate the trace positions and their corresponding amplitude returns along the profile to be equidistant within a specified unit distance or interval ($\Delta x|_a^b$). Figure 13 depicts this method. Conceptually, the distance-normalization process interpolates the amplitude returns from the actual trace positions within a unit distance of 2 m (see the black circles, y, for interval a = 10 m and b = 12 m in Figure 13) to the theoretical trace positions within that unit distance (see the red dashed lines, x, in Figure 13). In other words, the location of each individual trace is adjusted slightly to a theoretical location that is equidistant to its neighboring traces within a specified unit distance. The amplitude returns are interpolated to these theoretical locations (see the cyan crosses in Figure 13b).

Figure 13. Visual representation of distance normalization of a GPR profile using a piecewise cubic Hermite interpolating polynomial (PCHIP) linear interpolation method. Individual amplitude returns from the original or actual trace locations (y) are shown as *black circles* (*a*). Equidistant theoretical trace positions (x) denoted as *vertical red dashed lines*. Interpolated amplitude returns at the artificial locations are denoted as *cyan crosses* (*b*). A unit distance of 2 m was applied to this profile for the distance-normalization step.



In this study, the interpolation of the amplitude returns to these theoretical trace positions was completed using a piecewise cubic Hermite interpolating polynomial (PCHIP) linear interpolation method (Fritsch and Carlson 1980) in R. The PCHIP method is a shape-preserving piecewise

cubic interpolation approach such that the features (e.g., amplitude returns) in the GPR profile are maintained across each subinterval. In this case, the subinterval was a specified unit distance. In Figure 13*b*, the cyan crosses follow the general spline of the original amplitude returns (black circles) across this section of the GPR profile without losing information or changing the values significantly. Additionally, this method preserves the total number of traces from the original profile.

The unit distance (in meters) is the interval over which the GPR profile is normalized. The traces between each unit distance are preserved, meaning the amplitude returns within this interval will not be affected by amplitude returns outside of this interval. The default unit distance applied in this post-processing step is 2 m. Preliminary sensitivity tests show that smaller unit distances minimize shifting of the trace positions for the most accurate interpolated amplitude returns. If the total distance traveled during the profile is not an integer multiple of the unit distance, the last interval will contain the number of traces between the distance from the end of the previous interval to the total distance traveled at the end of the profile.

