GPRPy: Open-source ground-penetrating radar processing and visualization software

Alain M. Plattner[1]

https://doi.org/10.1190/tle39050332.1

Abstract

GPRPy is an open-source ground-penetrating radar software compatible with a range of ground-penetrating radar systems. Data processing and plotting can be performed by using graphical user interfaces or scripts that are generated automatically from the graphical user interfaces. This makes learning the software easy, and it enables researchers to share their scripts as part of a publication to ensure reproducible research. GPRPy enables profile data processing and visualization, velocity analysis, interpolation of 3D data cubes from profile data, and 3D interpolation for interfaces visible in multiple profiles. The software is written in Python and runs on all major operating systems.

Introduction

Since its original patents a century ago, ground-penetrating radar (GPR) has become one of the most popular methods to image shallow subsurface structures. GPR owes its popularity to relatively low cost in the field and high subsurface resolution compared to other geophysical methods. As a result, GPR has been a staple for archaeological surveys (e.g., Goodman and Piro, 2013; Urban et al., 2014), stratigraphy (e.g., Bristow and Jol, 2003; Pitman et al., 2019), hydrology (e.g., Rubin and Hubbard, 2005; Klotzsche et al., 2018), contaminant mapping (e.g., Wilson et al., 2009), soil science (e.g., Butnor et al., 2003), glaciers and ice sheets (e.g., Merz et al., 2015), civil engineering (e.g., Klysz and Balayssac, 2007; Lai et al., 2018), and ecosystems research (e.g., Paz et al., 2017). Jol (2008) provides an excellent overview of theory and applications of GPR.

Many GPR investigations do not require advanced processing to yield useful results. Nevertheless, the majority of GPR research publications use proprietary closed-source software provided by the GPR system manufacturer or independent packages (Jol, 2008). Reliance on proprietary software has a number of inherent disadvantages. First, the software has limited flexibility and does not fully allow users to adapt it to their needs. Second, reproducible research is limited. To recreate the processing steps published in an article, researchers need to have access to the proprietary software. This limitation will become increasingly severe as publishers and funding agencies push for transparent research. Third, courses teaching GPR processing are restricted by access to proprietary software.

Open-source software aims to overcome these obstacles. In recent years, a small number of open-source GPR processing and visualization packages have become available. Among these packages, MATGPR (Tzanis, 2006) is likely the most prominent. It has been available for more than a decade and includes a graphical user interface (GUI). MATGPR recently became closed source, but older open-source versions may still be available.

A disadvantage of MATGPR and other MATLAB-based GPR programs is that they require MATLAB, a proprietary software. Freely available MATLAB-like alternatives, such as Octave (Eaton et al., 2019), are typically incompatible with MATLAB GUIs. In general, GUIs greatly facilitate learning processing software. However, if a software is purely GUI controlled and cannot be automated, such as through computer scripts containing the processing and visualization steps, large-scale projects that require processing a large number of profiles are rendered unnecessarily tedious, and reproducible research is made more difficult.

On the other end of the open-source GPR software spectrum lies the recently published package RGPR (Huber and Hans, 2018), which is based on the programming language R. Instead of using a GUI, RGPR requires investigators to write scripts in R. Consequently, the learning curve for mastering RGPR is steep for investigators with little programming experience.

Irlib (Wilson et al., 2013; Wilson, 2017), another open-source GPR software, is Python based. It has a GUI for picking and data filtering Blue System IceRadar data files, but it requires the user to write Python scripts to process Sensors and Software (.dt1) files. As is the case of RGPR, this may impede users with little programming experience.

The aim of GPRPy is to provide a GPR processing and visualization platform that is freely available, powerful, easy to use, programmable, expandable, and that facilitates reproducible research. GPRPy is built on the free programming language Python (version 3). Hence, it can be used without requiring a proprietary programming environment such as MATLAB. Python, and therefore GPRPy, are compatible with a wide range of operating systems.

