demcoreg Documentation

Release 0.2.0

David Shean

Mar 02, 2017

Contents

CHAPTER 1

Introduction

demcoreg contains Python function libraries and command-line tools for DEM-coregistration.

These tools can also be used to align two arbitrary rasters with sub-pixel precision.

Documentation is a work in progress...

README

demcoreg

Python and shell scripts for co-registration of rasters, specifically digital elevation models (DEMs).

Overview

All DEMs have some horizontal and vertical geolocation error. It is important to remove relative offsets when differencing multiple DEMs for elevation change analyses. These tools offer several options to solve this problem. Most solve for the sub-pixel horizontal shift and vertical offset required to minimize errors over "static" control surfaces. The ASP pc_align tool can also solve for more complex transformations with rotations and scaling.

Features

- Multiple co-registration algorithms (ICP, NCC, SAD, Nuth and Kaab [2011])
- Automatic determination of static control surfaces (i.e., exposed bedrock) for arbitrary DEM timestamp
- Least-squares optimization to correct a group of DEMs

Some useful command-line utilities (run with no arguments for usage)

- dem_mask.py - generate mask of snow-free rock surfaces using reflectance, LULC, SNODAS, MODSCAG
- pc_align_wrapper.sh - wrapper around NASA Ames Stereo Pipeline pc_align utility for iterative closest point co-registration
- apply_dem_translation.py - update geotransform and applies vertical offset
- compute_dz.py - simple DEM difference calculation
- robust_stats.py - print out robust statistics for sampled DEM differences before/after co-registration
- ...
- coreglib.py - implementation of various co-registration algorithms: Nuth and Kaab (2011), normalized cross-correlation with sub-pixel refinement, sum of absolute differences

Examples

dem_mask.py output

filter_glas.py output

Documentation

http://demcoreg.readthedocs.io

Installation

Install the latest release from PyPI:

```
pip install demcoreg
```

Note: by default, this will deploy executable scripts in /usr/local/bin

Building from source

Clone the repository and install:

```
git clone https://github.com/dshean/demcoreg.git
pip install -e demcoreg
```

The -e flag ("editable mode", setuptools "develop mode") will allow you to modify source code and immediately see changes.

Core requirements

- GDAL/OGR
- NumPy
- pygeotools

Optional requirements (needed for some functionality)

- matplotlib
- SciPy
- NASA Ames Stereo Pipeline (ASP)

License

Citation

If you use any of this software for research applications that result in publications, please cite:

Shean, D. E., O. Alexandrov, Z. Moratto, B. E. Smith, I. R. Joughin, C. C. Porter, Morin, P. J., An automated, open-source pipeline for mass production of digital elevation models (DEMs) from very high-resolution commercial stereo satellite imagery, ISPRS J. Photogramm. Remote Sens, 116, 101-117, doi: 10.1016/j.isprsjprs.2016.03.012, 2016.

CHAPTER 2

Command-line utilities

apply_dem_translation

Apply existing pc_align translation to a DEM

```
usage: apply_dem_translation [-h] [-outdir OUTDIR] dem_fn log_fn
```

Required Arguments

dem_fn	DEM filename
log_fn	pc_align log filename

Optional Arguments

-outdir	Output directory

compute_dz

Compute difference between two rasters

```
usage: compute_dz [-h] [-outdir OUTDIR] fn1 fn2
```

Required Arguments

fn1	Raster filename 1
fn2	Raster filename 2

Optional Arguments

-outdir	Output directory

dem_align

Perform DEM co-registration using old algorithms

```
usage: dem_align [-h] [-mode {ncc,sad,nuth,none}] [-mask_fn MASK_FN]
                 [-outdir OUTDIR]
                 ref_fn src_fn
```

Required Arguments

ref_fn	Reference DEM filename
src_fn	Source DEM filename to be moved

Optional Arguments

-mode="ncc"	Type of co-registration to use
	Possible choices: ncc, sad, nuth, none
-mask_fn	Mask filename
-outdir	Output directory

dem_mask

Identify control surfaces for DEM co-registration

```
usage: dem_mask [-h] [--toa] [--toa_thresh TOA_THRESH] [--snodas]
                [--snodas_thresh SNODAS_THRESH] [--modscag]
                [--modscag_thresh MODSCAG_THRESH]
                [--bareground_thresh BAREGROUND_THRESH] [--no_icemask]
                [--filter {rock,rock+ice,rock+ice+water,not_forest}]
                [--dilate DILATE]
                dem_fn
```