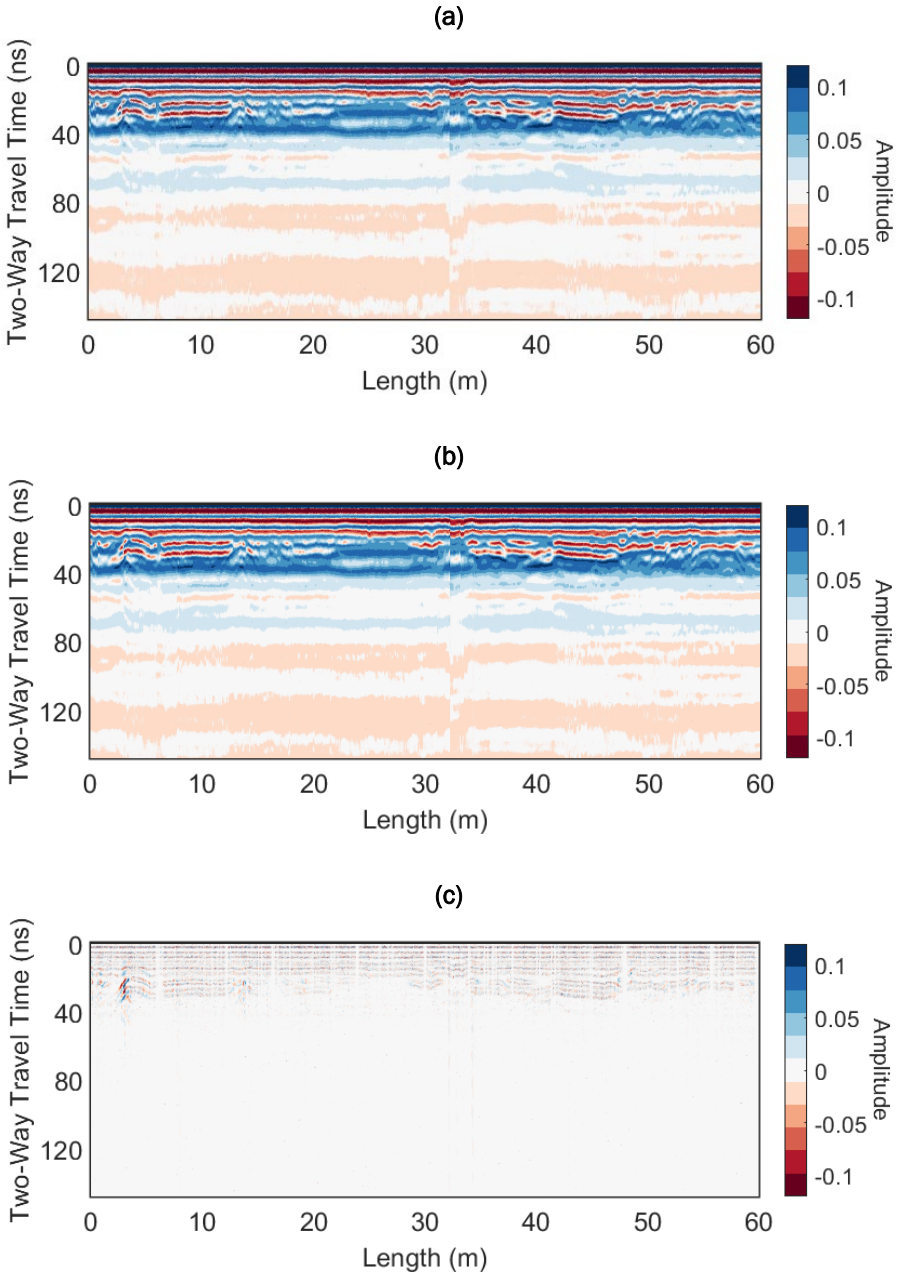
The trace numbers associated with a specified unit distance interval are identified from the list of actual distance traveled following static data removal. For a unit distance of 2 m, the trace numbers associated with a distance of 2 m, 4 m, 6 m, and so on from the start of the profile are identified and used as anchors (end points) of each interval.

The amplitude returns in the GPR profile are available in a matrix format when read through RGPR. The matrix is of size $[n \times m]$, where n is the number of vertical layers received by the GPR antenna (rows; TWTT) and m is the number of traces in the profile (columns; after static data are removed and time-zero correction is applied). The PCHIP interpolation method was executed over the horizontal axis (distance traveled) for each individual vertical layer in the GPR profile. In other words, for each row n, the PCHIP method is applied over all traces in m. As a result, a new matrix of the same size is created that contains the distance-normalized profile.

Figure 14 displays the effect of the distance-normalization script on the sample GPR profile. The top plot depicts the profile with static data removed and time-zero correction applied (but prior to distance normalization); the middle plot shows the distance-normalized profile; the bottom

plot displays the difference between the top and middle plots. A unit distance of 2 m was applied in this formulation. The difference plot highlights the location and magnitude of changes in the amplitude returns caused by the distance-normalization process.

Figure 14. Example of the distance-normalization post-processing step on the sample GPR profile detailed at the beginning of this section (see Fig. 8): (*a*) The GPR profile after static data removal and time-zero correction (duplicate of Fig. 12), (*b*) the GPR profile after distance normalization has been applied to *a*, and (*c*) the difference between *a* and *b* (specifically *a* subtract *b*).

