lidar Documentation

Release 0.7.1

Qiusheng Wu

Sep 27, 2022

Contents:

CHAPTER 1

Installation

1.1 Stable release

To install lidar, run this command in your terminal:

```
$ pip install lidar
```

This is the preferred method to install lidar, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

1.2 From sources

The sources for lidar can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/giswqs/lidar
```

Or download the tarball:

```
$ curl  -OL https://github.com/giswqs/lidar/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 2

Usage

To use lidar in a project:

```
import os
import pkg_resources
import lidar
import richdem as rd

# identify the sample data directory of the package
package_name = 'lidar'
data_dir = pkg_resources.resource_filename(package_name, 'data/')

# use the sample dem. Change it to your own dem if needed
in_dem = os.path.join(data_dir, 'dem.tif')
# set output directory. By default, use the temp directory under user's home directory
out_dir = os.path.join(os.path.expanduser("~"), "temp")

# parameters for identifying sinks and delineating nested depressions
min_size = 1000             # minimum number of pixels as a depression
min_depth = 0.3             # minimum depth as a depression
interval = 0.3      # slicing interval for the level-set method
bool_shp = False      # output shapefiles for each individual level

# extracting sinks based on user-defined minimum depression size
sink_path = lidar.ExtractSinks(in_dem, min_size, out_dir)
dep_id_path, dep_level_path = lidar.DelineateDepressions(sink_path, min_size, min_
↪depth, interval, out_dir, bool_shp)

# loading data and results
dem = rd.LoadGDAL(in_dem)
sink = rd.LoadGDAL(sink_path)
dep_id = rd.LoadGDAL(dep_id_path)
dep_level = rd.LoadGDAL(dep_level_path)

# plotting results
```

(continues on next page)

(continued from previous page)

```
dem_fig = rd.rdShow(dem, ignore_colours=[0], axes=False, cmap='jet', figsize=(6, 5.5))
sink_fig = rd.rdShow(sink, ignore_colours=[0], axes=False, cmap='jet', figsize=(6, 5.
↪5))
dep_id_fig = rd.rdShow(dep_id, ignore_colours=[0], axes=False, cmap='jet', figsize=(6,
↪ 5.5))
dep_level_path = rd.rdShow(dep_level, ignore_colours=[0], axes=False, cmap='jet',␣
↪figsize=(6, 5.5))
```

Check the example.py for more details.

CHAPTER 3

Modules

3.1 lidar package

3.1.1 Submodules

3.1.2 filtering module

Module for applying filters to image.

lidar.filtering.**GaussianFilter**(*in_dem*, *sigma=1*, *out_file=None*)

Applies a Gaussian filter to an image.

Parameters

- **in_dem** (*str*) – File path to the input image.
- **sigma** (*int, optional*) – Standard deviation. Defaults to 1.
- **out_file** (*str, optional*) – File path to the output image. Defaults to None.

Returns The numpy array containing the filtered image.

Return type np.array

lidar.filtering.**MeanFilter**(*in_dem*, *kernel_size=3*, *out_file=None*)

Applies a mean filter to an image.

Parameters

- **in_dem** (*str*) – File path to the input image.
- **kernel_size** (*int, optional*) – The size of the moving window. Defaults to 3.
- **out_file** (*str, optional*) – File path to the output image. Defaults to None.

Returns The numpy array containing the filtered image.

Return type np.array

lidar.filtering.**MedianFilter**(*in_dem*, *kernel_size=3*, *out_file=None*)
Applies a median filter to an image.

Parameters

- **in_dem** (*str*) – File path to the input image.
- **kernel_size** (*int, optional*) – The size of the moving window. Defaults to 3.
- **out_file** (*str, optional*) – File path to the output image. Defaults to None.

Returns The numpy array containing the filtered image.

Return type np.array

lidar.filtering.**np2rdarray**(*in_array*, *no_data*, *projection*, *geotransform*)
Converts an numpy array to rdarray.

Parameters

- **in_array** (*np.array*) – The input numpy array.
- **no_data** (*float*) – The no_data value of the array.
- **projection** (*str*) – The projection of the image.
- **geotransform** (*str*) – The geotransform of the image.

Returns The richDEM array.

Return type object

3.1.3 filling module

Module for filling surface depressions.

class lidar.filling.**Depression**(*id*, *count*, *size*, *volume*, *meanDepth*, *maxDepth*, *minElev*, *bndElev*, *perimeter*, *major_axis*, *minor_axis*, *elongatedness*, *eccentricity*, *orientation*, *area_bbox_ratio*)
Bases: object

The class for storing depression info.

lidar.filling.**ExtractSinks**(*in_dem*, *min_size*, *out_dir*, *filled_dem=None*)
Extract sinks (e.g., maximum depression extent) from a DEM.