GPRPy's data processing is optimized for speed and memory consumption by making use of data structures and functions from Python's NumPy module. The implemented GUIs (one for profile data processing and one for velocity analysis) simplify learning the software. Both GPRPy GUIs have the ability to automatically create Python scripts by internally storing the processing steps used in the GUI. The automatically generated Python scripts, together with the raw data, can be shared to allow researchers to reproduce the data processing and visualization steps carried out for an investigation. GPRPy is hosted on a collaborative software-developing website given in the section "Data and materials availability." This allows the community to use GPRPy as a base code for implementing their own data processing procedures and to make them instantly available. The current version of GPRPy can import radar data from

[1]University of Alabama, Tuscaloosa, Alabama, USA. E-mail: amplattner@ua.edu.

Sensors and Software GPR systems (.dt1 files), GSSI (.dzt files), MALÅ (.rd3 files), and ENVI standard BSQ files. The remainder of this article highlights some of the currently implemented processing and visualization procedures and provides data processing examples.

Software design

The GPRPy software package consists of two GUIs, one for profile data processing (Figure 1) and one for common-midpoint (CMP) or wide-angle reflection and refraction (WARR) data (Figure 2). The package also includes a series of functions for more advanced processing, such as the generation of 3D data cubes by interpolating profiles, interpolation of surfaces between picked points, and merging profiles. The computer code for the GUI is separated from the computer code that implements the data processing and visualization. This separation allows users to write Python code that directly accesses all of GPRPy's capabilities without needing to use the GUI. GPRPy exploits this separation through its automatically generated scripts.

GPRPy's constant-velocity *f–k* migration routine is incorporated from the software package Irlib (Wilson et al., 2013; Wilson, 2017) and is a Python translation of the original MATLAB code by Margrave and Lamoureux (2019). This migration method was originally introduced by Stolt (1978) for seismic data processing. Migration can remove artifacts resulting from arrivals that were not reflected perpendicularly below the transmitter receiver midpoint (Yilmaz, 2001). Among other things, a migration procedure leads to collapsing of hyperbolae caused by reflections from spatially confined objects, transformation of bow ties into synclines, and correction of dipping angles of reflectors. As a result, a migrated profile more closely resembles a geologic cross section than an unmigrated profile.

Common-offset profiles. Common-offset profiling, the most widely used GPR data acquisition scheme, involves moving transmitter and receiver antennae with fixed separation (offset) along a profile line. Three-dimensional data cubes are typically created by running parallel and intersecting profiles, processing the profiles individually, and then interpolating the processed profiles. GPRPy's profile GUI (Figure 1) enables the most commonly used processing steps. Subsurface velocities can be estimated by fitting hyperbolae to reflections from pipes or other spatially confined objects. Structures visible in a profile can be picked from within the GUI and exported into an ASCII file. If the profile has been topographically corrected using 3D coordinates, the picked points will additionally be exported as 3D coordinates, with locations interpolated between the 3D profile coordinates. The 3D coordinates of the picked points can later be used to create surfaces interpolating the same reflector that was imaged in different profiles. The section "Interpolated picked reflectors" shows an example of such an approach. GPRPy can export profiles as 3D structured grid files, which can be visualized with ParaView (Ayachit, 2015), MayaVi (Ramachandran and Varoquaux, 2011), or other 3D rendering programs capable of reading VTK files.

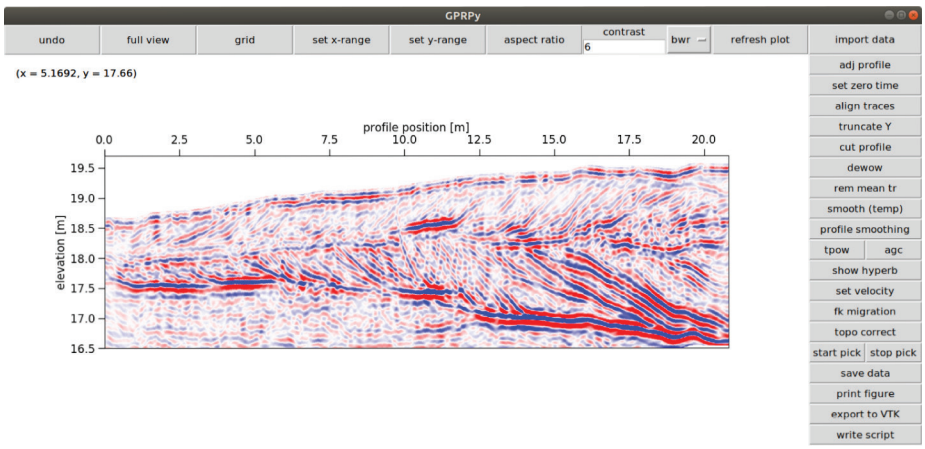

Figure 1. GUI for GPR common-offset profile data processing.