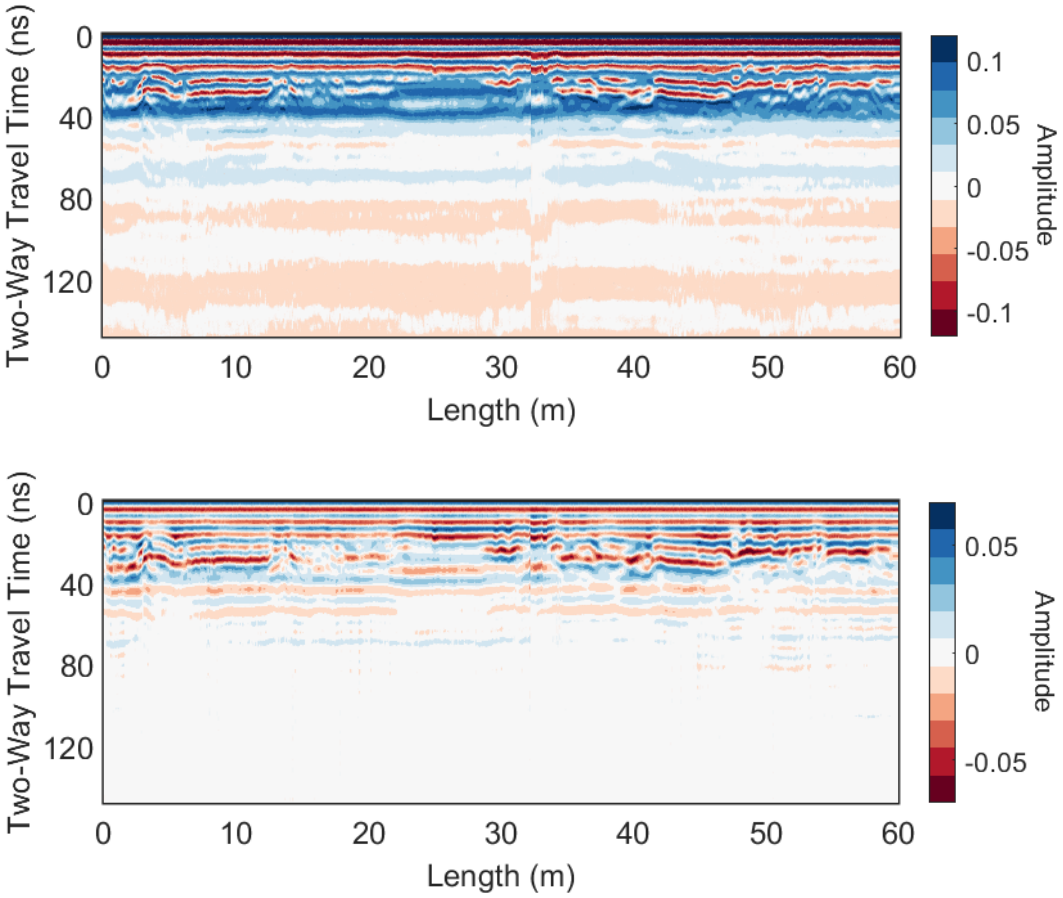


By leveraging the high-accuracy GPS coordinates of individual traces, the PCHIP interpolation method more accurately represents the true amplitude returns of the GPR profile compared to the local compression and stretch of amplitude returns in the traditional rubber-sheeting method. Further, the preservation of all traces in the profile allows for direct identification, tracking, and quantification of changes made to the profile for quality assurance and sensitivity analysis.

5.4 Data filtering in R

Data filtering is the fourth step in the overall post-processing script workflow. As stated previously, data filtering is required to reduce noise and enhance the visibility of significant sections of GPR profiles. This study used a preexisting data filtering function within RGPR as it fit within the script's workflow and filtered data in the manner intended for this effort. The RGPR `fFilter()` function applies a band-pass filter to the profile by performing a fast Fourier transform using a Hamming window. As written in RGPR, `fFilter()` requires a hard-coded variable f to define threshold frequency values. Band-pass threshold frequencies were defined as a function of the antenna used for data collection relative to the center frequency of the antenna (f_c, hertz). The threshold frequencies (hertz), `HighPass` and `LowPass`, were designed to be equivalent to 0.5 ($f_c/2$) and 1.5 ($2f_c$) times the center frequency of the antenna, respectively. This design allows the post-processing method to be flexible with any GPS antenna used in data collection. Figure 15 displays the effect of the data filtering script on the sample GPR profile. Compared to Figure 14*b*, which displays the profile after static data removal, time-zero correction, and distance normalization, the Figure 15 filtered profile has a smaller range in amplitude values (red, white, and blue color scheme is lighter). This is an expected result as the filter removes unwanted high (dark blue) and low (dark red) frequencies.

Figure 15. *Top*, duplicate of profile shown in Fig. 14*b*. *Bottom*, top profile with data filtering applied.



