THE COMMUNITY SENSOR MODEL STANDARD UPDATE. T. M. Hare, J. R. Laura, J. Mapel, K. L. Berry, K. Rodriguez, A. C. Paquette, U.S. Geological Survey, Astrogeology, Flagstaff, AZ, 86001 (thare@usgs.gov).

Introduction: Camera models are a key component of any digital photogrammetric system and are used to accurately project remotely sensed information (e.g., images) to the surface of a planetary body. The U.S. Geological Survey's Astrogeology Science Center (ASC) has begun to develop and test camera models using the Community Sensor Model (CSM) standard [1, 2] to support interoperability and broad use of planetary sensor definition across a range of custom and off-the-shelf software tools, as well as a RESTful SPICE web service to remove the need to download, manage, and version spacecraft positional and pointing information. Herein, we present continuing work ASC is undertaking to provide the CSM standard and associated SPICE information.

Background: A sensor model is a mathematical description of the relationship between the three-dimensional object (e.g. a target's surface) and the associated two-dimensional image plane. As described in [3], the quantities needed to define a sensor model can be divided in two broad categories: interior orientation and exterior orientation (or intrinsic and extrinsic matrices from the computer vision literature). The interior parameters are intrinsic to the sensor design and calibration and typically include focal length, location of the principal point, and lens distortions. For more complicated instruments, the interior parameters may also include wavelength dependencies, gain and pixel summing settings, and (for pushbroom sensors) the timing of line exposures and time delay integration (TDI) settings. The exterior parameters describe the location and orientation of the sensor with respect to the target's reference coordinate system. For planetary applications, this information is typically stored in the form of SPICE (Spacecraft, Planetary ephemeris, Instrument, C-Matrix, and Event) kernels and delivered by the Navigation and Ancillary Information Facility (NAIF, [4]).

The CSM Standard: The Community Sensor Model (CSM) Working Group was established by the U.S. defense and intelligence community with the goal of standardizing camera models for various remote sensor types [5]. The CSM standard, now at version 3.0.3, is a framework that provides a well-defined application program interface (API) for multiple types of sensors and has been widely adopted by remote sensing software systems. One of the changes from version 3.0.2 to 3.0.3 was the addition of variable target radii, which enables planetary support (Figure 1). Previously, only an Earth radius (WGS84) was available within the standard.

It is worth noting that the CSM defines a standard interface and does not make the creation of a camera sensor model technically any easier as the implementation details are left to the developer. By defining a standard interface, the CSM supports interoperability between different photogrammetric applications making the development and maintenance of multiple sensor models for similar instruments unnecessary. The CSM API has been designed and continuously tested by terrestrial industry experts and expanded to support necessary planetary parameters. Therefore, we assert that the planetary domain will benefit significantly from the adoption of the CSM standard that has more than a decade of design history and development by the CSM Working Group.

Last year our ASC programming team implemented the MESSENGER Mercury Dual Imaging System (MDIS) CSM framing camera model for both the narrow-angle and wide-angle cameras (NAC and WAC respectively) [2, 6]. In mid-2018, we open sourced and released a pushbroom CSM co-developed by BAE and ASC. After some initial setbacks, this CSM is **successfully** being tested in BAE's SOCET GXP for deriving digital elevation models from MRO's CTX and we have plans to begin testing HiRISE cameras [7] and LROC NAC cameras [8]. This CSM has also been recently updated to handle variable line-rate pushbrooms like HRSC [9].

CSM Testing Environment: CSM code is written in C++ for both performance and use of legacy code. To facilitate broad use in exploratory environments, improve testing, and support rapid prototyping, we have recently re-wrapped the CSM using the Simplified Wrapper and Interface Generator (SWIG, http://www.swig.org/) from the original CPython library. By changing the wrapping method to SWIG, we can not only provide Python bindings but also JAVA, C, and other SWIG-supported languages. Currently, we still focus on Python bindings as we assert that planetary science community adoption of python continues to increase rapidly [11]. The effort applied to wrapping the CSM has four benefits. First, all tests are written in Python using the PyTest framework that provides support for high level object mocking and easy integration into continuous integration environments. Second, Python bindings all use of the CSM within the near ubiquitous Jupyter notebook [10] environment. The Jupyter Notebook is a backend python kernel and web server with a browser-based frontend that allows users to share code, equations, visualizations, and any associated documentation (http://jupyter.org/). Third, using Python provides access to SPICE (via SpiceyPy; https://github.com/AndrewAnnex/SpiceyPy) and support for planetary images using the Geospatial Data Access Library (GDAL; http://www.gdal.org). Finally, Jupyter notebooks were heavily used to prototype, in

Python, the algorithms for the C++ implementation of the CSM standard.

CSM Availability: The ASC CSM implementation is available throughout the development process under an open source, public domain license that support maximum reuse by the community. The underlying CSM library (in C++) is built in a continuous integration environment and available via the anaconda (conda) package manager. The ASC maintains a public facing build of the CSM. The SWIG wrapper is available via the ASC GitHub website (https://github.com/USGS-Astrogeology/CSM-Swig) and via a conda installable package for linux-64, Macintosh OSX, and Windows. Finally, the MDIS-NAC and MDIS-WAC implementations are available via both distribution mechanisms (http://bit.ly/CSM-CameraModel). We intentionally separate the underlying library (C++), the python wrapper (SWIG) and our implemented camera models (C++/SWIG) to support modularity and standard separation of concerns. Usage examples, as we envision a developer or end-user performing exploratory analysis, are available as Jupyter notebooks in our CSM-SET (Sensor Exploitation Tool) repository (http://bit.ly/CSMSET_Jupyter). Currently the MDIS and CTX CSM implementations are highlighted, but we expect pushbroom CSM examples to be available soon.

