lidar Documentation

Qiusheng Wu

Mar 22, 2020

Contents:

CHAPTER 1

lidar

Author: Qiusheng Wu (https://wetlands.io)

lidar is a toolset for terrain and hydrological analysis using digital elevation models (DEMs). It is particularly useful for analyzing high-resolution topographic data, such as DEMs derived from Light Detection and Ranging (LiDAR) data.

- GitHub repo: https://github.com/giswqs/lidar
- Documentation: https://lidar.readthedocs.io.
- PyPI: https://pypi.org/project/lidar/
- Binder: https://gishub.org/lidar-cloud
- Free software: MIT license

Contents

1.1 Features

- Smoothing DEMs using mean, median, and Gaussian filters (see filtering.py)
- Extracting depressions from DEMs (see filling.py).
- Filtering out small artifact depressions based on user-specified minimum depression size (see filling.py).
- Generating refined DEMs with small depressions filled but larger depressions kept intact (see filling.py).
- Delineating depression nested hierarchy using the level-set method (see slicing.py).
- Delineating mount nested hierarchy using the level-set method (see mounts.py).
- Computing topological and geometric properties of depressions, including size, volume, mean depth, maximum depth, lowest elevation, spill elevation, perimeter, major axis length, minor axis length, elongatedness, eccentricity, orientation, and area-bbox-ratio (see slicing.py).
- Exporting depression properties as a csv file (see slicing.py).

1.2 Installation

lidar supports a variety of platforms, including Microsoft Windows, macOS, and Linux operating systems. Note that you will need to have **Python 3.x** installed. Python 2.x is not supported. The **lidar** Python package can be installed using the following command. If you encounter any errors, please check the *Dependencies* section below. The instruction below assumes that you have installed Anaconda. Open **Anaconda Prompt** and enter the following commands to create a conda environment and install required packages.

```
conda create -n pylidar python
conda activate pylidar
conda install -c conda-forge lidar
```

If you have installed **lidar** before and want to upgrade to the latest version, you can use the following command:

```
pip install lidar -U
```

1.3 Tutorials

Launch the interactive notebook tutorial for the **lidar** Python package with **mybinder.org** or **binder.pangeo.io** now:

1.3.1 A Quick Example

```
import os
import pkg_resources
from lidar import *

# 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 = 0.5      # minimum depth as a depression
interval = 0.3       # slicing interval for the level-set method
bool_shp = True     # output shapefiles for each individual level

# extracting sinks based on user-defined minimum depression size
out_dem = os.path.join(out_dir, "median.tif")
in_dem = MedianFilter(in_dem, kernel_size=3, out_file=out_dem)
sink_path = ExtractSinks(in_dem, min_size, out_dir)
dep_id_path, dep_level_path = DelineateDepressions(sink_path, min_size, min_depth,␣
↪interval, out_dir, bool_shp)

print('Results are saved in: {}'.format(out_dir))
```

Check the example.py for more details.

1.3.2 An Interactive Jupyter Notebook Tutorial

This tutorial can be accessed in three ways:

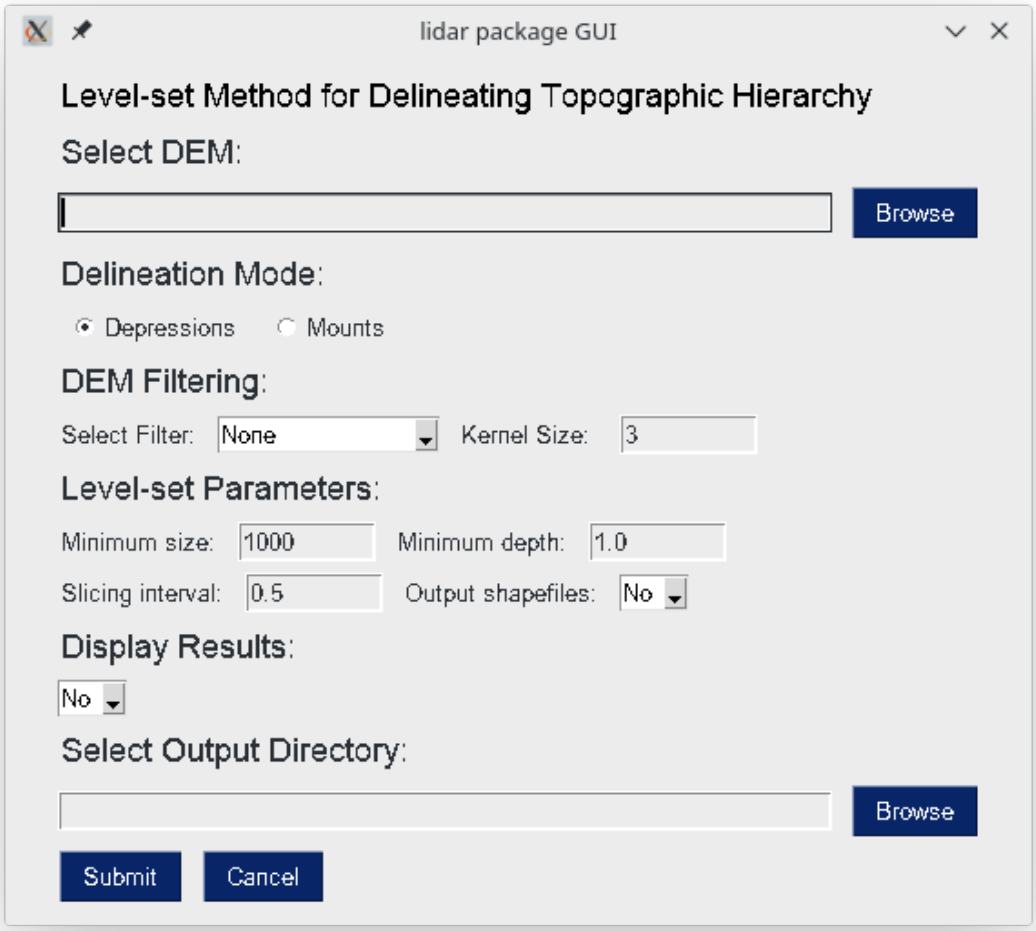
- HTML version: https://gishub.org/lidar-html
- Viewable Notebook: https://gishub.org/lidar-notebook
- Interactive Notebook: https://gishub.org/lidar-cloud

Launch this tutorial as an interactive Jupyter Notebook on the cloud - https://gishub.org/lidar-cloud.

1.4 lidar GUI

lidar also provides a Graphical User Interface (GUI), which can be invoked using the following Python script:

```
import lidar
lidar.gui()
```



1.5 Dependencies

lidar's Python dependencies are listed in its requirements.txt file. In addition, lidar has a C library dependency: GDAL >=1.11.2. How to install GDAL in different operating systems will be explained below. More informaton about GDAL can be found here.

It is highly recommended that you use a Python virtual environment (e.g., conda) to test the lidar package. Please follow the conda user guide to install conda if necessary. Once you have conda installed, you can use Terminal or an Anaconda Prompt to create a Python virtual environment. Check managing Python environment for more information.

Once GDAL has been installed, you can then proceed to install the **lidar** Python package using the following command:

```
conda create -n py37 python=3.7
conda activate py37
```

(continues on next page)

(continued from previous page)

```
conda install -c conda-forge gdal
pip install lidar
```

1.5.1 Linux

Debian-based Linux

The following commands can be used to install GDAL for Debian-based Linux distributions (e.g., Ubuntu, Linux Mint).

```
sudo add-apt-repository ppa:ubuntugis/ppa
sudo apt-get update
sudo apt-get install gdal-bin libgdal-dev
pip install lidar
```

If you encounter any compiling errors, try the following commands.

```
sudo apt-get install --reinstall build-essential
sudo apt-get install python3-dev
pip install wheel
```

Pacman-based Linux

The following commands can be used to install GDAL for Pacman-based Linux distributions (e.g., Arch Linux, Manjaro). You might need to use **sudo** if you encounter permission errors.

```
sudo pacman -S yaourt --noconfirm
yaourt -S gdal --noconfirm
yaourt -S python-gdal --noconfirm
pip install lidar
```

1.5.2 MacOS X

For a Homebrew based Python environment, do the following.

```
brew update
brew install gdal
```

Alternatively, you can install GDAL binaries from kyngchaos. You will then need to add the installed location `/Library/Frameworks/GDAL.framework/Programs` to your system path.