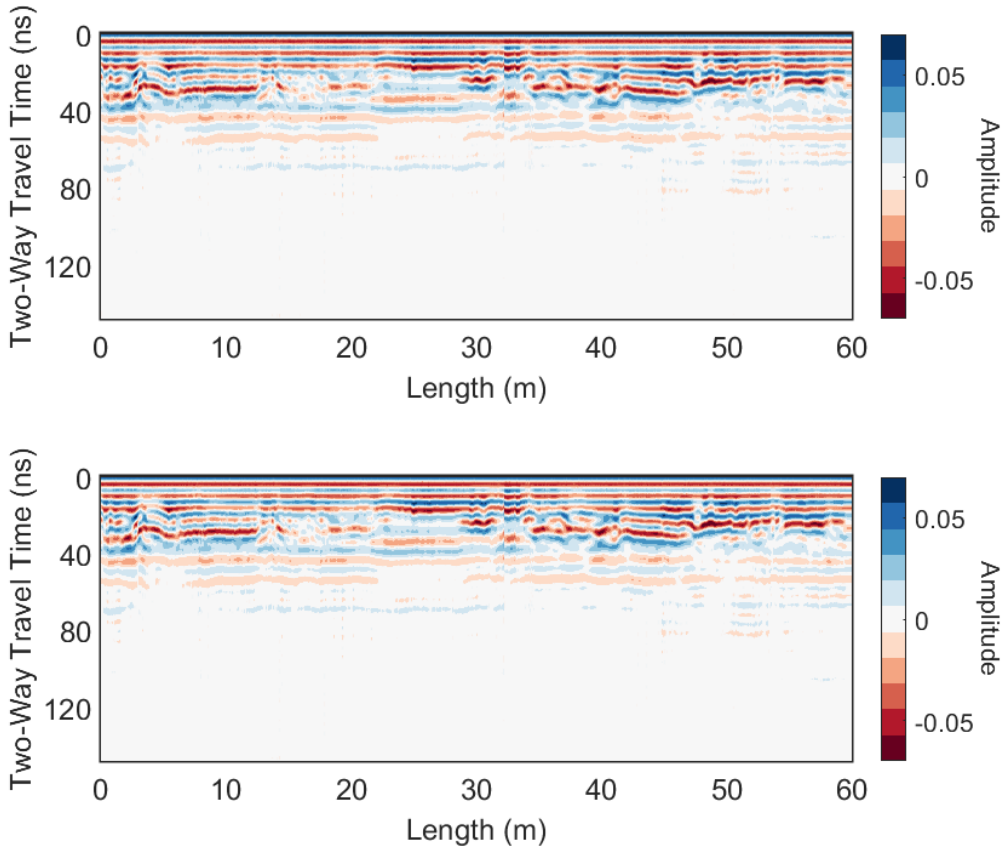
5.5 Stacking in R

Stacking is the fifth and final step in the overall post-processing script workflow. As previously mentioned, the purpose of stacking data is to improve the signal-to-noise ratio in a GPR profile while also maintaining data integrity. Stacking functions are applied to GPR profiles to improve the analysis of discrete reflections (e.g., a buried utility), to analyze continuous layers (e.g., bedrock or soil horizons), and to average out vertical "striping" caused by the antenna decoupling with the ground during data collection. However, overstacking traces can result in data becoming "smudged" and filtering out real, meaningful data.

To implement stacking in this script, the running or moving median—`runmed()`—R function was used to average traces within a GPR profile. The stacked output data are of the same dimensions as the original input variable (profile). As a default, the `runmed()` function calculates the median of

the entire dataset. If not used in the default manner, the parameter *k* sets the window size *a*. For this study, *k* is set to a default value of three, though an end user can select a different value if required for their processing needs. The value of *k* must be odd. If a user enters an even number, processing will be aborted; and the user will be alerted by a conditional statement requesting an odd value for *k*. A default value of three was selected as it most effectively removed high-frequency noise from the data without causing oversmoothing effects. The median function was selected as it is less sensitive to data outliers when compared to the mean. These parameters were selected in this study after evaluating data integrity based on a comparison between the input data and the stacked output data. Differences in reflection magnitude around near-surface responses were present when the window was increased higher than six traces. Figure 16 displays the effect of the stacking script on the sample GPR profile.

Figure 16. *Top*, duplicate of the profile shown in the Fig. 15 bottom panel. *Bottom*, top profile with stacking applied.



6 Conclusions and Recommendations

The script developed in this study eliminates the need for an SME to remove static data, time-zero correct, distance normalize, filter, and stack raw GSSI GPR profiles—which substantially reduces overall time spent on post-processing. At each step of development, comparison (difference) plots were made before and after each processing step was run to ensure each step worked as intended and that unwanted changes to the data were not made accidentally. As R programming is open source, this script also removes previous dependencies on costly and proprietary GPR-processing software.