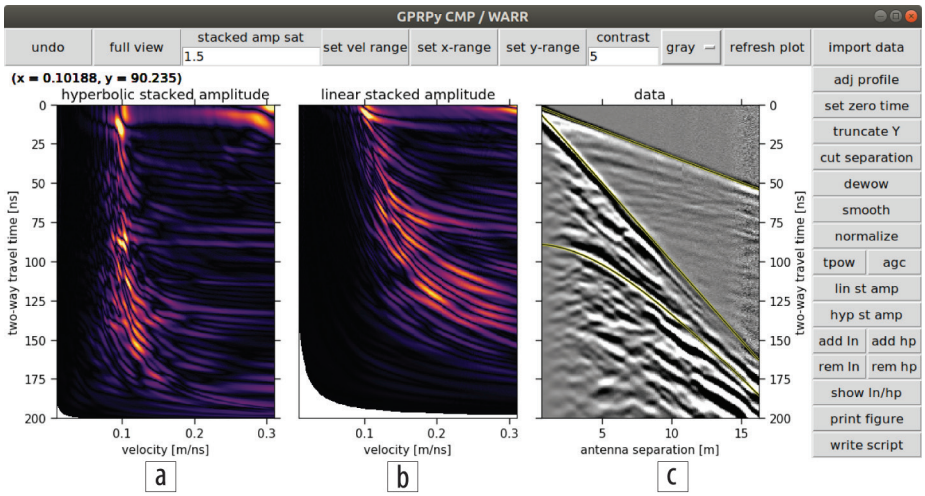

Figure 2. GUI for velocity analysis from CMP or WARR data. (a) Stacked amplitudes for hyperbolically shaped arrival-time patterns. (b) Stacked amplitudes for linearly shaped arrival-time patterns. (c) CMP or WARR data.

Velocity analysis from CMP/WARR data. All GPR studies require knowledge of a subsurface velocity to transform two-way traveltime to depth. The time-to-depth transformation currently implemented in GPRPy assumes a single average velocity to the target depth, such as the root mean square (rms) velocity by Dix (1955). This velocity can be obtained from tables, by fitting hyperbolae to profiles crossing buried pipes or other spatially confined reflectors, or through CMP or WARR surveys. In a CMP survey, transmitter and receiver antennae are placed on either side of a designated midpoint and are subsequently moved equidistantly away while recording traces. In a WARR survey, either the

transmitter or receiver antenna is fixed, and the other antenna is moved increasingly away while recording traces. In WARR and CMP data sets for flat topography, airwave and groundwave arrival times depend linearly on the antenna separation. The slope of this linear relationship is $1/v$, where v represents the velocity of the air or the average velocity of the shallowest portion of the ground. Reflections from horizontal interfaces have a hyperbolic shape given by $t = \sqrt{x^2 + 4d^2} / v_{rms}$. Here, x is the antennae separation, t is the two-way traveltime, d is the depth to the horizontal reflector, and v_{rms} is the average velocity between the surface and reflecting interface. GPRPy's velocity analysis GUI for CMP or WARR data (Figure 2) enables users to draw lines and hyperbolae on top of the data to estimate velocities using a trial-and-error approach. A coherency analysis presents an automatized approach for velocity determination. The implemented stacked amplitude analysis calculates the sum over all pixels in the CMP or WARR data that follow the idealized linearly or hyperbolically shaped arrival-time pattern for a given v_{rms} and t_0. If all pixels along the theoretical line or hyperbola have the same sign, such as when following a line or hyperbola visible in the data, then the sum of the pixel values has a large magnitude. On the other hand, if the pixel values switch signs, for example when the theoretical line or hyperbola crosses troughs and peaks of recorded waves, the resulting stacked amplitude has a low absolute value. The center and left panel of GPRPy's CMP/WARR GUI, as shown in Figures 2a and 2b, displays stacked amplitudes for linearly or hyperbolically shaped arrival-time patterns, respectively, for a chosen range of v_{rms} and t_0.