1.5.3 Windows

The instruction below assumes that you have installed Anaconda. Open **Anaconda Prompt** and enter the following commands to create a conda environment and install required packages

```
conda create -n py37 python=3.7
conda activate py37
conda install -c conda-forge gdal
```

(continues on next page)

(continued from previous page)

```
pip install richdem
pip install lidar
```

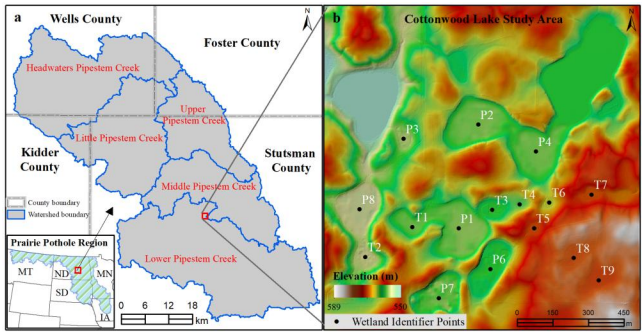
When installing the **richdem** package, if you encounter an error saying 'Microsoft Visual C++ 14.0 is required', please follow the steps below to fix the error and reinstall **richdem**. More infomration can be found at this link Fix Python 3 on Windows error - Microsoft Visual C++ 14.0 is required.

- Download Microsoft Build Tools for Visual Studio 2017
- Double click to install the downloaded installer - **Microsoft Build Tools for Visual Studio 2017**.
- Open **Microsoft Build Tools for Visual Studio 2017**
- Select **Workloads –> Visual C++ build tools** and click the install button

1.6 Examples

The images below show working examples of the level set method for delineating nested depressions in the Cottonwood Lake Study Area (CLSA), North Dakota. More test datasets (e.g., the Pipestem watershed in the Prairie Pothole Region of North Dakota) can be downloaded from http://gishub.org/2018-JAWRA-Data

The following example was conducted on a 64-bit Linux machine with a quad-core Intel i7-7700 CPU and 16 GB RAM. The average running time of the algorithm for this DEM was 0.75 seconds.



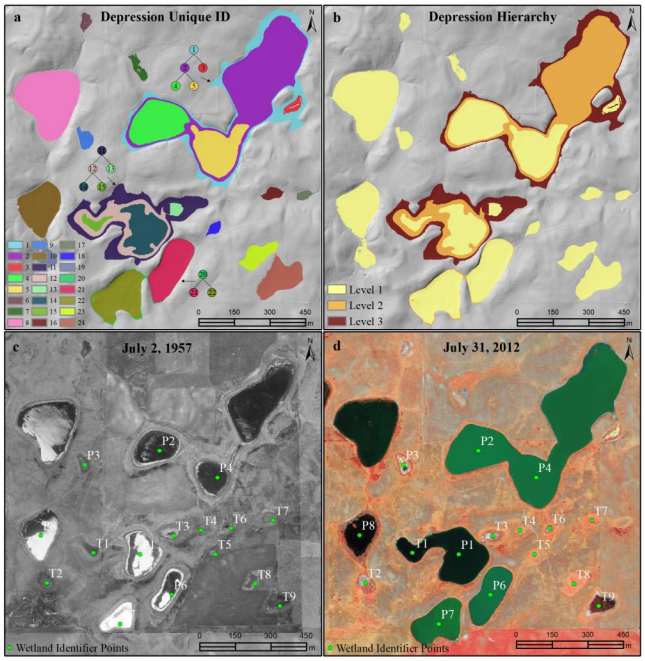

a
Depression Unique ID
b
Depression Hierarchy
Level 1
Level 2
Level 3
c
July 2, 1957
d
July 31, 2012
P1
P2
P3
P4
P6
P7
P8
T1
T2
T3
T4
T5
T6
T7
T8
T9
Wetland Identifier Points
0 150 300 450
m
N