Future enhancements to this script include data migration and expanding file-type compatibility. Data migration is a GPR post-processing step that refines velocity estimates by using hyperbolic summation. Migration is needed to improve depth calibrations and to collapse hyperbolas that could conceal deeper targets (Leach 2019). Additionally, the script developed in this study is currently compatible with only GSSI's DZT file format. Future efforts will expand this capability so that it is compatible with additional GPR manufacturer file types, such as from Sensors & Software, MALA, and Impulse Radar.

Future work will also involve combining this script with a GPR change-detection script that is currently under development to detect washouts in or under dams, levees, and roads. The change-detection script will be able to identify if subsurface changes have occurred between repeated data collections. Ultimately, the post-processing script developed in this study will serve as a precursor to GPR files processed through the change-detection script so that end users will be able to automatically post-process and interpret GPR data. Combined, these scripts may make GPR a more accessible and useful tool.

References

Abeynayake, C., and M. D. Tran. 2016. "Ground Penetrating Radar Applications in Buried Improvised Explosive Device Detection." In *Proceedings of the International Conference on Electromagnetics in Advanced Applications (ICEAA)*, 19–23 September, Cairns, QLD, Australia, 564–567. https://doi.org/10.1109/ICEAA.2016.77314555.

Al-Nuaimy, W. 1999. "Automatic Feature Detection and Interpretation in Ground-Penetrating Radar Data." PhD diss., University of Liverpool.

Amran, T. S. T., M. P. Ismail, M. A. Ismail, M. S. M. Amin, M. R. Ahmad, and N. S. M. Basri. 2017. "GPR Application on Construction Foundation Study." *IOP Conference Series: Materials Science and Engineering* 271:012089. http://dx.doi.org/10.1088/1757-899X/271/1/012089.

Baker, G. S., T. E. Jordan, and J. Pardy. 2007. "An Introduction to Ground Penetrating Radar (GPR)." In *Stratigraphic Analyses Using GPR*, edited by G. S. Baker and H. M. Jol. GSA Special Paper 432, 1–18. https://doi.org/10.1130/2007.2432(01).

Campbell, S., Z. Courville, S. Sinclair, and J. Wilner. 2017. "Brine, Englacial Structure and Basal Properties Near the Terminus of McMurdo Ice Shelf, Antarctica." *Annals of Glaciology* 58 (74): 1–11. https://doi.org/10.1017/aog.2017.26.

Ciampoli, L. B., R. Santarelli, E. M. Loreti, A. Ten, and A. Benedetto. 2020. "Structural Detailing of Buried Roman Baths Through GPR Inspection." *Archaeology Prospection* 2020:1–9. https://doi.org/10.1002/arp.1776.

Esri. 2016. "About Spatial Adjustment Rubbersheeting—Help | ArcGIS for Desktop". ArcMap. Accessed 29 January 2021. https://desktop.arcgis.com/en/arcmap/10.3/manage-data/editing-existing-features/about-spatial-adjustment-rubbersheeting.htm.

Fritsch, F. N., and R. E. Carlson. 1980. "Monotone Piecewise Cubic Interpolation." *SIAM Journal on Numerical Analysis* 17 (2): 238–246. https://doi.org/10.1137/0717021.

GSSI (Geophysical Survey Systems Inc.). 2017a. *RADAN 7 Manual*. MN43-199 Rev G. Nashua, NH: Geophysical Survey Systems Inc. https://www.geophysical.com/wp-content/uploads/2017/10/GSSI-RADAN-7-Manual.pdf.

GSSI (Geophysical Survey Systems Inc.). 2017b. *Antennas Manual*. MN30-903 Rev G. Nashua, NH: Geophysical Survey Systems Inc.

Huber, E., and G. Hans. 2018. "RGPR—An Open-Source Package to Process and Visualize GPR Data." In *2018 17th International Conference on Ground Penetrating Radar*, 18–21 June, Rapperswil, Switzerland, 1–4. https://doi.org/10.1109/ICGPR.2018.8441658.

Leach, P. 2019. *RADAN 7 for Archaeology, Forensics, and Cemeteries*. Nashua, NH: Geophysical Survey Systems Inc. https://www.geophysical.com/wp-content/uploads/2019/04/MN43203A-RADAN-for-Archaeology-Cemeteries-and-Forensics.pdf.