Automatic script generation. While using either the GUI for common-offset data or the GUI for CMP/WARR data, GPRPy stores each processing step and can generate a Python script containing the Python commands for each step. This script can be run from the command line to reproduce all of the data import, processing, visualization, and export steps. Users can edit individual processing steps and rerun the analysis without needing to repeat every step in the GUI. Such scripts, together with the raw data, could be shared as part of a publication. Since GPRPy is open source and can be installed on any system that can run Python, data processing becomes fully reproducible. Another advantage of automatically created scripts is that they can be used to automatically apply the same processing to a number of profiles (e.g., to create a 3D data cube).

Three-dimensional data. If a set of densely spaced and/or intersecting profiles is available, GPRPy can create a 3D data cube by interpolating the provided profiles and exporting the resulting data cube as a VTK file. The section "Data cube interpolation" shows a nearest-neighbor interpolation of a dense grid of 46 × 46 profiles collected for an archaeological study. When multiple profiles are available but are not densely spaced, 3D interpolation may not be suited, as it could create misleading artifacts due to unevenly distributed sampling. However, if expressions of the same structure are visible in multiple profiles, picked locations of the structure from individual profiles can be interpolated. The section "Reflector picking and 3D interpolation" shows an example of such an approach.

Data processing examples

The examples provided in this section are not intended as full geophysical investigations but highlight select capabilities of GPRPy. The data presented here were collected by a range of student investigators using GSSI and Sensors and Software GPR systems.

Common-offset profile. To image the interface between the El Capitan Meadow rock avalanche deposit (Yosemite Valley, California, USA) and the valley deposits underneath it, Liu and Plattner (2018) collected a common-offset GPR profile (Figure 3) in spring 2017. They used a Sensors and Software pulseEKKO PRO system with a 50 MHz antenna. The spacing between recorded traces was 0.6 m. Receiver and transmitter antennae were held at an approximately constant separation of 1 m using strings, but the rugged terrain led to small variations in antenna separation visible in the uneven airwave arrival times in Figure 3a. A reflector is visible in the raw data (Figure 3a), starting at the surface at approximately 10 m along the profile and dipping underground toward a two-way traveltime of 200 ns at approximately 80 m along the profile. However, the airwave arrivals are dominating. After dewow (high-pass along-trace filtering), aligning individual traces by their maximum amplitude, and subtracting the average trace from the profile, the reflector becomes more prominent (Figure 3b). To enhance the visibility of the reflector, I applied a t-power gain with exponent 1.2, together with smoothing along traveltime (running-mean replacement with window width of 10 samples, the equivalent of a low-pass filter) and smoothing along the profile (by replacing each trace with four copies of that trace and then applying a running-mean replacement over six traces). The last smoothing step led to a smoother, easier to interpret image without creating significant artifacts. The rms velocity 0.1 m/ns was obtained as explained in the section "Velocity analysis." An *f-k* migration additionally helped clean up the profile and corrected the dip of the reflector between 10 and 80 m along the profile (Figure 3c). Finally, the topographic correction from GPS points collected every 3 m along the profile shows that the reflector is roughly horizontal underneath the topography.

Data cube interpolation. In fall 2013, Princeton University first-year undergraduate students collected a dense grid of GPR profiles at an agricultural field close to a previously unearthed archaeological site in Cyprus. The raw data consist of 92 profiles, each 9 m long, arranged in a 46 × 46 profile grid, with 20 cm spacing between the profiles. The students used a GSSI GPR system with a 400 MHz antenna. Data quality of individual profiles was low due to high clay content of the soil, but combining the profiles revealed previously unknown structures (Figures 4 and 5). Processing of the individual profiles was minimal and only included subtraction of an average trace per profile and *f-k* migration for a velocity of 0.06 m/ns. A nearest-neighbor interpolation of the processed profiles (Figure 4) revealed roughly 1 m thick bands of consistent reflections at depths that are comparable to the depths of walls found in an adjacent excavation site. The first band in Figure 4 runs between 8 m on the bottom x axis and 6 m on the top x axis. The second band runs between 4.5 m on the left y axis and 6 m on the right y axis but ends where it meets the first band. Based on the location, geometry, and width of these reflectors, the investigators interpreted these bands as buried walls. To allow for a contrast strong enough

to be able to isolate the wall from its surroundings, I applied Gaussian smoothing to the data cube (Figure 5) and visualized the result using ParaView (Ayachit, 2015).

