ENABLING MULTI-MISSION/MULTI-INSTRUMENT SEARCH: THE PDS RING-MOON SYSTEMS NODE SOFTWARE ECOSYSTEM.

Robert S. French[1], Mark R. Showalter[1], Joseph N. Spitale[1], Yu-Jen Chang[1], Debra J. Stopp[1], Matthew S. Tiscareno[1], Mitchell K. Gordon[1], Mia J. T. Mace[1], and Emilie R. Simpson[1]. [1]SETI Institute, 339 Bernardo Ave, Suite 200, Mountain View, CA 94043, rfrench@seti.org.

Introduction: Planetary remote sensing data is generated by a variety of types of instruments (*e.g.*, framing cameras, spectrometers, and charged particle detectors) mounted on a variety of platforms (*e.g.*, Earth-based telescopes, orbiting telescopes, and interplanetary spacecraft) and produced and archived by a variety of teams over years or decades. This diversity is both a great strength and a great challenge. Its strength lies in the sheer volume and diverse types of data available for research. Its challenge lies in the difficulty of finding the information you want – usually a small data needle in an extremely large haystack.

We firmly believe that the key to effective search is rich, uniform metadata. Unfortunately, the metadata received as part of data deliveries to NASA's Planetary Data System (PDS) are often insufficient for this purpose. Each data set may include specific information deemed useful by the generating team – for example, the primary target in the field of view, the right ascension and declination boundaries of the image, or the name of the instrument-specific filter used. But what if the user wants to investigate a body that is in the image but not the primary target? Or wants to search based on body latitude and longitude or resolution? Or wants to search across missions and instruments by wavelength without being forced to learn the details of all the filters on each instrument?

This is exactly the problem we try to solve with Outer Planets Unified Search (OPUS), the flagship search engine of the RMS Node. OPUS provides multi-mission, multi-instrument search using widely applicable metadata, such as the surface geometry of every body in the field of view, the ring geometry of the visible ring system, or the wavelengths at which the observation was made. This metadata is produced at the RMS Node using a complex set of software that has been under development for nearly 15 years. This software, primarily written in Python, totals more than 200,000 lines of code. It is open source and available for public use.

Design considerations – maintainability and ease of use: Our software has been developed and used by many team members over many years. As such it needs to be written in a modern, accepted language using best practices with thorough testing and good documentation. Our software is written primarily in Python 3.8+ and we utilize nightly automatic test suites including code coverage.

Design considerations - speed: OPUS currently supports over 1.6 million observations. Each of these needs to be analyzed at the pixel level to produce more than 100 types of metadata, ultimately resulting in potentially tens or hundreds of billions of complex computations to compute all of the metadata from scratch.

Packages: Following is a list of the primary packages we have developed.

- **cspyce** is a Python interface to the NAIF CSPICE library that is focused on high performance. The underlying interface is written in C, which offers a significant speedup over Python-only implementations. We also provide a vectorized interface to nearly all CSPICE functions so that entire NumPy arrays can be processed at the C level without requiring many separate Python calls. We see speedups ranging from 50% to more than 10X compared to Python-only interfaces.

For example, if `et_array` is a 1000×1000 NumPy array of times, then

```
cspyce.spkezr.array("MARS",
    et_array, "J2000", "LT+S", "EARTH")
```

will return a 1000×1000×6 array containing the state vectors of Mars relative to Earth for those one million times.

Additional features include the use of Python exceptions to replace CSPICE error flags and support for body and frame aliases that handle SPICE IDs that have changed over time. cspyce is available for most platforms with `pip install cspyce`.

- **polymath** is a Python module that solves a fundamental problem with NumPy: how to perform operations on multi-dimensional data sets where some dimensions simply define the observation space (*e.g.*, pixels in an image) and other dimensions define the data to be operated on (*e.g.*, six-element state vectors). polymath defines new Python classes such as Scalar, Vector, Matrix, and Quaternion that perform parallel operations at the array level, where each element is of the given data type. All data types include efficient use of invalid data masks and full support for numerical derivatives. Other support includes intuitive array indexing, three-valued logic, and the saving and loading of state.