Morcous, G., and E. Erdogmus. 2009. *Use of Ground Penetrating Radar for Construction Quality Assurance of Concrete Pavement*. Nebraska Department of Roads Research Reports 79. Omaha, NE: University of Nebraska–Lincoln. http://digitalcommons.unl.edu/ndor/79.

Ristić, A., M. Govedarica, L. Pajewski, M. Vrtunski, and Z. Bugarinović. 2020. "Using Ground Penetrating Radar to Reveal Hidden Archaeology: The Case Study of the Württemberg-Stambol Gate in Belgrade (Serbia)." *Sensors* 20 (3): 607. https://doi.org/10.3390/s20030607.

Shao, S., X. Guo, and H. Ding. 2018. "Temporal Ground Penetrating Radar (GPR) Imaging of an Oil Release Within a Porous Medium: A Description of Anomalous GPR Characteristics during the Degradation Process and a Contaminated Area Determination Method." In *ICEG 2018: Proceedings of the 8th International Congress on Environmental Geotechnics Volume 1*, 850–858. Singapore: Springer.

Steelman, C. M., D. R. Klazinga, A. G. Cahill, A. L. Endres, and B. L. Parker. 2017. "Monitoring the Evolution and Migration of a Methane Gas Plume in an Unconfined Sandy Aquifer Using Time-Lapse GPR and ERT." *Journal of Contaminant Hydrology* 205:12–24. https://doi.org/10.1016/j.jconhyd.2017.08.011.

Yelf, R. 2004. "Where is True Time Zero?" In *Proceedings of the Tenth International Conference on Ground Penetrating Radar*, 21–24 June, Delft, The Netherlands, 279–282. https://doi.org/10.1109/ICGPR.2004.179979.