Interpolated picked reflectors. In fall 2017, Princeton University students collected a number of GPR lines crossing the Dune of Pilat in France. Time constraints restricted the students to only collecting a small number of profiles with large gaps in between. Hence, 3D interpolation of the profiles would not yield useful results. The profiles shown in Figure 6 run across the dune in a southeast direction. The linear reflectors visible in both profiles are interpreted as arising from internal surfaces cutting across both profiles. To create a 3D model of one of the surfaces, I used GPRPy's profile GUI (Figure 1) to pick points where the surface intersects the profiles. I then interpolated these points with spline interpolation of polynomial order 2 in the east direction and polynomial order 1 in the north direction (Figure 6).

Velocity analysis. To obtain the rms velocity for the study shown in the section "Common-offset profile" (Figure 3), Liu and Plattner (2018) collected a WARR data set on top of the El Capitan Meadow rock avalanche using a Sensors and Software pulseEKKO PRO system with 100 MHz antennae. Liu and Plattner (2018) kept the transmitter antenna in a fixed location, connected the receiver antenna to a trigger wheel, and recorded traces every 0.1 m while moving the receiver antenna away from the transmitter. The raw data (Figure 7a) show linearly shaped arrival-time patterns from the airwave and ground wave and a hyperbolically shaped arrival-time pattern from a buried reflector. To increase the relative signal strength for wider antennae offsets, I applied trace normalization after dewowing the data (Figure 7b). The stacked amplitudes for the WARR data for linearly shaped arrival-time patterns (Figure 2b) show two narrow regions of high values for early t_0: one for $v_{rms} \approx 0.1$ m/ns and one for $v_{rms} \approx 0.3$ m/ns. The slower region corresponds to the steeply dipping ground-wave arrival highlighted in the data panel (Figure 2c). The faster region corresponds to the highlighted gently dipping airwave arrival. For later arrival

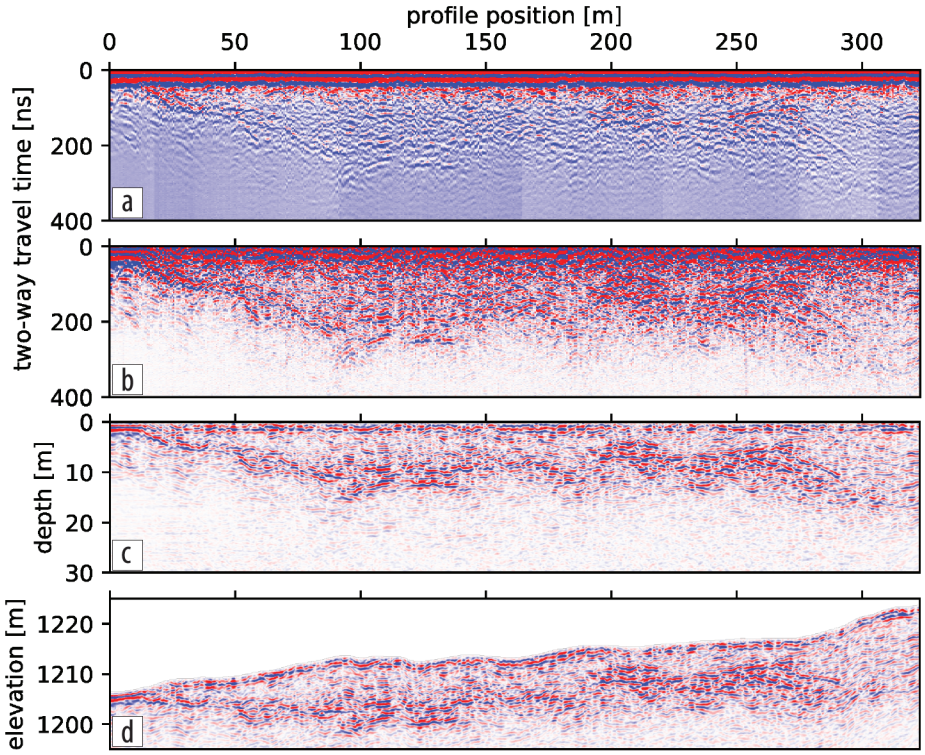

Figure 3. Example of common-offset data processing steps. (a) Raw data. (b) After trace alignment, dewow, and mean trace removal. (c) After along-trace smoothing (window width 10 samples), $t^{1.2}$ gain, time-to-depth conversion using an rms velocity of 0.1 m/ns, *f-k* migration, and along-profile smoothing (oversampling factor 4, running-mean window width six traces). (d) After topographic correction.

