COAL Documentation

Release 0.5.0-dev

Lewis John McGibbney, Kim D. Whitehall, Taylor Alexander Brown

Apr 30, 2017

Contents

Contents:

CHAPTER 1

Introduction

COAL is a Python library for processing hyperspectral imagery from remote sensing devices such as the Airborne Visible/InfraRed Imaging Spectrometer (AVIRIS). COAL is being developed as a 2016–2017 senior capstone collaboration between scientists at the Jet Propulsion Laboratory (JPL) and computer science students at Oregon State University (OSU). COAL aims to provide a suite of algorithms for classifying land cover, identifying mines and other geographic features, and correlating them with environmental data sets. COAL is Free and Open Source Software under the terms of the Apache License Version 2.0.

What is Pycoal?

pycoal provides a suite of algorithms (written in Python) to identify, classify, characterize, and quantify (by reporting a number of key metrics) the direct and indirect impacts of MTM and related destructive surface mining activities across the continental U.S.A (and further afield).

Dependencies

- Spectral Python: needed for the mineral classification and mining identification APIs.
- NumPy: needed for the mineral classification and mining identification APIs.
- GDAL: needed for the GIS processing API.

More information on coal can be seen at the project Website as well as the docs directory.

CHAPTER 2

Quickstart

In the examples directory you can find several Jupyter notebooks with specific applications of coal. You can launch a cloud Jupyter server using binder to edit the notebooks without installing anything. Try it out!

CHAPTER 3

Mineral Classification API

class `pycoal.mineral.`**`MineralClassification`**(*libraryFilename*, *classNames=None*, *threshold=0.0*, *inMemory=False*)

`__init__`(*libraryFilename*, *classNames=None*, *threshold=0.0*, *inMemory=False*)

Construct a new `MineralClassification` object with a spectral library in ENVI format such as the USGS Digital Spectral Library 06 or the ASTER Spectral Library Version 2.0 converted with `pycoal.mineral.AsterConversion.convert()`.

If provided, the optional class name parameter will initialize the classifier with a subset of the spectral library, otherwise the full spectral library will be used.

The optional threshold parameter defines a confidence value between zero and one below which classifications will be discarded, otherwise all classifications will be included.

In order to improve performance on systems with sufficient memory, enable the optional parameter to load entire images.

Parameters

- **`libraryFilename`** (*`str`*) – filename of the spectral library
- **`classNames`** (*`str[], optional`*) – list of names of classes to include
- **`threshold`** (*`float, optional`*) – classification threshold
- **`inMemory`** (*`boolean, optional`*) – enable loading entire image

`classifyImage`(*imageFilename*, *classifiedFilename*)

Classify minerals in an AVIRIS image using spectral angle mapper classification and save the results to a file.

Parameters

- **`imageFilename`** (*`str`*) – filename of the image to be classified
- **`classifiedFilename`** (*`str`*) – filename of the classified image

Returns None

static filterClasses (*classifiedFilename*)
Modify a classified image to remove unused classes.

Parameters **classifiedFilename** (*str*) – file of the classified image

Returns None

static subsetSpectralLibrary (*spectralLibrary*, *classNames*)
Create a copy of the spectral library containing only the named classes.

Parameters

- **spectralLibrary** (*SpectralLibrary*) – ENVI spectral library
- **classNames** (*str[]*) – list of names of classes to include

Returns subset of ENVI spectral library

Return type SpectralLibrary

static toRGB (*imageFilename*, *rgbImageFilename*, *red=680.0*, *green=532.5*, *blue=472.5*)
Generate a three-band RGB image from an AVIRIS image and save it to a file.

Parameters

- **imageFilename** (*str*) – filename of the source image
- **rgbImageFilename** (*str*) – filename of the three-band RGB image
- **red** (*float, optional*) – wavelength in nanometers of the red band
- **green** (*float, optional*) – wavelength in nanometers of the green band
- **blue** (*float, optional*) – wavelength in nanometers of the blue band

Returns None

class pycoal.mineral.**AsterConversion**

__init__()
This class provides a method for converting the ASTER Spectral Library Version 2.0 into ENVI format.

Parameters **None** –

classmethod convert (*data_dir=''*, *db_file=''*, *hdr_file=''*)
This class method generates an ENVI format spectral library file. data_dir is optional as long as db_file is provided. Note that generating an SQLite database takes upwards of 10 minutes and creating an ENVI format file takes up to 5 minutes. Note: This feature is still experimental.

Parameters

- **data_dir** (*str, optional*) – path to directory containing ASCII data files
- **db_file** (*str*) – name of the SQLite file that either already exists if data_dir isn't provided, or will be generated if data_dir is provided
- **hdr_file** (*str*) – name of the ENVI spectral library to generate (without the .hdr or .sli extension)

CHAPTER 4

Mining Identification API

class `pycoal.mining.`**`MiningClassification`**(*classNames=[u'Schwertmannite BZ93-1 s06av95a=b', u'Renyolds_TnlSldgWet SM93-15w s06av95a=a', u'Renyolds_Tnl_Sludge SM93-15 s06av95a=a']*)

`__init__`(*classNames=[u'Schwertmannite BZ93-1 s06av95a=b', u'Renyolds_TnlSldgWet SM93-15w s06av95a=a', u'Renyolds_Tnl_Sludge SM93-15 s06av95a=a']*)

Construct a new MiningClassification object given an optional list of spectral class names which defaults to coal mining proxies.

Parameters **`classNames`** (`str[]`) – list of class names to identify.

`classifyImage`(*imageFilename*, *classifiedFilename*)

Classify mines or other features in a PyCOAL mineral classified image by copying relevant pixels and discarding the rest in a new file.

Parameters

- **`imageFilename`** (`str`) – filename of the image to be classified
- **`classifiedFilename`** (`str`) – filename of the classified image

Returns None

CHAPTER 5

Environmental Correlation API

class `pycoal.environment.`**`EnvironmentalCorrelation`**

`__init__``()`
Construct a new `EnvironmentalCorrelation` object.

`createEmptyCopy`(*sourceFilename*, *destinationFilename*)
Create an empty copy of a PyCOAL classified image with the same size.

Parameters

- **`sourceFilename`** (*`str`*) – filename of the source image
- **`destinationFilename`** (*`str`*) – filename of the destination image

`intersectProximity`(*miningFilename*, *vectorFilename*, *proximity*, *correlatedFilename*)
Generate an environmental correlation image containing pixels from the mining classified image detected within a given distance of features within a vector layer.

Parameters

- **`miningImage`** (*`str`*) – filename of the mining classified image
- **`vectorLayer`** (*`str`*) – filename of vector layer
- **`proximity`** (*`float`*) – distance in meters
- **`correlatedImage`** (*`str`*) – filename of the correlated image

`proximity`(*featureFilename*, *proximityFilename*)
Generate a proximity map from the features.

Parameters

- **`featureFilename`** (*`str`*) – filename of the feature image
- **`proximityFilename`** (*`str`*) – filename of the proximity image

`rasterize`(*vectorFilename*, *featureFilename*)
Burn features from a vector image onto a raster image.

Parameters

- **vectorFilename** (*str*) – filename of the vector image
- **featureFilename** (*str*) – filename of the raster image

CHAPTER 6

Indices and tables

- genindex

Index

Symbols

A

C

E

F

I

M

P

R

S

T