Appendix: Post-Processing Script

To run this GPR post-processing script, users will need to download and install R and RStudio and load the following libraries: RGPR, dplyr, geosphere, pracma, lubridate, and bpa. This script is compatible with GSSI's DZT file types.

```
##############################################
#
# GPR Post Data Collection Processing Script
#
#     Static Data Removal
#     Time-Zero Correction
#     Distance Normalization
#     Data Filtering
#     Stacking
#
# Developed by:
#     ERDC/CRREL
#     09 August 2021
#
# Last updated:
#     03 June 2022
##############################################

########## LOAD REQUIRED LIBRARIES ##########
library(RGPR)
library(dplyr)
library(geosphere)
library(pracma)
library(lubridate)
library(bpa)

########## SPECIFY INPUT VARIABLES ##########
# Specify the path to the file directory
iDir <- "C://path//to//GPR//profiles"

# Set path as the working directory
setwd(iDir)

# List of all GPR profiles in iDir folder with extension *.DZT
dzt_files <- list.files(pattern = "*.DZT")

########## DEFINE / DECLARE FUNCTIONS ##########

### Declare remove static function
removeStaticDataGPR <- function(GPR_file){

  # Specify the GPR filename
  GPRFile <- GPR_file

  # Define number of traces
```

```
  numTrace <- length(GPRFile@traces)

  # Sampling frequency
  freq <- 24
  freq_int <- seq(1,numTrace,freq)
  if(freq_int[length(freq_int)] != numTrace){ freq_int <-
c(freq_int,numTrace) }

  # Pull lat/lon
  lon <- GPRFile@coord[,1]
  lat <- GPRFile@coord[,2]

  # Total linear distance traveled
  total_linear_distance_traveled <-
dist(c(lon[1],lat[1]),c(lon[numTrace],lat[numTrace]))

  # Actual distance traveled -
  # computed in computePositionsFromLonLat()
  actual_distance <- GPRFile@pos
  actual_distance <- actual_distance[freq_int]

  # Derivative - find the derivative of actual distance traveled
  #  and theoretical
  #  --> distance traveled at the sampling frequency.
  theoretical_distance <- linspace(0,total_linear_distance_trav-
eled,numTrace) #[m]
  theoretical_distance <- theoretical_distance[freq_int]
  derivative <- diff(actual_distance) / diff(theoretical_dis-
tance)

  # Set threshold for defining static data - relative to walking
speed
  static_data_int <- which(derivative <= (median(derivative) -
std(derivative)))
  static_data_int <- c(static_data_int, length(freq_int))

  # Remove static data from profile'
  static_data <- c()
  for(i in seq(2,length(static_data_int))){
    tmp <- seq(freq_int[static_data_int[i]]-
freq,freq_int[static_data_int[i]])
    static_data <- append(static_data,tmp,af-
ter=length(static_data))
  }

  GPRFileC1C2 <- GPRFile[,-static_data]

  # Rewrite actual distance traveled'
  update_actual_distance <- c()
  for(i in seq_along(GPRFileC1C2@traces)){
    update_actual_distance[i] <-
      dist(c(GPRFileC1C2@coord[1,1],GPRFileC1C2@coord[1,2]),
      c(GPRFileC1C2@coord[i,1],GPRFileC1C2@coord[i,2])) # [m]
  }
```

```
GPRFileC1C2@pos <- update_actual_distance
  GPRFileC1C2@traces <- seq(1,length(GPRFileC1C2@traces),1)

  return(GPRFileC1C2)

}

### Declare time zero correction function
timeZeroCorrectionGPR <- function(GPRC1C2_file){

  # Find where max amplitude value is located (in depth) at each
trace in profile
  maxamploc <- c()
  for (i in 1:length(GPRC1C2_file@traces)){
    maxamploc[i] <- which.max(GPRC1C2_file@data[,i])
  }

  # Find median location of max amp across profile length
  med_maxamploc <- as.integer(median(maxamploc))

  # Find the time (ns) associated with that amplitude (TZ correct
value)
  maxamptime <- GPRC1C2_file@depth[med_maxamploc]

  # Remove all the rows up to the TZ correct value row
  GPRFileC1C2TZ <- GPRC1C2_file[-1:-med_maxamploc,]

  # Shift the time values to start at 0 ns
  NewTime0 <- GPRFileC1C2TZ@depth[1]
  GPRFileC1C2TZ@depth <- GPRFileC1C2TZ@depth - NewTime0

  return(GPRFileC1C2TZ)
}

# Declare distance normalization function
distanceNormalizeDataGPR <- function(GPRC1C2TZ_file,ndist=2){

  # Specify the GPR variable
  GPRFileC1C2TZ <- GPRC1C2TZ_file

  # define vector of trace numbers
  trNo <- GPRFileC1C2TZ@traces

  # pull lat/lon coordinates
  lon <- GPRFileC1C2TZ@coord[,1]
  lat <- GPRFileC1C2TZ@coord[,2]

  # total distance of profile - linear distance
  actual_distance <- GPRFileC1C2TZ@pos

  ### Unit distance [m]
  # Create decision criteria for ndist in future; for now, play
with it
  NDIST <- ndist
```

```
  # unit distance vector
  Ax <- NDIST*(0:(actual_distance/NDIST))

  # determine index of linear distance at unit distance
  ind0 <- Ax
  ind0[1] <- 1
  ind <- 1
  scanInd <- Ax[1]
  for(i in 1:(length(Ax)-1)){
    chk_dist <- Ax[i]
    while(as.integer(chk_dist) != as.integer(Ax[i+1])){
      chk_dist <- actual_distance[ind]
      ind <- ind + 1
    }
    scanInd <- append(scanInd, ind-1, after=length(scanInd))
  }

  # append last point to position vector
  extrapIx <- actual_distance - Ax[length(Ax)]
  Ax <- append(Ax, max(Ax)+extrapIx, after=length(Ax))
  scanInd <- append(scanInd, trNo[length(trNo)], af-
ter=length(scanInd))
  scanInd[1] <- 1

  # ----------DISTANCE NORMALIZE ---------------
  # define empty radargram
  data <- GPRFileC1C2TZ@data
  dataDN <- zeros(size(data,1), length(trNo))

  # DISTANCE NORMALIZE -- PCHIP method
  if(any(diff(actual_distance)<0)){
    ax1 <- sort(actual_distance)
  } else {
    ax1 <- actual_distance
  }
  ax2 <- interp1(scanInd,Ax,xi=trNo,'linear')
  for (kk in 1:size(data,1)){
    dataDN[kk,] <- pchip(ax1,data[kk,],ax2) # Interpolated Radar-
gram
  }

  # Create DN profile
  GPRFileC1C2TZDN <- GPRFileC1C2TZ
  GPRFileC1C2TZDN@data <- dataDN
  GPRFileC1C2TZDN@pos <- ax2

  return(GPRFileC1C2TZDN)

}

### Declare data filtering function
dataFilteringGPR <- function(GPRC1C2TZDN_file){

  # Specify the GPR variable name
  GPRFile <- GPRC1C2TZDN_file
```

```
  # Pull antenna frequency
  Antenna<-GPRFile@freq

  # Set the high pass frequency
  HighPass <-Antenna*0.5

  # Set the low pass frequency
  LowPass<-Antenna*0.5 + Antenna

  # Combine low and high for a band pass frequency range
  f<-c(HighPass, LowPass)

  # Band pass filter using frequencies specified above
  GPRC1C2TZDNFIR <- fFilter(GPRFile, f, type = "bandpass",
plotSpec = FALSE)

  # Return filtered profile
  return(GPRC1C2TZDNFIR)

}

### Declare stacking function
stackingGPR <- function(GPRC1C2TZDNFIR_file,nStack = 3){

  # error check for number of stacks (nStack)
  # must be odd
  if (nStack %% 2 == 0){
    stop("Error: number of stacks must be odd")
  }

  GPRC1C2TZDNFIRS <- GPRC1C2TZDNFIR_file
  sData <- GPRC1C2TZDNFIRS@data
  nSamples <- length(GPRC1C2TZDNFIRS@depth)

  # apply running median filter to each layer in profile
  for(i in 1:nSamples){
    sData[i,] <- runmed(GPRC1C2TZDNFIRS@data[i,],nStack)
  }

  #Replace original data with stacked data
  GPRC1C2TZDNFIRS@data <- sData

  return(GPRC1C2TZDNFIRS)
}
```