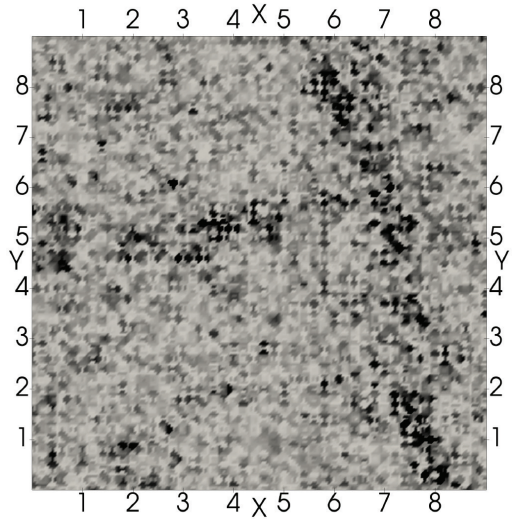

Figure 4. Horizontal slice at 0.35 m depth through data cube of nearest-neighbor interpolated profiles arranged in a 46 × 46 grid. Axes are in meters.

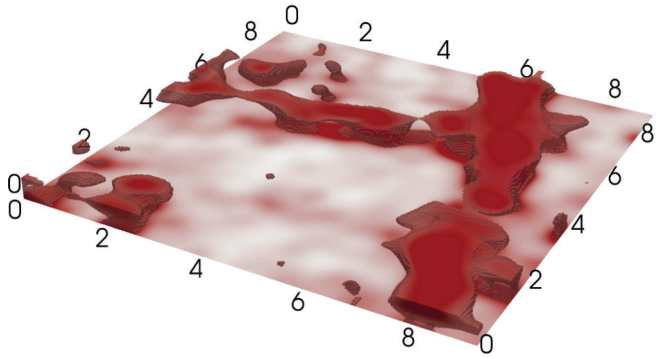

Figure 5. Extraction of 3D structure visible in Figure 4 through Gaussian smoothing. Axes are in meters, vertical exaggeration is by a factor of 4.

times, the stacked amplitudes for linearly shaped arrival-time patterns show smeared-out regions of high value, for which no clear linearly shaped arrival-time patterns are visible in the data. The stacked amplitudes for hyperbolically shaped arrival-time patterns (Figure 2a) contain two regions of high value for $v_{rms} \approx 0.1$ m/ns before $t_0 = 100$ ns: one at approximately 5 ns and one at 90 ns. Later high-value regions are smeared out, and corresponding hyperbolically shaped arrival-time patterns are difficult to see in the data. The hyperbolically shaped arrival-time pattern corresponding to values close to $v_{rms} \approx 0.1$ m/ns and $t_0 \approx 90$ ns is highlighted in Figure 2c. Liu and Plattner (2018) concluded from this investigation that $v_{rms} \approx 0.1$ m/ns down to a reflector at a two-way traveltime of 90 ns, which corresponds to a depth of 4.5 m.

Conclusions

GPRPy is an open-source GPR software package that includes GUIs for profile data processing and velocity analysis. Additional functions allow for 3D data interpolation. GPRPy is scriptable and can generate scripts automatically from the GUIs. In this article, I highlighted the basic structure of GPRPy and provided examples for some of the main features. The software, data sets, and scripts to reproduce the results are publicly available. GPRPy is available through a collaborative software-development platform, allowing the community to adapt the software to its needs and extend its capabilities. Researchers developing novel data processing algorithms can implement their work in GPRPy and avoid writing their own data import, visualization, and GUIs. Investigators using GPRPy as a tool for data processing can submit automatically generated scripts together with data as supplemental material with their research articles to allow for research reproducibility. TLE

Acknowledgments

Thanks to M. Pacheco for testing GPRPy and suggesting improvements and to F. J. Simons and A. Fonseca for their feedback on the article. F. J. Simons, A. C. Maloof, D. L. Cassidy Nolan, S. Paliskara, and Z. Zhang provided the data for Figures 1 and 6. C. Liu provided the data for Figures 2 and 3. J. Smith and K. Ryan provided the data for Figure 5. Figures 5 and 6 were created using ParaView (Ayachit, 2015). This material is based on work supported by the National Science Foundation under grant no. EAR-1550732.

