pybob Documentation

Robert McNabb

Sep 29, 2019

Contents

CHAPTER 1

modules

Modules

1.1 pybob.GeoImg

pybob.GeoImg is a class to handle geospatial imagery, in particular satellite images and digital elevation models.

class pybob.GeoImg.**GeoImg**(*in_filename*, *in_dir=None*, *datestr=None*, *datefmt='%m/%d/%y'*, *dtype=<MagicMock id='140009480761864'>*, *attrs=None*)

Create a GeoImg object from a GDAL-supported raster dataset.

__init__(*in_filename*, *in_dir=None*, *datestr=None*, *datefmt='%m/%d/%y'*, *dtype=<MagicMock id='140009480761864'>*, *attrs=None*)

Parameters

- **in_filename** (*str, gdal.Dataset*) – Filename or object to read in. If in_filename is a string, the GeoImg is created by reading the file corresponding to that filename. If in_filename is a gdal object, the GeoImg is created by operating on the corresponding object.
- **in_dir** (*str*) – (optional) directory where in_filename is located. If not given, the directory will be determined from the input filename.
- **datestr** (*str*) – (optional) string to pass to GeoImg, representing the date the image was acquired.
- **datefmt** – Format of datestr that datetime.datetime should use to parse datestr. Default is %m/%d/%y.
- **dtype** (*numpy datatype*) – numpy datatype to read input data as. Default is np.float32. See numpy docs for more details.

copy(*new_raster=None*, *new_extent=None*, *driver='MEM'*, *filename=''*, *newproj=None*, *datatype=<MagicMock name='mock.GDT_Float32' id='140009481516256'>*)

Copy the GeoImg, creating a new GeoImg, optionally updating the extent and raster.

Parameters

- **new_raster** (*array-like*) – New raster to use. If not set, the new GeoImg will have the same raster as the old image.
- **new_extent** (*array_like*) – New extent to use, given as xmin, xmax, ymin, ymax. If not set, the old extent is used. If set, you must also include a new raster.
- **driver** (*str*) – gdal driver to use to create the new GeoImg. Default is 'MEM' (in-memory). See gdal docs for more options. If a different driver is used, filename must also be specified.
- **filename** (*str*) – Filename corresponding to the new image, if not created in memory.

Returns new_geo A new GeoImg with the specified extent and raster.

crop_to_extent (*extent*, *pixel_size=None*, *bands=None*)

Crop image to given extent.

Parameters

- **extent** (*matplotlib.figure.Figure or array-like*) – Extent to which image should be cropped. If extent is a matplotlib figure handle, the image extent is taken from the x and y limits of the current figure axes. If extent is array-like, it is assumed to be [xmin, xmax, ymin, ymax]
- **pixel_size** (*float*) – Set pixel size of output raster. Default is calculated based on current pixel size and extent.
- **bands** (*array-like*) – Image band(s) to crop - default assumes first (only) band. Remember that numpy indices start at 0 - i.e., the first band is band 0.

Returns cropped_img new GeoImg object resampled to the given image extent.

display (*fig=None, cmap='gray', extent=None, sfact=None, showfig=True, band=[0, 1, 2], **kwargs*)

Display GeoImg in a matplotlib figure.

Parameters

- **fig** (*matplotlib.figure.Figure*) – figure handle to show image in. If not set, creates a new figure.
- **cmap** (*matplotlib colormap*) – colormap to use for the image. Default is gray.
- **extent** (*array-like*) – spatial extent to limit the figure to, given as xmin, xmax, ymin, ymax.
- **sfact** (*int*) – Factor by which to reduce the number of pixels plotted. Default is 1 (i.e., all pixels are displayed).
- **showfig** (*bool*) – Open the figure window. Default is True.
- **band** (*array-like*) – Image bands to use, if GeoImg represents a multi-band image.
- **kwargs** – Optional keyword arguments to pass to matplotlib.pyplot.imshow

Returns fig Handle pointing to the matplotlib Figure created (or passed to display).

find_corners (*nodata=<MagicMock id='140009481518496'>*, *mode='ij'*)

Find corner coordinates of valid image area.

Parameters

- **nodata** (*numeric*) – nodata value to use. Default is numpy.nan

- **mode** (*str*) – Type of coordinates to return. Options are 'ij', row/column index, or 'xy', x,y coordinate. Default is 'ij'.

Returns corners Array corresponding to the corner coordinates, estimated from the convex hull of the valid data.

find_valid_bbox(*nodata=<MagicMock id='140009480121200'>*)
Find bounding box for valid data.

Parameters **nodata** (*numeric*) – nodata value to use for the image. Default is numpy.nan

Returns bbox xmin, xmax, ymin, ymax of valid image area.

ij2xy(*ij*)
Return x,y coordinates for a given row, column index pair.

Parameters **ij** (*float*) – row (i) and column (j) index of pixel.

Returns xy x,y coordinates of i,j in the GeoImg's spatial reference system.

info()
Prints information about the GeoImg (filename, coordinate system, number of columns/rows, etc.).

is_area()
Check if pixel coordinates correspond to pixel corners.

Returns is_area True if pixel coordinates correspond to pixel corners.

is_point()
Check if pixel coordinates correspond to pixel centers.

Returns is_point True if pixel coordinates correspond to pixel centers.

is_rotated()
Determine whether GeoImg is rotated with respect to North.

mask(*mask*, *mask_value=True*)
Mask image values.

Parameters

- **mask** (*array-like*) – Array of same size as self.img corresponding to values that should be masked.
- **mask_value** (*bool or numeric*) – Value within mask to mask. If True, masks image where mask is True. If numeric, masks image where mask == mask_value.

match_sensor(*fname*, *datestr=None*, *datefmt=''*)
Attempts to pull metadata (e.g., sensor, date information) from fname, setting sensor_name, satellite, tile, datetime, and date attributes of GeoImg object.