For example, the following code creates an array of 360 three-element (X/Y/Z) vectors that form a circle of radius 5 at evenly spaced longitudes. It then converts the vectors to be unit length, and then derives the portion of each vector that is perpendicular to the vector (0,1,1).

```
vec = Vector3.from_cylindrical(5.,
   numpy.radians(numpy.arange(360.)))
unit_vec = vec.unit()
perp_vec = unit_vec.perp([0,1,1])
```

• **oops** (Object-Oriented Python and SPICE) is a Python module that utilizes cspyce and polymath to provide sophisticated modeling and geometric queries. Concepts supported include:

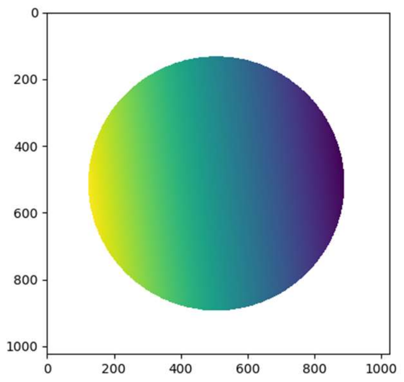
- *instrument types*: single pixels (such as occultations), framing cameras, push broom cameras, raster scans, raster slits, *etc*.;
- *fields of view*: flat, barrel and radial distortion, polynomial distortion, offset, subsampled, *etc*.;
- *frames of reference*: non-rotating, rotating, inclined, synchronous, *etc*.;
- *spatial paths*: circular, Keplerian, taken from SPICE kernels, *etc*.;
- *surfaces*: planetocentric spheroid and ellipsoid, planetographic spheroid and ellipsoid, limb, orbit plane, ring plane, *etc*.;
- *backplanes*: hundreds of metadata backplanes, such as emission, incidence, and phase angle, surface resolution, and distance to surface can be generated with a single function call.
- *instrument*: full support for popular missions and instruments, including Cassini ISS, UVIS, and VIMS; Galileo SSI; HST ACS, WFC3, WFPC2, and NICMOS; Juno JIRAM and JunoCam; JWST NIRCAM; New Horizons LORRI; and Voyager ISS. Additional instrument support can be added easily, because each instrument is defined using the same core building blocks described above.
- *SPICE kernel management*: Automatic management and furnishing of appropriate SPICE kernels.

For example, the code in the next column is all that is required to read in a Cassini ISS image, produce a 1024×1024 backplane of phase angles on the surface of Dione, and compute the distance between the observer (Cassini) and Dione.

Various simple scripts use oops to analyze all of the observations in a PDS volume/bundle to create text-based metadata files containing surface and ring geometry. These files are available for public download[1], and are also used to feed OPUS.

[1] https://pds-rings.seti.org/viewmaster/metadata

```
import oops
import hosts.cassini.iss as iss
obs = iss.from_file('N1501619321_1.IMG')
bp = oops.backplane.Backplane(obs)
pa = bp.phase_angle('DIONE')
dist = bp.center_distance('DIONE')
```



oops is also heavily utilized to facilitate the research of several RMS Node scientists, and we believe that it can dramatically reduce the effort required by many other researchers who process remote sensing data.

• **OPUS** is a web-based search engine that allows the specification of hundreds of parameters, browsing of results, storing desired observations in a shopping cart, and product-level downloading. It consists of an import pipeline that populates a MySQL database, a Python-based back end that supports a database query API, and a JavaScript-based front end that provides the end user experience. Although currently customized for the missions supported by the RMS Node, OPUS is fundamentally a generic application that could be applied to a variety of domains.

• **Minor utilities**. In addition to our major projects, we have a number of smaller modules available for targeted tasks. These include: **julian** for manipulating dates; **vicar** for reading and writing VICAR files; **gravity** for providing information about the gravity fields of oblate planets; **pdsparser** and **pdstable** for reading PDS3 labels and data tables; and **picmaker** for creating browse images from a wide variety of instrument-specific data formats.

Status and future plans: Our software has been used extensively within RMS Node projects, but has had limited exposure to outside users. Most are not yet available as easily-installable packages or appropriately documented for novice users. Users are welcome to use the software directly from our GitHub repos[2], but with limited support. We hope over time to make these packages more amenable to public use.

[2] https://github.com/SETI/pds-oops and pds-tools