Abbreviations

CRREL	Cold Regions Research and Engineering Laboratory
GIS	Geospatial information system
GNSS	Global Navigation Satellite System
GPR	Ground-penetrating radar
GSSI	Geophysical Survey Systems, Inc.
IIR	Infinite impulse response
PCHIP	Piecewise cubic Hermite interpolating polynomial
SME	Subject matter expert
TWTT	Two-way travel time

REPORT DOCUMENTATION PAGE

Form Approved

OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
September 2022	Technical Report / Final	FY19–FY22

4. TITLE AND SUBTITLE

Automated Ground-Penetrating-Radar Post-Processing Software in R Programming

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT

0602146A

6. AUTHOR(S)

Samantha N. Cook, Marissa J. Torres, Nathan J. Lamie,
Lee J. Perren, Scott M. Slone, and Bonnie J. Jones

5d. PROJECT NUMBER

AT2

5e. TASK NUMBER

SAT202

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

US Army Engineer Research and Development Center (ERDC)
Cold Regions Research and Engineering Laboratory (CRREL)
72 Lyme Road
Hanover, NH 03755-1290

8. PERFORMING ORGANIZATION REPORT NUMBER

ERDC/CRREL TR-22-18

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Headquarters, US Army Corps of Engineers
Washington, DC 20314-1000

10. SPONSOR/MONITOR'S ACRONYM(S)

USACE

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Ground-penetrating radar (GPR) is a nondestructive geophysical technique used to create images of the subsurface. A major limitation of GPR is that a subject matter expert (SME) needs to post-process and interpret the data, limiting the technique's use. Post-processing is time-intensive and, for detailed processing, requires proprietary software. The goal of this study is to develop automated GPR post-processing software, compatible with Geophysical Survey Systems, Inc. (GSSI) data, in open-source R programming. This would eliminate the need for an SME to process GPR data, remove proprietary software dependencies, and render GPR more accessible. This study collected GPR profiles by using a GSSI SIR4000 control unit, a 100 MHz antenna, and a Trimble GPS. A standardized method for post-processing data was then established, which includes static data removal, time-zero correction, distance normalization, data filtering, and stacking. These steps were scripted and automated in R programming, excluding data filtering, which was used from an existing package, RGPR. The study compared profiles processed using GSSI software to profiles processed using the R script developed here to ensure comparable functionality and output. While an SME is currently still necessary for interpretations, this script eliminates the need for one to post-process GSSI GPR data.

15. SUBJECT TERMS

Automated software, Data processing--Automation, GPR, Ground penetrating radar, Post-processing, R (Computer program language), RGPR, R programming, Subterranean

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (include area code)
Unclassified	Unclassified	Unclassified	SAR	44	