Parameters

- **fname** (*str*) – filename of image to parse
- **datestr** (*str*) – optional datestring to set date attributes
- **datefmt** (*str*) – optional datetime format for datestr

mean()
Returns mean (ignoring NaNs) of the image.

median()
Returns median (ignoring NaNs) of the image.

outside_image(*ij*, *index=True*)
Check whether a given point falls outside of the image.

Parameters

- **ij** (*array-like*) – Indices (or coordinates) of point to check.
- **index** (*bool*) – Interpret ij as raster indices (default is True). If False, assumes ij is coordinates.

Returns is_outside True if ij is outside of the image.

overlay(*raster*, *extent=None*, *vmin=0*, *vmax=10*, *sfact=None*, *showfig=True*, *alpha=0.25*, *cmap='jet'*)
Overlay raster on top of GeoImg.

Parameters

- **raster** (*array-like*) – raster to display on top of the GeoImg.
- **extent** (*array-like*) – Spatial of the raster. If not set, assumed to be same as the extent of the GeoImg. Given as xmin, xmax, ymin, ymax.
- **vmin** (*float*) – minimum color value for the raster. Default is 0.
- **vmax** (*float*) – maximum color value for the raster. Default is 10.
- **sfact** (*int*) – Factor by which to reduce the number of points plotted. Default is 1 (i.e., all points are plotted).
- **showfig** (*bool*) – Open the figure window. Default is True.
- **alpha** (*float*) – Alpha value to use for the overlay. Default is 0.25
- **cmap** (*str*) – matplotlib.pyplot colormap to use for the image. Default is jet.

Returns fig Handle pointing to the matplotlib Figure created (or passed to display).

random_points(*Npts*, *edge_buffer=None*)
Generate a random sample of points within the image.

Parameters

- **Npts** (*int*) – number of random points to sample.
- **edge_buffer** (*int*) – Optional buffer around edge of image, where pixels shouldn't be sampled. Default is zero.

Returns rand_pts array of N random points from within the image.

raster_points(*pts*, *nsize=1*, *mode='linear'*)
Interpolate raster values at a given point, or sets of points.

Parameters

- **pts** (*array-like*) – Point(s) at which to interpolate raster value. If points fall outside of image, value returned is nan.'
- **nsize** (*int*) – Number of neighboring points to include in the interpolation. Default is 1.
- **mode** (*str*) – One of 'linear', 'cubic', or 'quintic'. Determines what type of spline is used to interpolate the raster value at each point. For more information, see scipy.interpolate.interp2d. Default is linear.

Returns rpts Array of raster value(s) for the given points.

raster_points2 (*pts*, *nsize=1*, *mode='linear'*)
Interpolate raster values at a given point, or sets of points using multiprocessing for speed.

Parameters

- **pts** (*array-like*) – Point(s) at which to interpolate raster value. If points fall outside of image, value returned is nan.'
- **nsize** (*int*) – Number of neighboring points to include in the interpolation. Default is 1.
- **mode** (*str*) – One of 'linear', 'cubic', or 'quintic'. Determines what type of spline is used to interpolate the raster value at each point. For more information, see scipy.interpolate.interp2d.

Returns rpts Array of raster value(s) for the given points.

reproject (*dst_raster*, *driver='MEM'*, *filename=''*, *method=<MagicMock name='mock.GRA_Bilinear' id='140009479948888'>*)
Reproject the GeoImg to the same extent and coordinate system as another GeoImg.

Parameters

- **dst_raster** (GeoImg) – GeoImg to project given raster to.
- **driver** (*str*) – gdal driver to use to create the new GeoImg. Default is 'MEM' (in-memory). See gdal docs for more options. If a different driver is used, filename must also be specified.
- **filename** (*str*) – Filename corresponding to the new image, if not created in memory.
- **method** (*gdal_GRA*) – gdal resampling algorithm to use. Default is GRA_Bilinear. Other options include: GRA_Average, GRA_Cubic, GRA_CubicSpline, GRA_NearestNeighbour. See gdal docs for more options and details.

Returns new_geo reprojected GeoImg.

set_NDV (*NDV*)
Set nodata value to given value

Parameters **NDV** (*numeric*) – value to set to nodata.

shift (*xshift*, *yshift*)
Shift the GeoImg in space by a given x,y offset.

Parameters

- **xshift** (*float*) – x offset to shift GeoImg by.
- **yshift** (*float*) – y offset to shift GeoImg by.

std ()
Returns standard deviation (ignoring NaNs) of the image.

subimages (*N*, *Ny=None*, *sBuffer=0*)
Split the GeoImg into sub-images.

Parameters

- **N** (*int*) – number of column cells to split the image into.
- **Ny** (*int*) – number of row cells to split image into. Default is same as N.
- **sBuffer** (*int*) – number of pixels to overlap subimages by. Default is 0.

Returns sub_images list of GeoImg tiles.

to_area()
Change pixel location from center ('Point') to corner ('Area'). Shifts raster by half pixel in the -x, +y direction.

to_point()
Change pixel location from corner ('Area') to center ('Point'). Shifts raster by half pixel in the +x, -y direction.

unmask()
Remove mask from image. If mask is not set, has no effect.

write(*outfilename*, *out_folder=None*, *driver='GTiff'*, *dtype=None*, *bands=None*)
Write GeoImg to a gdal-supported raster file.

