CRISM Spectral Calculator Documentation

Release 0.1

Jay Laura

February 11, 2013

CONTENTS

Welcome to the CRISM Spectral calculator. This command line tool implements 35+ algorithms for the analysis of CRISM MRDR and potentially TRDR (TRR3) data products. Currently, only the surface algorithms have been implemented.

CRISM Spectral Analysis Suite (CSAS) is released into the public domain without warranty. Algorithms implementations are currently untested and should be verified against existing, published, results to confirm output.

CHAPTER

ONE

NEWS

1.11.13:

1. Initial Alpha Release

1.1 Installing CSAS

1.1.1 CSAS Dependencies

CSAS requires Python and depends on several other freely available Python modules. Prior to installing CSAS, you should make sure its dependencies are met.

Table 1.1: CSAS Dependencies

Dependency	Requirement
Python 2.5+	Required
NumPy	Required
GDAL	Required

Note that CSAS does not work with Python 3.x.

1.1.2 Installing

CSAS is distributed as a stand alone script and therefore does not require installation. This documentation is distributed locally, with CSAS. Simply place the script in a convenient directory.

Should you wish to access CSAS from any directory on your machine, it is necessary to append your PYTHONPATH with the path to CSAS. To do this, first not the absolute path to the directory where CSAS is installed. For example: '/Users/Jay/CSAS'. Then run the following from a command line (run cmd on a Windows machine / terminal on an OS X machine).

OS X

To allow CSAS to be called from any directory via the command line utilize the following. To have these changes be persistent append them to ~/.profile or ~/.bashrc.:

```
$ PYTHONPATH="/Users/Jay/CSAS:$PYTHONPATH"
$ export PYTHONPATH

where, /Users/Jay/CSAS is replaced with the path to your CSAS installation
```

Windows

To allow CSAS to be called from any directory via the command line utilize the following. To have these changes be persistent append them to autoexec.bat.:

```
$ set PYTHONPATH=%PYTHONPATH%;C:\CSASInstallDirectory
```

You can also append autoexec.bat graphically. Run *msconfig* from the start menu.

1.2 Algorithms

This page documents all implemented algorithms in two ways. First, we show the mathematical structure of the algorithm as documented in Pelkey (2007). Second, we show a usage example that can be copied to a command line.

Algorithms are broadly split into two categories as per Pelkey et. al (2007)[1]. We have implemented the surface algorithms for MRDR products. These are map projected, mosaiced TRR3 images that have band to wavelength mapping in the header. We have also supplied the SW lookup tables and code to allow for band to wavelength mappings for TRR3 files. Currently, TRR3 files are not supported due to issues reading the label.

In the algorithm formulations below we indicate the reflectance of a band using R and the wavelength of the band using wv. Note that wv is the nearest wavelength, not the absolute wavelength. This is because the lookup tables can undergo modification such that hard coded values would not be possible.

Warning: The detector used to capture an image dictates which wavelengths the image contains. Since all TRR3 images ship with an identical number of bands, we check the detector label (from the header). Note that algorithms not designed to run on IR data will still process a VNIR image, but the results will not be as expected.

Warning: The VNIR detector (S) supports, Reflectance 770nm, Red / Blue Ratio, Band Depth 530nm, Shoulder Height 600nm, Band Depth 640nm, Band Depth 860nm, Band Depth 920nm, RPeak1, Integrated Band Depth 1nm Visible, Reflectance 440nm, and IR Ratio 800nm / 1020nm.

Warning: The IR detector (L) supports, Integrated Band Depth 1000nm IR, 1.3µm reflectance, Olivine Index, LCP Index, HCP Index, Spectral Variance, ISlope1, Band Depth 1435nm, Band Depth 1500nm, Ice 1, Ice 2, Band Depth 1750, Band Depth 1900, Integrated Band Depth 2µm, Band Depth 2100nm, Band Deoth 2210nm, Band Depth 2290nm, 2.3µm Drop, 2.33µm & 2.53µm Band Depth, Band Depth 3000nm, Band Depth 3100nm, Band Depth 3200nm, Band Depth 3400nm, Band Depth 1270µmO2, 3.9µm Gauge, Band Depth 1400nm H20,Band Depth 2000nm CO2, Band Depth 2350, Band Depth 2600, IR Ratio 2, Reflectance 2700nm, Band Depth 2700, IR Ratio 3.

