PYTHON IMAGE PROCESSING USING GDAL FOR PLANETARY RESEARCH

T. M. Hare [a, *], Jay Laura [a], M. Milazzo [a]

[a]Astrogeology Science Center, U.S. Geological Survey, 2255 N. Gemini Dr., Flagstaff AZ 86001 (thare@usgs.gov)

Commission IV, WG IV/8

KEY WORDS: GDAL, Python, extraterrestrial, planetary, image processing, SciPy, Numpy, Matplotlib

ABSTRACT:

Geospatial Data Abstraction Library (GDAL) is a translator library for geospatial raster formats and provides a single abstract data model to read and write raster data. It is community maintained and released under an X/MIT style Open Source license. While written in C/C++, it has bindings for use within JAVA, PERL, Python, .NET and others. GDAL comes with an assortment of stand-alone utility applications for conversion, projection transformations, scaling, stretching, etc. but the real power comes from the application programming interface (API) which allows many other applications and simple image viewers to use the library internally. While GDAL already has basic support for several planetary formats including Planetary Data System (PDS), Integrated Software for Imagers and Spectrometers (ISIS versions 2 and 3) and Flexible Image Transport System (FITS) formats, there are many improvements that should be made.

The primary scripting language the GDAL team has adopted is Python. Python's ease of use, rapid prototyping capabilities, and the ability to call C/C++ routines often mitigate its potential weaknesses. Python was built to be extensible and thus has a huge standard library and a large growing base of science libraries (e.g. NumPy, SciPy, Pandas, Matplotlib). At the Astrogeology Science Center, many of our development needs are being prototyped in Python, and some production software is now being created in Python. For example, to support the NASA InSight and Mars2020 missions, specialized topographic slope software was being supported in an outdated ISIS2 code base. Using GDAL/Python and existing array filtering functions in SciPy, we were able to quickly port the original source code to Python and it is now part of our digital terrain model workflow. During the port, we were also able to easily incorporate histogram binning (NumPy), to combine histogram and cumulative slope graphs (Matplotlib), and create colorized slope figures (GDAL).

* Corresponding author