Parameters

- **outfilename** (*str*) – string representing the filename to be written to.
- **out_folder** (*str*) – optional string representing the folder to be written to. If not set, folder is either guessed from outfilename, or assumed to be the current folder.
- **driver** (*str*) – optional string representing the gdal driver to use to write the raster file. Default is GTiff. Options include: HDF4, HDF5, JPEG, PNG, JPEG2000 (if enabled). See gdal docs for more options.
- **datatype** (*numpy datatype*) – Type of data to write the raster as. Check GeoImg.numpy2gdal.keys() to see numpy data types implemented.
- **bands** (*array-like*) – Specify band(s) to write to file. Default behavior is all bands.

xy(*ctype='corner'*, *grid=True*)
Get x,y coordinates of all pixels in the GeoImg.

Parameters

- **ctype** (*str*) – coordinate type. If 'corner', returns corner coordinates of pixels. If 'center', returns center coordinates. Default is corner.
- **grid** (*bool*) – Return gridded coordinates. Default is True.

Returns x,y numpy arrays corresponding to the x,y coordinates of each pixel.

xy2ij(*xy*)
Return row, column indices for a given x,y coordinate pair.

Parameters **xy** (*float*) – x, y coordinates in the GeoImg's spatial reference system.

Returns ij i,j indices of x,y in the image.

1.2 pybob.ICESat

pybob.ICESat provides an interface to extracted ICESat(-1) data stored in HDF5 format. Pre-extracted ICESat tracks for each of the RGI subregions can be found here.

class pybob.ICESat.**ICESat**(*in_filename, in_dir=None, cols=['lat', 'lon', 'd_elev', 'd_UTCTime'], ellipse_hts=True*)
Create an ICESat dataset from an HDF5 file containing ICESat data.

__init__(*in_filename, in_dir=None, cols=['lat', 'lon', 'd_elev', 'd_UTCTime'], ellipse_hts=True*)

Parameters

- **in_filename** (*str*) – Filename (optionally with path) of HDF5 file to be read in.

- **in_dir** (*str*) – Directory where in_filename is located. If not given, the directory will be determined from the input filename.
- **cols** (*array-like*) – Columns to read in and set as attributes in the ICESat object. Default values are ['lat', 'lon', 'd_elev', 'd_UTCTime'], and will be added if not specified.
- **ellipse_hts** (*bool*) – Convert elevations to ellipsoid heights. Default is True.

The following example:

```
>>> ICESat('donjek_icesat.h5', ellipse_hts=True)
```

will return an ICESat object with lat, lon, elevation, and UTCTime attributes, with elevations given as height above WGS84 ellipsoid.

clean (*el_limit=-500*)
Remove all elevation points below a given elevation.

Parameters **el_limit** (*float*) – minimum elevation to accept

clip (*bounds*)
Remove ICEsat data that falls outside of a given bounding box.

Parameters **bounds** (*array-like*) – bounding box to clip to, given as xmin, xmax, ymin, ymax

display (*fig=None*, *extent=None*, *sfact=1*, *showfig=True*, *geo=False*, ***kwargs*)
Plot ICESat tracks in a figure.

Parameters

- **fig** (*matplotlib.figure.Figure*) – Figure to plot tracks in. If not set, creates a new figure.
- **extent** (*array-like*) – Spatial extent to limit the figure to, given as xmin, xmax, ymin, ymax.
- **sfact** (*int*) – Factor by which to reduce the number of points plotted. Default is 1 (i.e., all points are plotted).
- **showfig** (*bool*) – Open the figure window. Default is True.
- **geo** (*bool*) – Plot tracks in lat/lon coordinates, rather than projected coordinates. Default is False.
- **kwargs** – Optional keyword arguments to pass to matplotlib.pyplot.plot

Returns fig Handle pointing to the matplotlib Figure created (or passed to display).

from_ellipse ()
Convert ICESat elevations from ellipsoid heights, based on the data stored in the HDF5 file.

get_bounds (*geo=False*)
Return bounding coordinates of the dataset.

Parameters **geo** (*bool*) – Return geographic (lat/lon) coordinates (default is False)

Returns bounds xmin, xmax, ymin, ymax values of dataset.

Example: >>> xmin, xmax, ymin, ymax = my_icesat.get_bounds()

mask (*mask*, *mask_value=True*, *drop=False*)
Mask values

Parameters

- **mask** (*array-like*) – Array of same size as self.img corresponding to values that should be masked.
- **mask_value** (*bool or numeric*) – Value within mask to mask. If True, masks image where mask is True. If numeric, masks image where mask == mask_value.
- **drop** (*bool*) – remove values from dataset (True), or mask using numpy.masked_array (False). Default is False.

project(*dest_proj*)

Project the ICESat dataset to a given coordinate system, using pyproj.transform. ICESat.project does not overwrite the lat/lon coordinates, so calling ICESat.project will only update self.x,self.y for the dataset, self.xy, and self.proj.

Parameters **dest_proj** (*str or pyproj.Proj*) – Coordinate system to project the dataset into. If dest_proj is a string, ICESat.project() will create a pyproj.Proj instance with it.

Example: Project icesat_data to Alaska Albers (NAD83) using epsg code:

```
>>> icesat_data.project('epsg:3338')
```

to_csv(*out_filename*)

Write ICESat data to csv format using pandas.

Parameters **out_filename** (*str*) – Filename (optionally with path) of file to read out.

to_ellipse()

Convert ICESat elevations to ellipsoid heights, based on the data stored in the HDF5 file.

to_shp(*out_filename*, *driver='ESRI Shapefile'*, *epsg=4326*)

Write ICESat data to shapefile.

