PlanetServer Python API – Visualization and Analysis of CRISM images.

A. Halder [1], R. Marco Figuera[1], A. P. Rossi[1], M. Minin[1], A. Zinzi[2,3], [1]Physics and Earth Sciences, Jacobs University, 28759 Bremen, Germany; a.halder@jacobs-university.de; [2]ASI Science Data Center, c/o ASI, Via del Politecnico snc, 00133, Rome, Italy; [3]INAF-OAR, Via Frascati n. 33, 00078, Monte Porzio Catone (RM), Italy

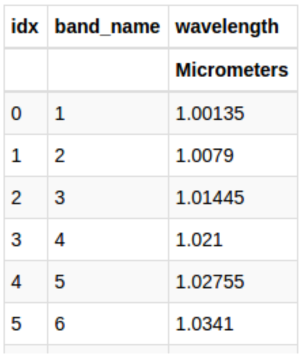
Introduction: Analysis of spectroscopic data collected through remote sensing satellites is a major way of characterizing mineral diversity on Mars. For example, the spectroscopic data collected by the CRISM imaging spectrometer (Mars Reconnaissance Orbiter) [1] can be analysed by using a software called ENvironment for Visualizing Images (ENVI). Although ENVI allows for the processing of such hyperspectral data, it is expensive and also difficult to implement. In order to accomplish the results in a more user friendly and cost-effective way, PlanetServer - an open-source online platform - is being developed that will provide access to visualization and analysis of Mars CRISM data [2].

PlanetServer is equipped with two hyperspectral analysis tools: Spectral analysis of a pixel and band math RGB combinations [2]. Both analyses use the Web Coverage Processing Service (WCPS) OGC standard [3], an SQL-like language that is capable to query information along an image cube and perform RGB combinations. This abstract focuses on the ongoing work in PlanetServer on developing the spectral analysis tool for the band math RGB combinations on CRISM data.

PlanetServer Python API: In order to characterize minerals from CRISM data, [4] developed a combination of CRISM multispectral summary products to parameterize the mineral diversity on Mars. For example, [5] suggested the existence of hydrated silicates in Nili Fossae and so a combination of products (e.g. BD1900H, BD2200, BD2500 and D2300) from [4] were used to characterize the materials. More recently, [6] created revised summary products in order to retrieve information about the different minerals. In the ongoing work in PlanetServer, a Python interface is under development [7] that integrates all the CRISM products mentioned in [6] and performs different RGB band math combinations. The Python API has been constructed as follows:

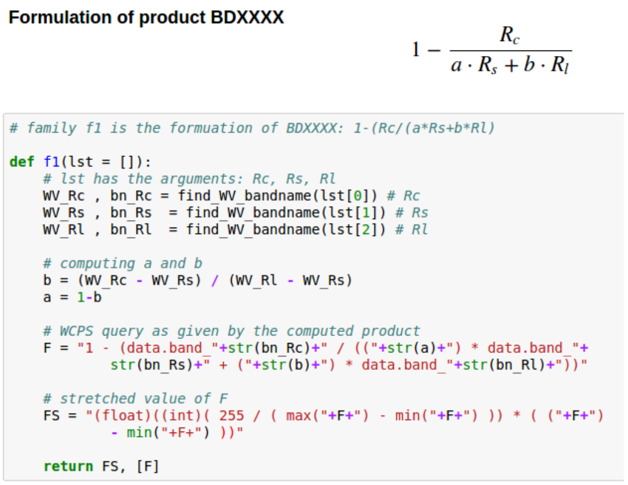
1. **Band name and wavelength lookup table:** An Astropy [8] table (see Figure 1) is first made on Jupyter Notebook by extracting data on the different band names and the corresponding wavelengths from a file called 'metadata.txt' included in [7] containing information about the different band names and wavelengths from CRISM data.

2. **Implementation of CRISM summary products:** Another Jupyter Notebook is made in order to write down the formulation of all the summary products mentioned in [6]. The formulation of a product is coded in a function which takes the product's parameters as input (e.g. the product BD2290 will have its parameters as R2290, R2250, R2350 [6]), finds the corresponding band name and wavelength associated with each parameter using the lookup table and finally computes the formulated output in terms of a WCPS query [3]. An example of the formulation of a product on Jupyter Notebook is shown in Figure 2. Formulations of all the products are written similarly as respective functions which compute and create the corresponding WCPS queries.



idx	band_name	wavelength
		Micrometers
0	1	1.00135
1	2	1.0079
2	3	1.01445
3	4	1.021
4	5	1.02755
5	6	1.0341

Figure 1: Snap of Astropy lookup table for band names and corresponding wavelengths (idx stands for index).



Formulation of product BDXXXX

$$1 - \frac{R_c}{a \cdot R_s + b \cdot R_l}$$

```
# family f1 is the formuation of BDXXXX: 1-(Rc/(a*Rs+b*Rl)

def f1(lst = []):
    # lst has the arguments: Rc, Rs, Rl
    WV_Rc , bn_Rc = find_WV_bandname(lst[0]) # Rc
    WV_Rs , bn_Rs  = find_WV_bandname(lst[1]) # Rs
    WV_Rl , bn_Rl  = find_WV_bandname(lst[2]) # Rl

    # computing a and b
    b = (WV_Rc - WV_Rs) / (WV_Rl - WV_Rs)
    a = 1-b

    # WCPS query as given by the computed product
    F = "1 - (data.band_"+str(bn_Rc)+" / (("+str(a)+") * data.band_"+
            str(bn_Rs)+" + ("+str(b)+") * data.band_"+str(bn_Rl)+"))"

    # stretched value of F
    FS = "(float)((int)( 255 / ( max("+F+") - min("+F+") )) * ( ("+F+")
            - min("+F+") ))"

    return FS, [F]
```

Figure 2: Formulation of the product BDXXXX.