Required Arguments

dem_fn	DEM filename

Optional Arguments

--toa=False	Use top-of-atmosphere reflectance values (requires pregenerated "dem_fn_toa.tif")
--toa_thresh=0.4	Top-of-atmosphere reflectance threshold (default: 0.4, valid range 0.0-1.0), mask values greater than this value
--snodas=False	Use SNODAS snow depth products
--snodas_thresh=0.2	SNODAS snow depth threshold (default: 0.2 m), mask values greater than this value
--modscag=False	Use MODSCAG fractional snow cover products
--modscag_thresh=50	MODSCAG fractional snow cover percent threshold (default: 50%, valid range 0-100), mask greater than this value
--bareground_thresh=80	Percent bareground threshold (default: 80%, valid range 0-100), mask greater than this value (only relevant for global bareground data)

--no_icemask=False Don't mask glacier polygons

--filter="rock+ice+water" Preserve these LULC pixels (default: "rock+ice+water")

Possible choices: rock, rock+ice, rock+ice+water, not_forest

--dilate Dilate mask with this many iterations (default: None)

Utility to automate reference surface identification for raster co-registration

Note: Initial run may take a long time to download and process required data (NLCD, global bareground, glacier polygons)

Can control location of these data files with DATADIR environmental variable

export DATADIR=dir

Dependencies: gdal, wget, requests, bs4

demcoreg.dem_mask.**get_bareground**(*datadir=None*)
Calls external shell script *get_bareground.sh* to fetch:

~2010 global bare ground, 30 m

Note: unzipped file size is 64 GB! Original products are uncompressed, and tiles are available globally (including empty data over ocean)

The shell script will compress all downloaded tiles using lossless LZW compression.

http://landcover.usgs.gov/glc/BareGroundDescriptionAndDownloads.php

demcoreg.dem_mask.**get_glacier_poly**(*datadir=None*)
Calls external shell script *get_glacier_poly.sh* to fetch:

Randolph Glacier Inventory (RGI) glacier outline shapefiles

Full RGI database: rgi50.zip is 410 MB

The shell script will unzip and merge regional shp into single global shp

http://www.glims.org/RGI/

demcoreg.dem_mask.**get_icemask**(*ds*, *datadir=None*, *glac_shp_fn=None*)
Generate glacier polygon raster mask for input Dataset res/extent

demcoreg.dem_mask.**get_modis_tile_list**(*geom*)
Helper function to identify MODIS tiles that intersect input geometry

modis_gird.py contains dictionary of tile boundaries (tile name and WKT polygon ring from bbox)

See: https://modis-land.gsfc.nasa.gov/MODLAND_grid.html

demcoreg.dem_mask.**get_modscag**(*dem_dt*, *outdir=None*, *tile_list=('h08v04', 'h09v04', 'h10v04', 'h08v05', 'h09v05')*, *pad_days=7*)
Function to fetch and process MODSCAG fractional snow cover products for input datetime

Products are tiled in MODIS sinusoidal projection

example url: https://snow-data.jpl.nasa.gov/modscag-historic/2015/001/MOD09GA.A2015001.h07v03.005.2015006001833.snow_fraction.tif

demcoreg.dem_mask.**get_nlcd**(*datadir=None*)
Calls external shell script *get_nlcd.sh* to fetch:

2011 Land Use Land Cover (nlcd) grids, 30 m

http://www.mrlc.gov/nlcd11_leg.php

demcoreg.dem_mask.**get_snodas**(*dem_dt*, *outdir=None*)
Function to fetch and process SNODAS snow depth products for input datetime

http://nsidc.org/data/docs/noaa/g02158_snodas_snow_cover_model/index.html

1036 is snow depth

filename format: us_ssmv11036tS__T0001TTNATS2015042205HP001.Hdr

demcoreg.dem_mask.**getparser**()

demcoreg.dem_mask.**main**()