Note: Do to compatibility issues between the GDAL PDS driver and the TRR3 data product, we rewrite a label, appended with '_fixed.lbl' for each input image. Additionally, we derive an ENVI .hdr file. These workarounds allow GDAL to support the TRR3 data files without changes to the label files supplied by the team.

Warning: If an output with the same name already exists, the new output overwrites the old output. That is, if you run the script twice without supplying a file name, the new output will overwrite your old output.

1.2.1 Surface

Reflectance at 770µm

Rationale : rock / dust

Algorithm : $R770$

Code Snippet:

```
$ python csas.py --R770 input.IMG output.tif
```

Red / Blue Ratio

Rationale : rock / dust

Algorithm : $\frac{R770}{R440}$

Code Snippet:

```
$ python csas.py --RBR input.IMG output.tif
```

Band Depth at 530nm

Rationale : Crystalline ferric minerals

Algorithm :

$$a = \frac{wv530 - wv440}{wv709 - wv440}$$
$$b = 1.0 - a$$
$$bd530 = \frac{R530}{(a * R709) + (b * R440)}$$

Code Snippet:

```
$ python csas.py --BD530 input.IMG output.tif
```

Shoulder 600nm

Rationale : Select ferric minerals

Algorithm :

$$a = \frac{wv600 - wv533}{wv710 - wv533}$$
$$b = 1.0 - a$$
$$bd600 = \frac{R600}{(a * R530) + (b * R709)}$$

Code Snippet:

```
$ python csas.py --SH600 input.IMG output.tif
```

Band Depth 640nm

Rationale : Select ferric minerals, especially maghemite

Algorithm :

$$a = \frac{wv648 - wv600}{wv709 - wv600}$$
$$b = 1.0 - a$$
$$bd640 = \frac{R600}{(b * R600) + (a * R709)}$$

Code Snippet:

```
$ python csas.py --BD640 input.IMG output.tif
```

Band Depth 860

Rationale : Select ferric minerals ('hematite band')

Algorithm :

$$a = \frac{wv860 - wv800}{wv984 - wv800}$$
$$b = 1.0 - a$$
$$bd860 = \frac{R860}{(a * R800) + (b * R984)}$$

Code Snippet:

```
$ python csas.py --BD860 input.IMG output.tif
```

Band Depth 920

Rationale : Select ferric minerals ('Pseudo BDI1000 VIS')

Algorithm :

$$a = \frac{wv920 - wv800}{wv984 - wv800}$$
$$b = 1.0 - a$$
$$bd920 = \frac{R920}{(b * R800) + (a * R984)}$$

Code Snippet:

```
$ python csas.py --BD920 input.IMG output.tif
```

RPeak 1

Rationale : Fe mineralogy

Algorithm : Wavelength where 1st derivative=0 of 5th order polynomial fit to R600, R648, R680, R710, R740, R770, R800, R830

Code Snippet:

```
$ python csas.py --RPEAK1 input.IMG output.tif
```

Warning: This is computationally expensive, and a slow implementation. Expect runtimes between 15 and 30 minutes for a 1GB image.

Integrated Band Depth 1µm Visible

Rationale : Crystalline Fe+2 or Fe+3 minerals

Algorithm [Divide R830, R860, R890, R915 by RPEAK1 then] integrate over (1 - normalized radiances)

Code Snippet:

```
$ python csas.py ----BDI1000VIS input.IMG output.tif
```

Warning: This algorithm must first run RPeak1 (15 - 30 minutes) and then integrate again over 4 bands. This is computationally expensive and will require 35+ minutes.

Integrated Band Depth 1000 IR

Rationale : Crystalline Fe+2 minerals; corrected for overlying aerosol induced slope

Algorithm : Divide R1030, R1050, R1080, R1150 by linear fit from peak R between 1.3 - 1.87 microns to R2530 extrapolated backwards, then integrate over (1 - normalized radiances)

Code Snippet:

```
$ python csas.py ----BDI1000IR input.IMG output.tif
```

Warning: Not yet implemented.

1.3µm Reflectance (IR Albedo)

Rationale : IR Albedo

Algorithm : $R1330$

Code Snippet:

```
$ python csas.py --IRAlbedo input.IMG output.tif
```

Olivine Index

Rationale : Olivine will be strongly +; based on fayalite

Algorithm : $\frac{R1695}{(0.1*R1080)+(0.1*R1210)+(0.4*R1330)+(0.4R1470)} - 1$

Code Snippet:

```
$ python csas.py --OlIndex input.IMG output.tif
```

Olivine Index 2

Rationale : Olivine Index for TRDR v3

Algorithm : Unknown

Warning: Not yet implemented by the CAT team.