Data and materials availability

GPRPy is hosted on https://github.com/nsgeophysics/gprpy and is permanently available from https://www.doi.org/10.5281/zenodo.2556982. Installation instructions are provided at https://nsgeophysics.github.io/gprpy/. Data and Python scripts to reproduce the examples presented in this article are available from https://www.doi.org/10.5281/zenodo.3357978.

Corresponding author: amplattner@ua.edu

References

Ayachit, U., 2015, The ParaView guide: A parallel visualization application: Kitware.

Bristow, C. S., and H. M. Jol, 2003, Ground penetrating radar in sediments: Geological Society of London, https://doi.org/10.1144/GSL.SP.2003.211.

Butnor, J. R., J. A. Doolittle, K. H. Johnsen, L. Samuelson, T. Stokes, and L. Kress, 2003, Utility of ground-penetrating radar as a root biomass survey tool in forest systems: Soil Science Society of America Journal, **67**, no. 5, 1607–1615, https://doi.org/10.2136/sssaj2003.1607.

Dix, C. H., 1955, Seismic velocities from surface measurements: Geophysics, **20**, no. 1, 68–86, https://doi.org/10.1190/1.1438126.

Eaton, J. W., D. Bateman, S. Hauberg, and R. Wehbring, 2019, GNU Octave version 5.2.0 manual: A high-level interactive language for numerical computations, https://www.gnu.org/software/octave/doc/v5.2.0/, accessed 30 March 2020.

Goodman, D., and S. Piro, 2013, GPR remote sensing in archaeology: Springer, https://doi.org/10.1007/978-3-642-31857-3.

Huber, E., and G. Hans, 2018, RGPR — An open-source package to process and visualize GPR data: Proceedings of the 17th International Conference on Ground Penetrating Radar, https://doi.org/10.1109/ICGPR.2018.8441658.

Jol, H. M., 2008, Ground penetrating radar: Theory and applications: Elsevier.

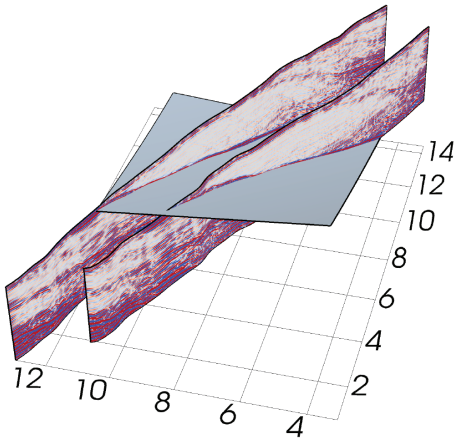

Figure 6. Profile lines with surface interpolated between points picked along reflectors visible in the profiles. Axes are in meters.

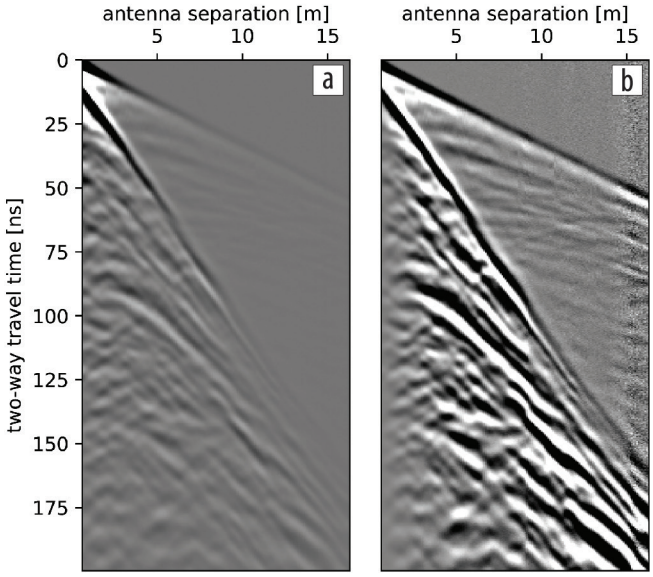

Figure 7. WARR data. (a) Raw data. (b) After dewow and trace normalization.