demcoreg.dem_mask.**mask_bareground**(*ds*, *minperc=80*, *mask_glaciers=True*)
Generate raster mask for exposed bare ground from global bareground data

demcoreg.dem_mask.**mask_nlcd**(*ds*, *valid='rock+ice+water'*, *datadir=None*, *mask_glaciers=True*)
Generate raster mask for exposed rock in NLCD data

demcoreg.dem_mask.**proc_modscag**(*fn_list*, *extent=None*, *t_srs=None*)
Process the MODSCAG products for full date range, create composites and reproject

glas_proc

Process and filter ICESat GLAS points

```
usage: glas_proc [-h] [-extent EXTENT] [-name NAME] [-dem_fn DEM_FN] fn
```

Required Arguments

fn	GLAH14 HDF5 filename

Optional Arguments

-extent	Output spatial extent
-name	Output file prefix
-dem_fn	Output file prefix

robust_stats

Compute robust stats from dz or pc_align sample.csv

```
usage: robust_stats [-h] [-outdir OUTDIR] fn
```

Required Arguments

fn	Input filename

Optional Arguments

-outdir	Output directory

vol_stats

Compute volume/mass change stats from DEM difference

```
usage: vol_stats [-h] [-rho RHO] fn
```

Required Arguments

fn	Elevation difference filename (dz.tif)

Optional Arguments

-rho=0.917	Density for mass change calculation (default: 0.917)

pc_align_wrapper

Wrapper for ASP pc_align utility

dem_coreg

Co-register a single DEM to a reference DEM:

1. *apply_mask.py* to apply mask from *dem_mask.py* output to refdem
2. *pc_align_wrapper.sh*
3. *apply_dem_translation.py* on DEM-32m.tif and DEM_8m.tif

dem_coreg_all

Co-register a batch of DEMs to a reference DEM:

1. *toa.sh* to create top-of-atmosphere reflectance for DigitalGlobe images
2. *dem_mask.py* to identify control surfaces
3. *dem_coreg.sh* to perform co-registration

CHAPTER 3

coreglib module

Library of functions that can be used for horizontal co-registration of raster data

These were written in 2012-2013, and should be cleaned up and tested before use

The ASP pc_align ICP co-registration is usually superior to these approaches

demcoreg.coreglib.**bin_by**(*x*, *y*, *nbins=360*)

demcoreg.coreglib.**compute_offset_ncc**(*dem1*, *dem2*, *pad=(9, 9)*, *plot=False*)
Compute horizontal offset between input rasters using normalized cross-correlation (NCC) method

demcoreg.coreglib.**compute_offset_nuth**(*dh*, *slope*, *aspect*)
Compute horizontal offset between input rasters using Nuth and Kaab [2011] (nuth) method

demcoreg.coreglib.**compute_offset_sad**(*dem1*, *dem2*, *pad=(9, 9)*, *plot=False*)
Compute horizontal offset between input rasters using sum of absolute differences (SAD) method

demcoreg.coreglib.**find_first_peak**(*corr*)
Find row and column indices of the first correlation peak.

Parameters **corr** (*np.ndarray*) – the correlation map

Returns

- **i** (*int*) – the row index of the correlation peak
- **j** (*int*) – the column index of the correlation peak
- **corr_max1** (*int*) – the value of the correlation peak
- *Original code from openPIV pyprocess*

demcoreg.coreglib.**find_subpixel_peak_position**(*corr*, *subpixel_method='gaussian'*)
Find subpixel approximation of the correlation peak.

This function returns a subpixels approximation of the correlation peak by using one of the several methods available. If requested, the function also returns the signal to noise ratio level evaluated from the correlation map.

Parameters

- **corr** (*np.ndarray*) – the correlation map.
- **subpixel_method** (*string*) – one of the following methods to estimate subpixel location of the peak: 'centroid' [replaces default if correlation map is negative], 'gaussian' [default if correlation map is positive], 'parabolic'.

Returns

- **subp_peak_position** (*two elements tuple*) – the fractional row and column indices for the sub-pixel approximation of the correlation peak.
- *Original code from openPIV pyprocess*

demcoreg.coreglib.**func**(*x*, *a*, *b*, *c*)

demcoreg.coreglib.**genplot**(*x*, *y*, *fit*)

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index