Parameters

- **in_dem** (*str*) – File path to the input DEM.
- **min_size** (*int*) – The minimum number of pixels to be considered as a sink.
- **out_dir** (*str*) – File path to the output directory.
- **fill_dem** (*str, optional*) – The filled DEM.

Returns The richDEM array containing sinks.

Return type object

lidar.filling.**extract_sinks_by_bbox**(*bbox*, *filename*, *min_size=10*, *tmp_dir=None*, *crs='EPSG:5070'*, *kernel_size=3*, *resolution=10*, *keep_files=True*, *ignore_warnings=True*)
Extract sinks from a DEM by HUC8.

Parameters

- **bbox** (*list*) – The bounding box in the form of [minx, miny, maxx, maxy].
- **filename** (*str, optional*) – The output depression file name.
- **min_size** (*int, optional*) – The minimum number of pixels to be considered as a sink. Defaults to 10.
- **tmp_dir** (*str, optional*) – The temporary directory. Defaults to None, e.g., using the current directory.
- **crs** (*str, optional*) – The coordinate reference system. Defaults to "EPSG:5070".
- **kernel_size** (*int, optional*) – The kernel size for smoothing the DEM. Defaults to 3.
- **resolution** (*int, optional*) – The resolution of the DEM. Defaults to 10.
- **keep_files** (*bool, optional*) – Whether to keep the intermediate files. Defaults to True.
- **ignore_warnings** (*bool, optional*) – Whether to ignore warnings. Defaults to True.

lidar.filling.**extract_sinks_by_huc8**(*huc8*, *min_size=10*, *filename=None*, *tmp_dir=None*, *wbd=None*, *crs='EPSG:5070'*, *kernel_size=3*, *resolution=10*, *keep_files=True*, *error_file=None*, *ignore_warnings=True*)

Extract sinks from a DEM by HUC8.

Parameters

- **huc8** (*str*) – The HUC8 code, e.g., 01010002
- **min_size** (*int, optional*) – The minimum number of pixels to be considered as a sink. Defaults to 10.
- **filename** (*str, optional*) – The output depression file name. Defaults to None, e,g., using the HUC8 code.
- **tmp_dir** (*str, optional*) – The temporary directory. Defaults to None, e.g., using the current directory.
- **wbd** (*str | GeoDataFrame, optional*) – The watershed boundary file. Defaults to None.
- **crs** (*str, optional*) – The coordinate reference system. Defaults to "EPSG:5070".
- **kernel_size** (*int, optional*) – The kernel size for smoothing the DEM. Defaults to 3.
- **resolution** (*int, optional*) – The resolution of the DEM. Defaults to 10.
- **keep_files** (*bool, optional*) – Whether to keep the intermediate files. Defaults to True.
- **error_file** (*_type_, optional*) – The file to save the error IDs. Defaults to None.
- **ignore_warnings** (*bool, optional*) – Whether to ignore warnings. Defaults to True.

lidar.filling.**extract_sinks_by_huc8_batch**(*huc_ids=None*, *min_size=10*, *out_dir=None*, *tmp_dir=None*, *wbd=None*, *crs='EPSG:5070'*, *kernel_size=3*, *resolution=10*, *keep_files=False*, *reverse=False*, *error_file=None*, *ignore_warnings=True*, *ignored_ids=[]*, *overwrite=False*)

Extract sinks from NED by HUC8.

Parameters

- **huc8** (*str*) – The HUC8 code, e.g., 01010002
- **min_size** (*int, optional*) – The minimum number of pixels to be considered as a sink. Defaults to 10.
- **filename** (*str, optional*) – The output depression file name. Defaults to None, e,g., using the HUC8 code.
- **tmp_dir** (*str, optional*) – The temporary directory. Defaults to None, e.g., using the current directory.
- **wbd** (*str | GeoDataFrame, optional*) – The watershed boundary file. Defaults to None.
- **crs** (*str, optional*) – The coordinate reference system. Defaults to "EPSG:5070".
- **kernel_size** (*int, optional*) – The kernel size for smoothing the DEM. Defaults to 3.
- **resolution** (*int, optional*) – The resolution of the DEM. Defaults to 10.
- **keep_files** (*bool, optional*) – Whether to keep the intermediate files. Defaults to True.
- **reverse** (*bool, optional*) – Whether to reverse the HUC8 list. Defaults to False.
- **error_file** (*_type_, optional*) – The file to save the error IDs. Defaults to None.
- **ignore_warnings** (*bool, optional*) – Whether to ignore warnings. Defaults to True.
- **overwrite** (*bool, optional*) – Whether to overwrite the existing files. Defaults to False.

lidar.filling.**get_dep_props**(*objects*, *resolution*)

Computes depression attributes.

Parameters

- **objects** (*object*) – The labeled objects.
- **resolution** (*float*) – The spatial reoslution of the image.