Klotzsche, A., F. Jonard, M. C. Looms, J. van der Kruk, and J. A. Huisman, 2018, Measuring soil water content with ground penetrating radar: A decade of progress: Vadose Zone Journal, **17**, no. 1, https://doi.org/10.2136/vzj2018.03.0052.

Klysz, G., and J.-P. Balayssac, 2007, Determination of volumetric water content of concrete using ground-penetrating radar: Cement and Concrete Research, **37**, no. 8, 1164–1171, https://doi.org/10.1016/j.cemconres.2007.04.010.

Lai, W. W.-L., X. Dérobert, and P. Annan, 2018, A review of ground penetrating radar application in civil engineering: A 30-year journey from locating and testing to imaging and diagnosis: NDT & E International, **96**, 58–78, https://doi.org/10.1016/j.ndteint.2017.04.002.

Liu, C., and A. Plattner, 2018, Near surface geophysical imaging of the internal structure of El Capitan Meadow Rock Avalanche in Yosemite National Park, California: Presented at Fall Meeting, AGU.

Margrave, G. F., and M. P. Lamoureux, 2019, Numerical methods of exploration seismology: With algorithms in MATLAB: Cambridge University Press, https://doi.org/10.1017/9781316756041.

Merz, K., A. G. Green, T. Buchli, S. M. Springman, and H. Maurer, 2015, A new 3-D thin-skinned rock glacier model based on helicopter GPR results from the Swiss Alps: Geophysical Research Letters, **42**, no. 11, 4464–4472, https://doi.org/10.1002/2015GL063951.

Paz, C., F. J. Alcalá, J. M. Carvalho, and L. Ribeiro, 2017, Current uses of ground penetrating radar in groundwater-dependent ecosystems research: The Science of the Total Environment, **595**, 868–885, https://doi.org/10.1016/j.scitotenv.2017.03.210.

Pitman, S. J., H. M. Jol, J. Shulmeister, and D. E. Hart, 2019, Storm response of a mixed sand gravel beach ridge plain under falling relative sea levels: A stratigraphic investigation using ground penetrating radar: Earth Surface Processes and Landforms, **44**, no. 8, 1610–1617, https://doi.org/10.1002/esp.4598.

Ramachandran, P., and G. Varoquaux, 2011, Mayavi: 3D visualization of scientific data: Computing in Science & Engineering, **13**, no. 2, 40–51, https://doi.org/10.1109/MCSE.2011.35.

Rubin, Y., and S. S. Hubbard, 2005, Hydrogeophysics: Springer.

Stolt, R. H., 1978, Migration by Fourier transform: Geophysics, **43**, no. 1, 23–48, https://doi.org/10.1190/1.1440826.

Tzanis, A., 2006, MATGPR: A freeware MATLAB package for the analysis of common-offset GPR data: Geophysical Research Abstracts, **8**, EGU06–A-09488.

Urban, T. M., J. F. Leon, S. W. Manning, and K. D. Fisher, 2014, High resolution GPR mapping of Late Bronze Age architecture at Kalavasos-Ayios Dhimitrios, Cyprus: Journal of Applied Geophysics, **107**, 129–136, https://doi.org/10.1016/j.jappgeo.2014.05.020.

Wilson, N., 2017, Irlib version v0.4.1, https://doi.org/10.5281/zenodo.439723.

Wilson, N. J., G. E. Flowers, and L. Mingo, 2013, Comparison of thermal structure and evolution between neighboring subarctic glaciers: Journal of Geophysical Research, **118**, no. 3, 1443–1459, https://doi.org/10.1002/jgrf.20096.

Wilson, V., C. Power, A. Giannopoulos, J. Gerhard, and G. Grant, 2009, DNAPL mapping by ground penetrating radar examined via numerical simulation: Journal of Applied Geophysics, **69**, no. 3-4, 140–149, https://doi.org/10.1016/j.jappgeo.2009.08.006.

Yilmaz, Ö., 2001, Seismic data analysis: Processing, inversion, and interpretation of seismic data: Society of Exploration Geophysicists, https://doi.org/10.1190/1.9781560801580.