HCP Index

Rationale : Pyroxene is strongly +; favors high-Ca pyroxene

Algorithm : $100 * (\frac{R1470-R1080}{R1470+R1080} / \frac{R1470-R2067}{R1470+R2067})$

Code Snippet:

```
$ python csas.py --HCP input.IMG output.tif
```

Note: Algorithm differs from published - coded as per CAT

LCP Index

Rationale : Pyroxene is strongly +; favors low-Ca pyroxene

Algorithm : $100 * (\frac{R1330-R1080}{R1330+R1080} / \frac{R1330-R1815}{R1330+R1815})$

Code Snippet:

```
$ python csas.py --LCP input.IMG output.tif
```

Note: Algorithm differs from published - coded as per CAT

Spectral Variance

Rationale : Ol & Px will have high values; Type 2 areas will have low values

Algorithm : Variance of observed data from a line fit from 1.0 - 2.3µm

Code Snippet:

```
$ python csas.py --VAR input.IMG output.tif
```

Warning: Not yet implemented.

ISlope1

Rationale : Ferric coating on dark rock

Algorithm : $\frac{R1815-R2530}{wv2530-wv1815}$

Code Snippet:

```
$ python csas.py --ISLOPE1 input.IMG output.tif
```

Band Depth 1435nm

Rationale : CO2 surface ice

Algorithm :

$$a = \frac{wv1430 - wv1370}{wv1470 - wv1370}$$
$$b = 1.0 - a$$
$$bd1435 = \frac{R1470}{(b * R1370) + (a * R1470)}$$

Code Snippet:

```
$ python csas.py --BD1435 input.IMG output.tif
```

Band Depth 1500mn

Rationale : H2O surface ice

Algorithm : $1.0 - (\frac{R1558+R1505}{R1808+R1367})$

Code Snippet:

```
$ python csas.py --BD1500 input.IMG output.tif
```

Note: Algorithm differs from published - coded as per CAT (reduced instrument noise)

ICER1

Rationale : CO2, H20 ice mixtures

Algorithm : $\frac{R1430}{R1510}$

Code Snippet:

```
$ python csas.py --ICER1 input.IMG output.tif
```

Band Depth 1750

Rationale : Gypsum

Algorithm :

$$a = \frac{wv1750 - wv1557}{wv1815 - wv1557}$$
$$b = 1.0 - a$$
$$bd1750 = \frac{R1750}{(b * R1557) + (a * R1815)}$$

Code Snippet:

```
$ python csas.py --BD1750 input.IMG output.tif
```

Band Depth 1900

Rationale : H2O, chemically bound or adsorbed

Algorithm : $1.0 - (\frac{R1972+R1927}{R2006+R1874})$

Code Snippet:

```
$ python csas.py --BD1900 input.IMG output.tif
```

Note: Algorithm differs from published - coded as per CAT (reduced instrument noise)

Integrated Band Depth 2µm

Rationale : Pyroxene abundance and particle size

Algorithm : divide R1660, R1815, R2140, R2210, R2250, R2290, R2330, R2350, R2390, R2430, R2460 by linear fit from peak R between 1.3 - 1.87 microns to R2530, then integrate over (1 - normalized radiances)

Code Snippet:

```
$ python csas.py --BDI2000 input.IMG output.tif
```

Warning: Not yet implemented.

Band Depth 2100

Rationale : Monohydrated minerals

Algorithm :

$$a = \frac{((wv2120 - wv2140)/2) - wv1930}{wv2250 - wv1930}$$
$$b = 1.0 - a$$
$$bd2100 = 1.0 - (\frac{(R2120 + R2140) * 0.5}{(b * R1930) + (a * R2250)})$$

Code Snippet:

```
$ python csas.py --BD2100 input.IMG output.tif
```

Band Depth 2210

Rationale : Al-OH minerals: monohydrated minerals

Algorithm :

$$a = \frac{wv2210 - wv2140}{wv2250 - wv2140}$$
$$b = 1.0 - a$$
$$bd2210 = 1.0 - (\frac{(R2210}{(b * R2140) + (a * R2250)})$$