Parameters

- **out_filename** (*str*) – Filename (optionally with path) of file to read out.
- **driver** (*str*) – Name of driver fiona should use to create the outpufile. Default is 'ESRI Shapefile'.
- **epsg** (*int*) – EPSG code to project data to. Default is 4326, WGS84 Lat/Lon.

Example:

```
>>> donjek_icesat.to_shp('donjek_icesat.shp', epsg=3338)
```

will write donjek_icesat to a shapefile in Alaska Albers projection (EPSG:3338)

unmask()

Remove a mask if it has been applied. **TODO: Not implemented yet!**

Returns

pybob.ICESat.**extract_ICESat**(*in_filename*, *workdir=None*, *outfile=None*)

Extract ICESat data given the extent of a GeoImg.

Note - currently not supported outside of UiO working environment.

Parameters

- **in_filename** (*str*) – Filename (optionally with path) of DEM to be opened using GeoImg.
- **workdir** (*str*) – Directory where in_filename is located. If not given, the directory will be determined from the input filename.

- **outfile** (*str*) – name of the output file [default='ICESat_DEM.h5']

1.3 pybob.VelField

1.4 pybob.bob_tools

pybob.bob_tools is a collection of the tools that didn't really fit other places.

pybob.bob_tools.**bin_data**(*bins*, *data2bin*, *bindata*, *mode='mean'*, *nbinned=False*)
Place data into bins based on a secondary dataset, and calculate statistics on them.

Parameters

- **bins** (*array-like*) – array-like structure indicating the bins into which data should be placed.
- **data2bin** (*array-like*) – data that should be binned.
- **bindata** (*array-like*) – secondary dataset that decides how data2bin should be binned. Should have same size/shape as data2bin.
- **mode** (*str*) – How to calculate statistics of binned data. One of 'mean', 'median', 'std', 'max', or 'min'.
- **nbinned** (*bool*) – Return a second array, nbinned, with number of data points that fit into each bin. Default is False.

Returns binned, nbinned calculated, binned data with same size as bins input. If nbinned is True, returns a second array with the number of inputs for each bin.

pybob.bob_tools.**doy2mmdd**(*year*, *doy*, *string_out=True*, *outform='%Y/%m/%d'*)
Return a string or a datetime object given a year and a day of year.

Parameters

- **year** (*int*) – Year of the input date, e.g., 2018.
- **doy** (*int*) – Day of the year of the input date, e.g., 1 for 1 Jan.
- **string_out** (*bool*) – Return a string (True) or a datetime object (False). Default is True.
- **outform** (*str*) – Format for string to return. Default is *%Y/%m/%d*, or 2018/01/01 for 1 January 2018.

Returns date datetime.datetime or string representation of the input date.

```
>>> bt.doy2mmdd(2018, 1, string_out=True)
`2018/01/01`
```

```
>>> bt.doy2mmdd(2018, 1, string_out=False)
datetime.datetime(2018, 1, 1, 0, 0)
```

pybob.bob_tools.**mkdir_p**(*out_dir*)
Add bash mkdir -p functionality to os.makedirs.

Parameters **out_dir** – directory to create.

pybob.bob_tools.**mmdd2doy**(*year*, *month*, *day*, *string_out=True*)
Return a string or an int representing a day of year, given a date.

Parameters

- **year** (*int*) – Year of the input date, e.g., 2018.
- **mm** (*int*) – Month of the year of the input date, e.g., 1 for Jan.
- **dd** (*int*) – Day of the month of the input date.
- **string_out** (*bool*) – Return a string (True) or an int (False). Default is True.

Returns doy day of year representation of year, month, day

```
>>> bt.mmdd2doy(2018, 1, 1, string_out=True)
`1`
```

pybob.bob_tools.**standard_landsat**(*instring*)

Given a string of a landsat scenename, make a standard (pre-Collection) filename, of the form LSSPPPRRRYYYYDDDXXX01.

1.5 pybob.coreg_tools

pybob.coreg_tools provides a toolset for coregistering DEMs, based on the method presented by Nuth and Kääb (2011).

pybob.coreg_tools.**RMSE**(*indata*)

Return root mean square of indata.

Parameters **indata** (*array-like*) – differences to calculate root mean square of

Returns myrmse RMSE of indata.

pybob.coreg_tools.**create_stable_mask**(*img*, *mask1*, *mask2*)

Create mask representing stable terrain, given exclusion (i.e., glacier) and inclusion (i.e., land) masks.

Parameters

- **img** (*pybob.GeoImg*) – GeoImg to pull extents from
- **mask1** (*str*) – filename for shapefile representing pixels to exclude from stable terrain (i.e., glaciers)
- **mask2** (*str*) – filename for shapefile representing pixels to include in stable terrain (i.e., land)

Returns stable_mask boolean array representing stable terrain

pybob.coreg_tools.**dem_coregistration**(*masterDEM*, *slaveDEM*, *glaciermask=None*, *landmask=None*, *outdir='.'*, *pts=False*, *full_ext=False*, *return_var=True*, *alg='Horn'*, *magnlimit=2*)

Iteratively co-register elevation data.

Parameters

- **masterDEM** (*str, pybob.GeoImg*) – Path to filename or GeoImg dataset representing "master" DEM.
- **slaveDEM** (*str, pybob.GeoImg*) – Path to filename or GeoImg dataset representing "slave" DEM.
- **glaciermask** (*str*) – Path to shapefile representing points to exclude from co-registration consideration (i.e., glaciers).
- **landmask** (*str*) – Path to shapefile representing points to include in co-registration consideration (i.e., stable ground/land).