TABLE 1. Characteristics of nested depressions in Cottonwood Lake Study Area

Depression ID	Level	Area (m^2)	Volume (m^3)	Mean depth (m)	Maximum depth (m)	Lowest elevation (m)	Spill elevation (m)	Children IDs	Region ID
1	3	226,232	666,183	2.94	5.16	558.11	563.28	[2, 3]	1
2	2	175,406	218,509	1.25	2.96	558.11	561.08	[4, 5]	1
3	1	2,736	438	0.16	0.31	560.96	561.08		1
4	1	33,901	34,478	1.02	1.76	558.11	559.88		1
5	1	32,710	27,463	0.84	1.44	558.43	559.88		1
6	1	2,548	588	0.23	0.56	556.18	556.74		2
7	1	3,643	527	0.14	0.41	560.91	561.32		3
8	1	56,564	20,290	0.36	0.43	551.32	551.75		4
9	1	4,245	2,326	0.55	1.12	557.05	558.17		5
10	1	27,623	30,946	1.12	2.04	555.00	557.04		6
11	3	98,493	311,711	3.16	5.46	557.93	563.39	[12, 13]	7
12	2	55,592	77,730	1.40	2.46	557.93	560.39	[14, 15]	7
13	1	2,624	863	0.33	0.69	559.70	560.39		7
14	1	25,798	13,677	0.53	1.06	557.93	558.99		7
15	1	3,909	596	0.15	0.43	558.56	558.99		7
16	1	3,336	727	0.22	0.66	569.52	570.18		8
17	1	2,042	337	0.16	0.38	568.32	568.70		9
18	1	2,228	613	0.27	0.80	569.53	570.33		10
19	1	6,277	2,684	0.43	1.09	556.00	557.09		11
20	2	63,461	191,783	3.02	5.69	558.00	563.69	[21, 22]	12
21	1	29,264	65,802	2.25	3.29	560.00	563.29		12
22	1	31,666	101,089	3.19	5.29	558.00	563.29		12
23	1	10,232	4,882	0.48	1.29	577.00	578.30		13
24	1	19,938	23,678	1.19	2.20	576.19	578.39		14

1.7 References

The level-set algorithm in the **lidar** package has been published in the following article:

- **Wu, Q.**, Lane, C.R., Wang, L., Vanderhoof, M.K., Christensen, J.R., & Liu, H. (2019). Efficient Delineation of Nested Depression Hierarchy in Digital Elevation Models for Hydrological Analysis Using Level-Set Method.

Journal of the American Water Resources Association. DOI: 10.1111/1752-1688.12689 (preprint)

Applications of the level-set and contour-tree methods for feature extraction from LiDAR data:

- **Wu, Q.**, & Lane, C.R. (2017). Delineating wetland catchments and modeling hydrologic connectivity using LiDAR data and aerial imagery. *Hydrology and Earth System Sciences*. 21: 3579-3595. DOI: 10.5194/hess-21-3579-2017
- **Wu, Q.**, Deng, C., & Chen, Z. (2016). Automated delineation of karst sinkholes from LiDAR-derived digital elevation models. *Geomorphology*. 266: 1-10. DOI: 10.1016/j.geomorph.2016.05.006
- **Wu, Q.**, Su, H., Sherman, D.J., Liu, H., Wozencraft, J.M., Yu, B., & Chen, Z. (2016). A graph-based approach for assessing storm-induced coastal changes. *International Journal of Remote Sensing*. 37:4854-4873. DOI: 10.1080/01431161.2016.1225180
- **Wu, Q.**, & Lane, C.R. (2016). Delineation and quantification of wetland depressions in the Prairie Pothole Region of North Dakota. *Wetlands*. 36(2):215–227. DOI: 10.1007/s13157-015-0731-6
- **Wu, Q.**, Liu, H., Wang, S., Yu, B., Beck, R., & Hinkel, K. (2015). A localized contour tree method for deriving geometric and topological properties of complex surface depressions based on high-resolution topographic data. *International Journal of Geographical Information Science*. 29(12): 2041-2060. DOI: 10.1080/13658816.2015.1038719
- **Wu, Q.**, Lane, C.R., & Liu, H. (2014). An effective method for detecting potential woodland vernal pools using high-resolution LiDAR data and aerial imagery. *Remote Sensing*. 6(11):11444-11467. DOI: 10.3390/rs61111444

1.8 Reporting Bugs

Report bugs at https://github.com/giswqs/lidar/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

1.9 Credits

- The algorithms are built on richdem, numpy, scipy, scikit-image, and pygdal.
- This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 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.

2.2 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 3

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 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/giswqs/lidar/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

lidar could always use more documentation, whether as part of the official lidar docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/giswqs/lidar/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *lidar* for local development.

1. Fork the *lidar* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/lidar.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv lidar
$ cd lidar/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 lidar tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/giswqs/lidar/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_lidar
```

4.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 5

Credits

5.1 Development Lead

- Qiusheng Wu <giswqs@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.2.0 (2018-09-16)

6.2 0.1.6 (2018-05-21)

6.3 0.1.5 (2018-05-16)

6.4 0.1.3 (2018-05-15)

6.5 0.1.0 (2018-05-14)

CHAPTER 7

Indices and tables

- genindex
- modindex
- search