Returns A list of depression objects with attributes.

Return type list

lidar.filling.**np2rdarray**(*in_array*, *no_data*, *projection*, *geotransform*)

Converts an numpy array to rdarray.

Parameters

- **in_array** (*array*) – The input numpy array.
- **no_data** (*float*) – The no_data value of the array.
- **projection** (*str*) – The projection of the image.

- **geotransform** (*str*) – The geotransform of the image.

Returns The richDEM array.

Return type object

lidar.filling.**polygonize**(*img*, *shp_path*)
Converts a raster image to vector.

Parameters

- **img** (*str*) – File path to the input image.
- **shp_path** (*str*) – File path to the output shapefile.

lidar.filling.**regionGroup**(*img_array*, *min_size*, *no_data*)
Identifies regions based on region growing method

Parameters

- **img_array** (*array*) – The numpy array containing the image.
- **min_size** (*int*) – The minimum number of pixels to be considered as a depression.
- **no_data** (*float*) – The no_data value of the image.

Returns The labelled objects and total number of labels.

Return type tuple

lidar.filling.**write_dep_csv**(*dep_list*, *csv_file*)
Saves the depression list info to a CSV file.

Parameters

- **dep_list** (*list*) – A list of depression objects with attributes.
- **csv_file** (*str*) – File path to the output CSV file.

3.1.4 slicing module

Module for the level-set algorithm.

lidar.slicing.**DelineateDepressions**(*in_sink*, *min_size*, *min_depth*, *interval*, *out_dir*, *bool_level_shp=False*)
Delineates nested depressions.

Parameters

- **in_sink** (*str*) – The file path to the sink image.
- **min_size** (*int*) – The minimum number of pixels to be considered as a depression.
- **min_depth** (*float*) – The minimum depth to be considered as a depression.
- **interval** (*float*) – The slicing interval.
- **out_dir** (*str*) – The file path to the output directory.
- **bool_level_shp** (*bool, optional*) – Whether to generate shapefiles for each individual level. Defaults to False.

Returns The output level image, and the output object image.

Return type tuple

class lidar.slicing.**Depression**(*id*, *level*, *count*, *size*, *volume*, *meanDepth*, *maxDepth*, *minElev*, *bndElev*, *inNbrId*, *regionId*, *perimeter*, *major_axis*, *minor_axis*, *elongatedness*, *eccentricity*, *orientation*, *area_bbox_ratio*)

Bases: `object`

The class for storing depression info.

lidar.slicing.**extract_levels**(*level_img*, *obj_img*, *min_size*, *no_data*, *out_img_dir*, *out_shp_dir*, *template*, *bool_comb=False*)

Extracts individual level image.

Parameters

- **level_img** (*np.array*) – The numpy array containing the level image.
- **obj_img** (*np.array*) – The numpy array containing the object image.
- **min_size** (*int*) – The minimum number of pixels to be considered as a depression.
- **no_data** (*float*) – The no_data value of the image.
- **out_img_dir** (*str*) – The output image directory.
- **out_shp_dir** (*str*) – The output shapefile directory.
- **template** (*str*) – The file path to the template image.
- **bool_comb** (*bool, optional*) – Whether to extract combined level image. Defaults to False.

Returns The single level image, properties of region grouped level image, properties of region grouped object image.

Return type tuple

lidar.slicing.**getMetadata**(*img*)

Gets rdarray metadata.

Parameters **img** (*rdarray*) – The richDEM array containing the image.

Returns no_data, projection, geotransform, cell_size

Return type tuple

lidar.slicing.**get_image_paras**(*image_paras*)

Gets image parameters.

Parameters **image_paras** (*dict*) – The dictionary containing image parameters.

Returns A tuple containing no_data, min_size, min_depth, interval, resolution.

Return type tuple

lidar.slicing.**get_min_max_nodata**(*image*)

Gets the minimum, maximum, and no_data value of a numpy array.

Parameters **image** (*np.array*) – The numpy array containing the image.

Returns The minimum, maximum, and no_data value.

Return type tuple

lidar.slicing.**img_to_shp**(*in_img_dir*, *out_shp_dir*)

Converts images in a selected folder to shapefiles

Parameters

- **in_img_dir** (*str*) – The input iimage directory.

- **out_shp_dir** (*str*) – The output shapefile directory.

lidar.slicing.**levelSet**(*img*, *region_id*, *obj_uid*, *image_paras*)
Identifies nested depressions using level-set method.

Parameters

- **img** (*np.array*) – The numpy array containing the image.
- **region_id** (*int*) – The unique id of the region.
- **obj_uid** (*int*) – The object id of the region.
- **image_paras** (*dict*) – The dictionary containing image parameters.