- **outdir** (*str*) – Location to save co-registration outputs.
- **pts** (*bool*) – If True, program assumes that masterDEM represents point data (i.e., ICESat), as opposed to raster data. Slope/aspect are then calculated from slaveDEM. masterDEM should be a string representing an HDF5 file continaing ICESat data.
- **full_ext** (*bool*) – If True, program writes full extents of input DEMs. If False, program writes input DEMs cropped to their common extent. Default is False.
- **return_var** (*bool*) – return variables representing co-registered DEMs and offsets (default).
- **alg** (*str*) – Algorithm for calculating Slope, Aspect. One of 'ZevenbergenThorne' or 'Horn'. Default is 'Horn'.
- **magnlimit** (*float*) – Magnitude threshold for determining termination of co-registration algorithm, calculated as sum in quadrature of dx, dy, dz shifts. Default is 2 m.

Returns masterDEM, outslave, out_offs if return_var=True, returns master DEM, co-registered slave DEM, and x,y,z shifts removed from slave DEM.

If co-registration fails (i.e., there are too few acceptable points to perform co-registration), then returns original master and slave DEMs, with offsets set to -1.

pybob.coreg_tools.**get_aspect**(*geoimg*, *alg='Horn'*)

Wrapper function to calculate DEM aspect using gdal.DEMProcessing.

Parameters

- **geoimg** (*pybob.GeoImg*) – GeoImg object of DEM to calculate aspect
- **alg** (*str*) – Algorithm for calculating Aspect. One of 'ZevenbergenThorne' or 'Horn'. Default is 'Horn'.

Returns geo_aspect new GeoImg object with aspect raster

pybob.coreg_tools.**get_slope**(*geoimg*, *alg='Horn'*)

Wrapper function to calculate DEM slope using gdal.DEMProcessing.

Parameters

- **geoimg** (*pybob.GeoImg*) – GeoImg object of DEM to calculate slope
- **alg** (*str*) – Algorithm for calculating Slope. One of 'ZevenbergenThorne' or 'Horn'. Default is 'Horn'.

Returns geo_slope new GeoImg object with slope raster

1.6 pybob.ddem_tools

pybob.ddem_tools provides a number of tools for working with DEM differencing and calculating volume changes. Primarily designed for glaciers, but could be useful for calculating other volume changes as well.

pybob.ddem_tools.**area_alt_dist**(*DEM*, *glacier_shapes*, *glacier_inds=None*, *bin_width=None*)

Calculate an Area-Altitude Distribution for a glacier outline(s), given an input DEM.

Parameters

- **DEM** (*pybob.GeoImg*) – input DEM.
- **glacier_shapes** (*array-like*) – mask representing glacier outlines. Can be boolean or integer depending on whether one or several AADs should be calculated.

- **glacier_inds** (*array-like*) – array representing glacier indices in glacier_shapes. If unspecified, only one AAD is returned.
- **bin_width** (*float*) – width of elevation bands to calculate area distribution in. If unspecified, result will be the minimum of 50m or 10% of the elevation range.

Returns bins, aads array, or list of arrays, representing elevation bands and glacier area (in DEM horizontal units) per elevation band.

pybob.ddem_tools.**calculate_dV_curve**(*aad*, *dh_curve*)

Given a dh(z) curve and AAD values, calculate volume change.

Parameters

- **aad** (*np.array*) – Area-Altitude Distribution/Hypsometry with which to calculate volume change.
- **dh_curve** (*np.array*) – dh(z) curve to use.

Returns dV glacier volume change given input curves.

pybob.ddem_tools.**calculate_dV_map**(*dDEM*, *ind_glac_mask*, *ind_glac_vals*)

Calculate glacier volume changes from a dDEM using a map of glacier outlines.

Parameters

- **dDEM** (*pybob.GeoImg*) – difference DEM to get elevation changes from
- **ind_glac_mask** (*array-like*) – glacier mask with different values for each glacier
- **ind_glac_vals** (*array-like*) – list of unique index values in glacier mask for which to calculate volume changes.

Returns ind_glac_vals, ind_vol_chgs index and volume changes for the input indices.

pybob.ddem_tools.**get_bins**(*DEM*, *glacier_mask=None*, *bin_width=None*)

Get elevation bins for a DEM, given an optional glacier mask and an optional width. If unspecified, bin_width is calculated as the minimum of 50 units, or 10% of the DEM (or glacier, if mask provided) elevation range. Bin values represent the lower bound of the elevation band, and are rounded to be a multiple of the bin width.

Parameters

- **DEM** (*array-like*) – The DEM to get elevation bins for.
- **glacier_mask** (*array-like*) – mask representing glacier outline, or region of interest.
- **bin_width** (*float*) – width of bins to calculate.

Returns bins the elevation bins.

pybob.ddem_tools.**get_elev_curve**(*DEM*, *dDEM*, *glacier_mask=None*, *bins=None*, *mode='mean'*, *outlier=False*, *fill=False*, *poly_order=3*)

Get a dh(z) curve for a glacier/region of interest, given a DEM and a difference DEM (dDEM). Available modes are 'mean'/'median', calculating the mean(median) of each elevation bin, or poly, fitting a polynomial (default third-order) to the means of each elevation bin.

Parameters

- **DEM** (*array-like*) – DEM to determine z in dh(z)
- **dDEM** (*array-like*) – difference DEM to determine dh in dh(z)
- **glacier_mask** (*array-like*) – mask representing glacier outline

- **bins** (*array-like*) – values representing the lower edge of elevation bins
- **mode** (*str*) – how to determine the dh(z) relationship
- **outlier** (*bool*) – filter outliers using an iterative 3-sigma filter
- **fill** (*bool*) – fill missing bins using a polynomial fit (default third order)
- **poly_order** (*int*) – order for any polynomial fitting

Returns bins, curve, bin_areas elevation bins, dh(z) curve, and number of pixels per elevation bin.

pybob.ddem_tools.**nice_split**(*fname*)
Given a filename of the form dH_DEM1_DEM2, return DEM1, DEM2.