The last step to realize our CSM Python testing environment is to more seamlessly provide access to the NAIF supplied SPICE via a RESTful web service. The RESTful frontend for this project is called the ASC Pfeffernusse project (https://github.com/USGS-Astrogeology/pfeffernusse). The backend which handles the different missions and SPICE is called the ASC Abstraction Layer for Ephemerides (ALE, https://github.com/USGS-Astrogeology/ale). These projects currently depend on the Python libraries SpiceyPy, flask, and numpy. REST is acronym for REpresentational State Transfer and RESTful web services can be thought of as micro-transactions to access simple http addresses. The goal for this implementation is that given an image or stereo-pair, allow the user to simply request the ISD (Image Support Data) which contains the positional description for each image as required by the CSM. With the ISD in hand, the CSM can be fully tested within an interactive Jupyter notebook environment.

Conclusion: Prototype development and subsequent adoption of the CSM standard is the first step in realizing highly interoperable sensor models that can be used and shared across NASA's and international planetary missions. Using the CSM and SPICE web service within an interactive Jupyter Notebook, we find an ideal exploratory environment for sensor model development, data analysis, validation, portability, and finally the capability of demonstrating results to collaborators.

Acknowledgments: This effort has been supported by NASA's Planetary Spatial Data Infrastructure (PSDI) interagency agreement.

References: [1] Hare T. M. and Kirk R.L., 2017, LPSC XLVIII, abs #1111. [2] Hare T. M., et al., 2017, 3rd Planetary Data Workshop, abs #7130. [3] National Geospatial-Intelligence Agency, (2011), Frame Sensor Model Metadata Profile Supporting Precise Geopositioning, NGA.SIG.0002_2.1. [4] Acton, C.H. 1996, Planetary and Space Science, Vol. 44, No. 1, 65-70. [5] Community Sensor Model Working Group, (2010), Community Sensor Model Technical Requirements Document, v. 3.0, NGA.STND.0017_3. [6] Hawkins, S.E., Boldt, J.D., Darlington, E.H. et al. Space Sci. Rev., 2007, 131: 247. doi:10.1007/s11214-007-9266-3. [7] Fergason, R.L. et al., 2016, Space Sci. Rev., 1572-9672, doi:10.1007/s11214-016-0292-x. [8] Burns, K. N., et al., 2012, ISPRS, doi:0.5194/isprsarchives-XXXIX-B4-483-2012. [9] Kirk, R. L., et al., 2017, ISPRS, doi:10.5194/isprs-archives-XLII-3-W1-69-2017 [10] Fernando Pérez, Brian E. Granger, IPython: A System for Interactive Scientific Computing, Computer Science. and Eng., vol. 9, no. 3, pp. 21-29, May/June 2007, doi:10.1109/MCSE.2007.53. URL: http://ipython.org. [11] Laura, J. R., et al., 2015, LPSC XLVI, abs #2208.

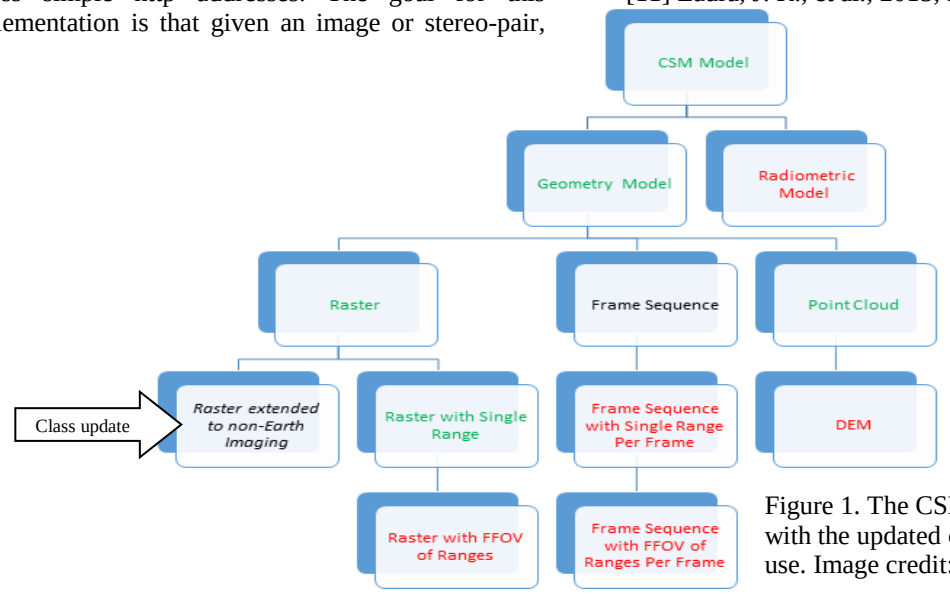

Figure 1. The CSM API hierarchy diagram with the updated class to support planetary use. Image credit: CSM Working Group.