Code Snippet:

```
$ python csas.py --BD2210 input.IMG output.tif
```

Band Depth 2290

Rationale : Mg,Fe-OH minerals (at 2.3); also CO2 ice (at 2.292 microns)

Algorithm :

$$a = \frac{wv2290 - wv2250}{wv2350 - wv2250}$$
$$b = 1.0 - a$$
$$bd2290 = 1.0 - (\frac{(R2290}{(b * R2250) + (a * R2350)})$$

Code Snippet:

```
$ python csas.py --BD2290 input.IMG output.tif
```

2.3µm Drop

Rationale : Hydrated minerals; particularly clays

Algorithm :

$$slope = \frac{band2530 - band1815}{wv2530 - wv1815}$$
$$cr2290 = R1815 + slope * (wv2290 - wv1815)$$
$$cr2320 = R1815 + slope * (wv2320 - wv1815)$$
$$cr2330 = R1815 + slope * (wv2330 - wv1815)$$
$$cr2120 = R1815 + slope * (wv2120 - wv1815)$$
$$cr2170 = R1815 + slope * (wv2170 - wv1815)$$
$$cr2210 = R1815 + slope * (wv2210 - wv1815)$$
$$drop = 1.0 - (\frac{\frac{R2290}{cr2290} + \frac{R2320}{cr2330} + \frac{R2330}{cr2330}}{\frac{R2120}{cr2120} + \frac{R2170}{cr2170} + \frac{R2210}{cr2210}})$$

Code Snippet:

```
$ python csas.py --D2300 input.IMG output.tif
```

SIndex

Rationale : Hydrated minerals; particularly sulfates

Algorithm : $1.0 - (\frac{R2100+R2400}{2*R2290})$

Code Snippet:

```
$ python csas.py --SINDEX input.IMG output.tif
```

ICER2

Rationale : CO2 ice will be >>1, H2O ice and soil will be about 1

Algorithm : $\frac{R2530}{R2600}$

Code Snippet:

```
$ python csas.py --ICER2 input.IMG output.tif
```

BDCarb

Rationale : Carbonate overtones

Algorithm :

$$a = \frac{(0.5(wv2330 + wv2120)) - wv2230}{wv2390 - wv2230}$$
$$b = 1.0 - a$$
$$c = \frac{(0.5(wv2530 + wv2120)) - wv2390}{wv2600 - wv2390}$$
$$d = 1.0 - c$$
$$BDCarb = 1.0 - (\sqrt{\frac{R2330}{(b * R2230) + (a * R2390)} * \frac{R2530}{(d * R2230) + (c * R2600)}})$$

Code Snippet:

```
$ python csas.py --BDCARB input.IMG output.tif
```

BD3000

Rationale : H2O, chemically bound or adsorbed

Algorithm : $1.0 - (\frac{R3000}{R2530 * \frac{R2530}{R2210}})$

Code Snippet:

```
$ python csas.py --SINDEX input.IMG output.tif
```

Band Depth 3100

Rationale : H2O ice

Algorithm :

$$a = \frac{wv3120 - wv3000}{wv3250 - wv3000}$$
$$b = 1.0 - a$$
$$bd3100 = 1.0 - (\frac{(R3120}{(b * R3000) + (a * R3250)})$$

Code Snippet:

```
$ python csas.py --BD3100 input.IMG output.tif
```

Band Depth 3200

Rationale : CO2 ice

Algorithm :

$$a = \frac{wv3320 - wv3250}{wv3390 - wv3250}$$
$$b = 1.0 - a$$
$$bd3200 = 1.0 - \frac{(R3320}{(b * R3250) + (a * R3390)}$$

Code Snippet:

```
$ python csas.py --BD3200 input.IMG output.tif
```

BD3400

Rationale : Carbonates; organics

Algorithm :

$$c = \frac{(0.5(wv3390 + wv3500)) - wv3250}{wv3630 - wv3250}$$
$$d = 1.0 - c$$
$$BD3400 = 1.0 - \frac{(R3390 + R3500) * 0.5}{(d * R3250) + (c * R3630)}$$

Code Snippet:

```
$ python csas.py --BD3400 input.IMG output.tif
```

CINDEX

Rationale : Carbonates

Algorithm : $\frac{R3750+(R3750-R3630)}{(wv3750-wv3630)*(wv3920-wv3750)}/(R3920 - 1)$

Code Snippet:

```
$ python csas.py --BD3400 input.IMG output.tif
```

Note: Algorithm differs from published - coded as per CAT