Parameters **fname** (*str*) – filename to split

Returns name1, name2 DEM names parsed from input filename.

pybob.ddem_tools.**nmad**(*data*, *nfact=1.4826*)
Calculate the normalized median absolute deviation (NMAD) of an array.

Parameters

- **data** (*array-like*) – input data
- **nfact** (*float*) – normalization factor for the data; default is 1.4826

Returns nmad (normalized) median absolute deviation of data.

pybob.ddem_tools.**outlier_removal**(*bins*, *DEM*, *dDEM*, *nsig=3*)
Iteratively remove outliers in an elevation bin using a 3-sigma filter.

Parameters

- **bins** (*array-like*) – lower bound of elevation bins to use
- **DEM** (*array-like*) – DEM to determine grouping for outlier values
- **dDEM** (*array-like*) – elevation differences to filter outliers from
- **nsig** (*float*) – number of standard deviations before a value is considered an outlier.

Returns new_ddem ddem with outliers removed (set to NaN)

1.7 pybob.hexagon_tools

pybob.hexagon_tools is a collection of tools for working with KH-9 Hexagon imagery.

pybob.hexagon_tools.**join_halves**(*img*, *overlap*, *indir='.'*, *outdir='.'*, *color_balance=True*)
Join scanned halves of KH-9 image into one, given a common overlap point.

Parameters

- **img** (*str*) – KH-9 image name (i.e., DZB1215-500454L001001) to join. The function will look for open image halves img_a.tif and img_b.tif, assuming 'a' is the left-hand image and 'b' is the right-hand image.
- **overlap** (*array-like*) – Image coordinates for a common overlap point, in the form [x1, y1, x2, y2]. Best results tend to be overlaps toward the middle of the y range. YMMV.
- **indir** (*str*) – Directory containing images to join ['.']
- **outdir** (*str*) – Directory to write joined image to ['.']

- **color_balance** (*bool*) – Attempt to color balance the two image halves before joining [True].

1.8 pybob.image_tools

pybob.image_tools is a collection of tools related to working with images.

pybob.image_tools.**create_mask_from_shapefile**(*geoimg*, *shapefile*, *buffer=None*)
Create a boolean mask representing the polygons in a shapefile.

Parameters

- **geoimg** (*pybob.GeoImg*) – input GeoImg to pull spatial extents from
- **shapefile** (*str*) – path to polygon shapefile

Returns mask boolean array corresponding to rasterized polygons, of same shape as geoimg.img

pybob.image_tools.**hillshade**(*dem*, *azimuth=315*, *altitude=45*)
Create a hillshade image of a DEM, given the azimuth and altitude.

Parameters

- **dem** (*pybob.GeoImg*) – GeoImg representing a DEM.
- **azimuth** (*float*) – Solar azimuth angle, in degress from North. Default 315.
- **altitude** (*float*) – Solar altitude angle, in degrees from horizon. Default 45.

Returns shade numpy array of the same size as dem.img, representing the hillshade image.

pybob.image_tools.**nanmedian_filter**(*img*, ***kwargs*)
Calculate a multi-dimensional median filter that respects NaN values and masked arrays.

Parameters

- **img** (*array-like*) – image on which to calculate the median filter
- **kwargs** – additional arguments to ndimage.generic_filter Note that either size or footprint must be defined. size gives the shape that is taken from the input array, at every element position, to define the input to the filter function. footprint is a boolean array that specifies (implicitly) a shape, but also which of the elements within this shape will get passed to the filter function. Thus size=(n,m) is equivalent to footprint=np.ones((n,m)). We adjust size to the number of dimensions of the input array, so that, if the input array is shape (10,10,10), and size is 2, then the actual size used is (2,2,2).

Returns filtered Filtered array of same shape as input.

pybob.image_tools.**rasterize_polygons**(*geoimg*, *shapefile*, *burn_handle=None*, *dtype=<MagicMock name='mock.GDT_Int16' id='140009480504264'>*)
Create rasterized polygons given a GeoImg and a shapefile. Useful for creating an index raster with corresponding to polygon IDs.

Parameters

- **geoimg** (*pybob.GeoImg*) – input GeoImg to pull spatial extents from.
- **shapefile** (*str*) – path to polygon shapefile
- **burn_handle** (*str*) – field to pull values to rasterize. Default looks for the FID field in the shapefile.

- **dtype** (*gdal.GDT*) – gdal datatype of rasterized layer. Default is gdal.GDT_Int16.

Returns rasterized, inds rasterized polygon array and values corresponding to polygons that were rasterized.

1.9 pybob.plot_tools

pybob.plot_tools.**plot_chips_corr_matrix**(*srcimg*, *destimg*, *corrmat*)

Parameters

- **srcimg** –
- **destimg** –
- **corrmat** –

Returns

pybob.plot_tools.**plot_ddem_results**(*img*, *colormap='seismic'*, *caxsize='2.5%'*, *clim=None*, *sfact=None*)

Parameters

- **img** –
- **colormap** –
- **caxsize** –
- **clim** –
- **sfact** –

Returns

pybob.plot_tools.**plot_dh_elevation**(*dDEM*, *DEM*, *glacier_mask=None*, *binning=None*, *bin_width=50*, *polyorder=None*)

Parameters

- **dDEM** –
- **DEM** –
- **glacier_mask** –
- **binning** –
- **bin_width** –
- **polyorder** –

Returns

pybob.plot_tools.**plot_geoimg_sidebyside**(*img1*, *img2*, *com_extent=None*, *fig=None*, *cmap='gray'*, *output_directory='.'*, *filename=None*)

Parameters

- **img1** –
- **img2** –
- **com_extent** –

- **fig** –
- **cmap** –
- **output_directory** –
- **filename** –

Returns

pybob.plot_tools.**plot_nice_histogram**(*data_values*, *outfilename*, *output_directory*)

Parameters

- **data_values** –
- **outfilename** –
- **output_directory** –

Returns

pybob.plot_tools.**plot_polygon_df**(*polygon_df*, *fig=None*, *ax=None*, *mpl='mpl_polygon'*, ***kwargs*)

Plot a GeoDataFrame of polygons to figure.