3. **User input and API output:** The user only needs to give four input arguments to the Python API: Coverage Identification (region of interest on Mars) and the name of three CRISM summary products that are stored in each channel (Red, Green and Blue). The three products are executed according to their respective formulations and all three WCPS query outputs are then combined to produce the analysed image url. An example of the functionality of the API can be seen as follows. 3 products from [6] have been chosen - for red: SINDEX2, for green: BD2100_2, for blue: BD1900_2. With a given Coverage ID that is defined by PlanetServer, e.g. ‘frt0000b385_07_if164l_trr3’, the API shows the url of the analysed image - shown in Figure 3.

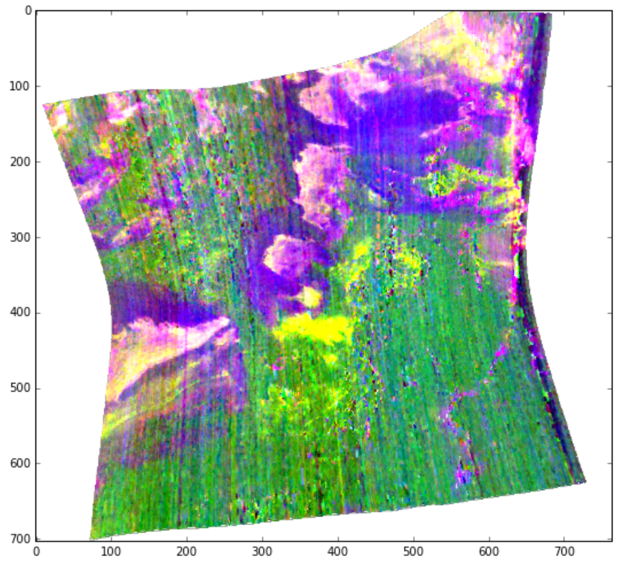


Figure 3: RGB combination using the HYD product [6] (Red:SINDEX2 ; Green:BD2100_2 ; Blue:BD1900_2) showing hydrated minerals found in the image ‘frt0000b385_07_if164l_trr3’. Polyhydrated sulfates are shown in pink, Monohydrated sulfates in green/yellow and blue indicates other hydrated minerals such as clays, carbonate or zeolite.

Ongoing work: Extension of the API to other datasets besides Mars CRISM data is also under development (e.g. for Moon M3 data). The inclusion of spectral analysis of a pixel in the PlanetServer API is also being evaluated. The possibility of the datasets to be queryed by external services is now being used by MATISSE [9]. In particular Mars data will be accessed directly from PlanetServer using initially the ‘getCoveragesIntersectBoundingBox’ function for geographic search and then the ‘ProcessCoverage’ function to obtain a FITS file that can be ingested by the tool. At the end of this process MATISSE can further process observations making higher-order products, such as mosaics or ratios with both online and offline visualization options.

This will not only help to easily access and visualize planetary remote sensed data but also provide an open source platform that can be availed by anybody across the globe without the need for any expensive software. Once the API is published, a set of Jupyter Notebooks will be provided in order to give an overview of its capabilities. An early release of PlanetServer’s Python API is already accessible at [7].

References

[1] S. Murchie et al. Compact Reconnaissance Imaging Spectrometer for Mars (CRISM) on Mars Reconnaissance Orbiter (MRO). *Journal of Geophysical Research: Planets*, 112(E5), 2007.

[2] R. Marco Figuera et al. Online characterization of planetary surfaces: PlanetServer, an open-source analysis and visualization tool. *Planetary and Space Science, submitted*, 2016.

[3] P. Baumann. The OGC Web Coverage Processing Service (WCPS) Standard. *Geoinformatica*, 14(4), 2010.

[4] S. M. Pelkey et al. CRISM multispectral summary products: Parameterizing mineral diversity on Mars from reflectance. *Journal of Geophysical Research: Planets*, 112 (E8), 2007.

[5] B. L. Ehlmann et al. Identification of hydrated silicate minerals on Mars using MRO-CRISM: Geologic context near Nili Fossae and implications for aqueous alteration. *Journal of Geophysical Research: Planets*, 114(E2), 2009.

[6] C. E. Viviano-Beck et al. Revised CRISM spectral parameters and summary products based on the currently detected mineral diversity on Mars. *Journal of Geophysical Research: Planets*, 119(6), 2014.

[7] A. Halder et al. PlanetServer Python API. 2016. URL https://github.com/planetserver/PS_Python_API.

[8] Astropy Collaboration. Astropy: A community Python package for astronomy. , 558:A33, 2013.

[9] A. Zinzi et al. Matisse: A novel tool to access, visualize and analyse data from planetary exploration missions. *Astronomy and Computing*, 15, 2016.