Returns (level image, depression list)

Return type tuple

lidar.slicing.**np2rdarray**(*in_array*, *no_data*, *projection*, *geotransform*)
Converts numpy array to rdarray.

Parameters

- **in_array** (*np.array*) – The input numpy array containing the image.
- **no_data** (*float*) – The no_data value of the image.
- **projection** (*str*) – The projection coordinate system of the image.
- **geotransform** (*str*) – The geotransform of the image.

Returns The richDEM array containing the image.

Return type rdarray

lidar.slicing.**obj_to_level**(*obj_img*, *dep_list*)
Derives depression level image based on the depression id image and depression list.

Parameters

- **obj_img** (*np.array*) – The numpy array containing the object image.
- **dep_list** (*list*) – A list containing depression info.

Returns The numpy array containing the object level image.

Return type np.array

lidar.slicing.**polygonize**(*img*, *shp_path*)
Converts a raster image to vector.

Parameters

- **img** (*str*) – File path to the input image.
- **shp_path** (*str*) – File path to the output shapefile.

lidar.slicing.**regionGroup**(*img_array*, *min_size*, *no_data*)
IdentifIies regions based on region growing method

Parameters

- **img_array** (*np.array*) – The numpy array containing the image.
- **min_size** (*int*) – The minimum number of pixels to be considered as a depression.
- **no_data** (*float*) – The no_data value of the image.

Returns The labelled objects and total number of labels.

Return type tuple

lidar.slicing.**set_image_paras**(*no_data*, *min_size*, *min_depth*, *interval*, *resolution*)
Sets the input image parameters for level-set method.

Parameters

- **no_data** (*float*) – The no_data value of the input DEM.
- **min_size** (*int*) – The minimum number of pixels to be considered as a depression.
- **min_depth** (*float*) – The minimum depth to be considered as a depression.
- **interval** (*float*) – The slicing interval.
- **resolution** (*float*) – The spatial resolution of the DEM.

Returns A dictionary containing image parameters.

Return type dict

lidar.slicing.**updateLevel**(*dep_list*, *obj_uid*)
Updates the inner neighbors of each depression.

Parameters

- **dep_list** (*list*) – A list containing depression info.
- **obj_uid** (*int*) – The unique id of an object.

Returns A list containing depression info.

Return type list

lidar.slicing.**writeObject**(*img_array*, *obj_array*, *bbox*)
Writes depression objects to the original image.

Parameters

- **img_array** (*np.array*) – The output image array.
- **obj_array** (*np.array*) – The numpy array containing depression objects.
- **bbox** (*list*) – The bounding box of the depression object.

Returns The numpy array containing the depression objects.

Return type np.array

lidar.slicing.**writeRaster**(*arr*, *out_path*, *template*)
Saves an numpy array as a GeoTIFF.

Parameters

- **arr** (*np.array*) – The numpy array containing the image.
- **out_path** (*str*) – The file path to the output GeoTIFF.
- **template** (*str*) – The file path to the template image containing projection info.

Returns The numpy array containing the image.

Return type np.array

lidar.slicing.**write_dep_csv**(*dep_list*, *csv_file*)
Saves the depression list to a CSV file.

Parameters

- **dep_list** (*list*) – A list containing depression info.

- **csv_file** (*str*) – File path to the output CSV file.

3.1.5 mounts module

Module for delineating the nested hierarchy of elevated features (i.e., mounts).

lidar.mounts.**DelineateMounts**(*in_dem*, *min_size*, *min_height*, *interval*, *out_dir*, *bool_shp=False*)
Delineates the nested hierarchy of elevated features (i.e., mounts).

Parameters

- **in_dem** (*str*) – File path to the input DEM.
- **min_size** (*int*) – The minimum number of pixels to be considered as an object.
- **min_height** (*float*) – The minimum depth of the feature to be considered as an object.
- **interval** (*float*) – The slicing interval.
- **out_dir** (*str*) – The output directory.
- **bool_shp** (*bool, optional*) – Whether to generate shapefiles. Defaults to False.

Returns File paths to the depression ID and level.

Return type tuple

lidar.mounts.**FlipDEM**(*dem*, *delta=100*, *out_file=None*)
Flips the DEM.

Parameters

- **dem** (*np.array*) – The numpy array containing the image.
- **delta** (*int, optional*) – The base value to be added to the flipped DEM. Defaults to 100.
- **out_file** (*str, optional*) – File path to the output image. Defaults to None.

Returns The numpy array containing the flipped DEM.

Return type np.array

lidar.mounts.**get_min_max_nodata**(*dem*)
Gets the minimum, maximum, and no_data value of a numpy array.

Parameters **dem** (*np.array*) – The numpy array containing the image.

Returns The minimum, maximum, and no_data value.

Return type tuple

3.1.6 Module contents

Top-level package for lidar.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

l

Index