Parameters

- **polygon_df** (*geopandas.GeoDataFrame*) – GeoDataFrame of polygon(s) to plot.
- **fig** (*matplotlib.pyplot.Figure*) – Optional existing figure to plot polygons to. If unset, creates a new figure.
- **ax** (*matplotlib.axes*) – Optional axis handle to plot polygons to. If unset, uses fig.gca()
- **mpl** (*str*) – GeoDataFrame column name containing multipolygon indices. Default is mpl_polygon.
- **kwargs** – Keyword options to pass to matplotlib.collections.PatchCollection.

Returns fig, polygon_df Figure handle of the plot created, and updated geodataframe with plot geometries.

Examples:

```
>>> rgi = gpd.read_file('07_rgi60_Svalbard.shp')
>>> f, rgi = plot_polygon_df(rgi, color='w', edgecolor='k', lw=.2, alpha=0.5)
```

pybob.plot_tools.**plot_shaded_dem**(*dem*, *azimuth=315*, *altitude=45*, *fig=None*, *extent=None*, *alpha=0.35*, *colormap='terrain'*, ***kwargs*)

Plot a shaded relief image of a DEM.

Parameters

- **dem** (*pybob.GeoImg*) – GeoImg representing a DEM.
- **azimuth** (*float*) – Solar azimuth angle, in degress from North. Default 315.
- **altitude** (*float*) – Solar altitude angle, in degrees from horizon. Default 45.
- **fig** (*matplotlib.figure.Figure*) – Figure to show image in. If not set, creates a new figure.
- **extent** (*array-like*) – Spatial extent to limit the figure to, given as xmin, xmax, ymin, ymax.
- **alpha** (*float*) – Alpha value to set DEM to. Default is 0.35.

- **colormap** (*str*) – colormap style for matplotlib
- **kwargs** – Optional keyword arguments to pass to plt.imshow

Returns fig Handle pointing to the matplotlib Figure created (or passed to display).

pybob.plot_tools.**save_results_quicklook**(*img*, *raster*, *com_ext*, *outfilename*, *vmin=0*, *vmax=10*, *sfact=2*, *output_directory='.'*)

Parameters

- **img** –
- **raster** –
- **com_ext** –
- **outfilename** –
- **vmin** –
- **vmax** –
- **sfact** –
- **output_directory** –

Returns

pybob.plot_tools.**set_pretty_fonts**(*font_size=24*, *legend_size=16*)

sets matplotlib fonts to be nice and pretty for graphs that don't completely suck.

pybob.plot_tools.**truncate_colormap**(*cmap*, *minval=0*, *maxval=1*, *n=100*)

Parameters

- **cmap** –
- **minval** –
- **maxval** –
- **n** –

Returns

CHAPTER 2

scripts

scripts

2.1 calculate_mmaster_dh_curves.py

Calculate dH curves from MMASTER dH images, based on shapefiles of glacier outlines.

```
usage: calculate_mmaster_dh_curves.py [-h] [--glac_mask GLAC_MASK]
                                      [--outlier OUTLIER]
                                      [--pct_comp PCT_COMP]
                                      [--namefield NAMEFIELD]
                                      [--out_folder OUT_FOLDER]
                                      [--plot_curves]
                                      dH_folder basedem glac_outlines
```

2.1.1 Positional Arguments

dH_folder	Path to folder with dH images.
basedem	Path to base DEM to use.
glac_outlines	Shapefile of glacier outlines

2.1.2 Named Arguments

--glac_mask	(optional) raster of glacier outlines with unique values for each glacier
--outlier	Value to use as outlier [None]
--pct_comp	Coverage of glacier elevation range curve must cover to be included [0.67] Default: 0.67

--namefield Field with identifying glacier name [RGIId]

Default: "RGIId"

--out_folder Path to write csv files [.]

Default: "."

--plot_curves Plot curves of dH(z) for each dDEM for each glacier [False]

Default: "."

2.2 dem_coregistration.py

Iteratively calculate co-registration parameters for two DEMs, as seen in Nuth and Kääb (2011).

```
usage: dem_coregistration.py [-h] [-a MASK1] [-b MASK2] [-o OUTDIR] [-i] [-f]
                             [-g ALG]
                             masterdem slavedem
```

2.2.1 Positional Arguments

masterdem path to master DEM to be used for co-registration

slavedem path to slave DEM to be co-registered

2.2.2 Named Arguments

-a, --mask1 Glacier mask. Areas inside of this shapefile will not be used for coregistration [None]

-b, --mask2 Land mask. Areas outside of this mask (i.e., water) will not be used for coregistration. [None]

-o, --outdir Directory to output files to (creates if not already present). [.]

Default: "."

-i, --icesat Process assuming that master DEM is ICESat data [False].

Default: False

-f, --full_ext Write full extent of master DEM and shifted slave DEM. [False].

Default: False

-g, --alg Algorithm to calculate slope, aspect. One of 'ZevenbergenThorne' or 'Horn'. [Horn]

Default: "Horn"

2.3 dem_coregistration_grid.py

Iteratively calculate co-registration parameters for sub-grids of two DEMs, as seen in Nuth and Kääb (2011).

```
usage: dem_coregistration_grid.py [-h] [-a MASK1] [-b MASK2] [-s MYSIZE]
                                  [-o OUTDIR] [-i] [-f]
                                  masterdem slavedem
```

2.3.1 Positional Arguments

masterdem path to master DEM to be used for co-registration

slavedem path to slave DEM to be co-registered

2.3.2 Named Arguments

-a, --mask1 Glacier mask. Areas inside of this shapefile will not be used for coregistration [None]

-b, --mask2 Land mask. Areas outside of this mask (i.e., water) will not be used for coregistration. [None]

-s, --mysize determines size of individual blocks in coordinate system units [30000]

Default: 30000

-o, --outdir Directory to output files to (creates if not already present). [.]

Default: "."

-i, --icesat Process assuming that master DEM is ICESat data [False].

Default: False

-f, --full_ext Write full extent of master DEM and shifted slave DEM. [False].

Default: False

2.4 dem_difference.py

Difference co-registered DEM pairs, write metadata file with information.

```
usage: dem_difference.py [-h] [--folder FOLDER] [-mask MASK] [-slope SLOPE]
                         [-s_outlier S_OUTLIER] [-outlier OUTLIER]
                         [-o OUTFILE]
                         DEM1 DEM2
```

2.4.1 Positional Arguments

DEM1 Path to DEM 1

DEM2 Path to DEM 2

2.4.2 Named Arguments

--folder Folder with two co-registered DEMs.

-mask Glacier mask (optional)

-slope Terrain slope (optional)

-s_outlier Set differences above/below to NaN to calculate statistics only.

Default: 100

-outlier Set differences above/below to NaN

-o, --outfile Specify output filename

2.5 extract_ICESat.py

Extracts ICESat data to the extent of a DEM or Image (GeoImg)

```
usage: extract_ICESat.py [-h] [-d WORKDIR] [-o OUTFILE] DEM
```

2.5.1 Positional Arguments

DEM path to DEM from which the extent is used to extract ICESat data

2.5.2 Named Arguments

-d, --workdir Current working directory. [default=./]

-o, --outfile Output filename. [default=ICESat_DEM.h5]

2.6 find_aster_dem_pairs.py

Find ASTER dDEM pairs based on area overlap, time separation.

```
usage: find_aster_dem_pairs.py [-h] [--overlap OVERLAP] [--tmin_sep TMIN_SEP]
                               [--tmax_sep TMAX_SEP] [--imagename IMAGENAME]
                               [--datefield DATEFIELD]
                               footprints
```

2.6.1 Positional Arguments

footprints Shapefile of image footprints to read in.

2.6.2 Named Arguments

--overlap Amount of fractional area that should overlap to use as candidate pair. [default: 0.75]

Default: 0.75

--tmin_sep	Minimum amount of time (in years) to separate images [default: 2 years]
	Default: 2
--tmax_sep	Maximum amount of time (in years) to separate images [default: 1000 years]
	Default: 1000
--imagename	Name of the shapefile field that contains the DEM filenames
	Default: "filename"
--datefield	Name of the shapefield field that contains the date information [None]

2.7 generate_panchromatic.py

Generate simulated panchromatic image for Landsat TM data.

```
usage: generate_panchromatic.py [-h] [-o OUTPUTSCENE] inputscene
```

2.7.1 Positional Arguments

inputscene	Base Landsat scene name (do not specify bands) to be read in.

2.7.2 Named Arguments

-o, --outputscene	Output scene name (if unspecified, defaults to inputscenename_B8.TIF)

2.8 image_footprint.py

Create footprint of valid image area for one (or more) images.

```
usage: image_footprint.py [-h] [-o OUTSHAPE] [-b BUFFER] [--chop [CHOP]]
                          image [image ...]
```

2.8.1 Positional Arguments

image	Image(s) to read in

2.8.2 Named Arguments

-o, --outshape	Shapefile to be written. [default: Footprints.shp]
	Default: "Footprints.shp"
-b, --buffer	buffer size to use [0]
	Default: 0
--chop	Amount of image to crop in m [default 1000]

2.9 image_footprint_from_met.py

Create footprint of valid image area for one (or more) images using the .zip.met file downloaded from earthdata.nasa.gov

```
usage: image_footprint_from_met.py [-h] [-o OUTSHAPE] [-b BUFFER] [-p]
                                   [-t_srs OUT_SRS]
```

2.9.1 Named Arguments

-o, --outshape Shapefile to be written. [default: Footprints.shp]

Default: "Footprints.shp"

-b, --buffer buffer size to use [0]

Default: 0

-p, --processed Use if images have been processed (i.e., there are folders with met filesin them.)

Default: False

-t_srs, --out_srs EPSG code for output spatial reference system [defaults to WGS84 Lat/Lon]

2.10 write_qgis_meta.py

Parse a QGIS document and write metadata as an XML file for given layer(s).

```
usage: write_qgis_meta.py [-h] [--layer LAYER [LAYER ...]] [--outdir OUTDIR]
                          xmlfile
```

2.10.1 Positional Arguments

xmlfile XML file to open and parse

2.10.2 Named Arguments

--layer layer(s) to write metadata for. If left empty, writes metadata for each layer.

--outdir directory to write metadata to. If left empty, writes metadata to same directory as original layer.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

Symbols

A

B

C

D

E

F

G

H

I

J

M

N

O

P

R

S

T

U

W

X