LiDAR image classification

Author: Maryam Moghadam
Version: 1
Type of Investigation: Noise removal and segmentation on Lidar Data
Goal of lab: This lab will introduce the GatorEye LiDAR database provided by National Forests in Florida (NFF) at the USDA Forest Service. This lab will guide students to open LiDAR image and convert it into DEM (Digital Elevation Model), DSM (Digital Surface Map), and DTM (digital terrain model) for Terrain and hydrological analysis. The lab will then demonstrate different noise removal algorithms and also performs segmentation and slicing to fill the depression sinks on DEM using level set algorithm.
Operating Systems: Students need to have a Windows operating system to install LAStools for image conversion and the code can be ran on Jupyter Notebook on any Operating system.
Hardware: No specific hardware configuration or requirement
Support computer languages: Python
Files/Data/Documents: http://www.speclab.org/gatoreye-data-access.html
Semester: Spring 2020
Date: March 30, 2020

Detail Procedure:

Part1: Downloading data and choosing your sample lidar image

1. Visit http://www.speclab.org/gatoreye-data-access.html and choose the 2019 Florida Apalachicola 110819 dataset
2. Paste "I agree to the GatorEye data usage Terms and Conditions" when it asks for a password
3. From the downloaded dataset choose one .las file for the testing purpose of this project
4. Create your own project directory
5. Name your .las file to lidar.las
6. Paste the chosen lidar.las file into your project directory

Part2: Installation of LAStools on a Windows operating system

1. Note: LAStools only runs on Windows operating system, so you have to create a virtual machine with a Windows operating system on Microsoft AZURE or any other cloud provider if you have another operating system. If you do not have access to it, have difficulties or permission access issues generating these files from LAStools, they are available on my GitHub repository at "https://github.com/MaryamMoghadam/Lidar/raw/master/lidar-dem.zip" for you to download. If you downloaded them directly from my GitHub repository, you can skip all the steps bellow and move to Part3. Otherwise follow the steps 2 to 9 to extract DEM, DSM, and sink from the las files on your own.
2. Visit http://lastools.org to download LAStools on your windows operating system
3. Open the Lastools.zip file and Navigate to the bin folder. Copy las2dem.exe file to your project directory
4. Open the Windows command prompt: Win + X then select Command prompt
5. Navigate to your project directory

6. Type the command: “las2dem -i lidar.las -o dem.tif” to create a digital elevation model, TIN from the LAS file
7. Type the command: “las2dem -i lidar.las -o dsm.tif” to create a digital surface map, TIN from the LAS file
8. Type the command: “las2dem -i lidar.las -o sink.tif” to create a digital terrain model, TIN from the LAS file

Part3: Installing Jupyter Notebook on your local machine

1. Python is a requirement (Python 3.3 or greater, or Python 2.7) for installing the Jupyter Notebook so make sure you have it installed on your local machine otherwise you can install Anaconda which will install Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science automatically on your local machine
2. Download Anaconda from https://www.anaconda.com/distribution/
3. Follow the instructions on popup windows to install it
4. You now have Jupyter Notebook installed
5. You can run Jupyter Notebook on your local machine by running “jupiter notebook” command on your command prompt or terminal windows
6. Create a new python file by pressing the button on the top right side of the Jupyter Notebook window and name your program
7. Try to run a “Hello World!” program in python on your new program and attach a screen shot of your notebook below

Part4: Setting up working environment/directory

1. In the created python file, define a new path and include your working directory using “os.path” module of python
2. Print your created path to make sure you are in the correct working directory. If you downloaded the data from my GitHub and moved it to your project directory directly you can skip to step 6 otherwise follow step 3 to 5 to download it using python modules in your program
3. If you are trying to download the data from my GitHub repository directly into your working directory you can import and use “urlib.request” module to open the URL and download the data from it
4. Since this file is in zip format you can import and use “zipfile” module of python to unzip it.
5. To be able to extract the files from zip and the provided URL and move them to your working directory you can use “shutil” module of python
6. You would then have to add the downloaded files into your project directory so you could use them
7. Now print your project directory to make sure you are in the correct working directory
8. List your data directory to make sure you have all three of dem.tif dsm.tif and sink.tif in your project directory
9. Attach a screenshot of the list of the files included in your directory

10. Now create a new path which will be the directory for you to save your output results
11. Import the python Lidar library

Part5: Level Set Algorithm implementation\Sink extraction

Level Set algorithm is a tool for numerical analysis of surfaces and shapes. It can perform numerical computations that involves curves and surfaces on a fixed Cartesian grid without having to parameterize these objects. It is easy to follow shapes that change topology by time by level set algorithm, for example: shapes that split in two and develop holes could be segmented using level set algorithm.

1. The holes that were described above are called sinks in Terrain and hydrological analysis and you can extract them from your DEM file using "ExtractSinks" function of the pythons Lidar library
2. ExtractSinks function uses a number as the minimum number of the pixels that has to be consider for an area to be considered as a hole/depression/sink
3. So, you would have to define a new variable to give it the minimum number of the pixels for a depression. Based on the published papers on this topic [1] 1000 is a proper value in this type of analysis
4. Now you can create a new image that only has the sinks segmented from your DEM
   Example: sink = lidar.ExtractSinks(in_dem, min_size, out_dir)
5. You can now read this image as a NumPy array using imageio module
6. And plot it using the Matplotlib library
7. Attach a screen shot of your plot in the chart below

Part6: Level Set Algorithm implementation\Depression slicing

1. After defining the width of an area to be considered as depression you should define a minimum depth for an area to be considered a sink, based on the paper that was introduces before the proper minimum depth value could be 0.3
2. Now you have to define a slicing interval for your level set algorithm which could be set to 0.3 as well
3. Now you have to let your function know that you want to consider all of the probabilities for an area to be considered as a depression to avoid only considering Boolean shapes whit the probability of either 0 or 1 or highest or lowest so you have to set bool_shp to false

4. You can now use the DelinateDepressions function from the Lidar library to slice the depressions by also counting for the depth value and attach the segmented sinks with the depth sliced depression image to have a full depression level presentation
   Example: dep_id, dep_level = lidar.DelineateDepressions(sink, min_size, min_depth, interval, out_dir, bool_shp)
5. You can now read these images as a NumPy array using imageio module
6. And plot it using the Matplotlib library
7. Report the Run time statistics of this slicing process in the box bellow

Part7: Noise Removal\Median Filtering

The Median Filter is a non-linear digital filtering technique, often used to remove noise from an image or signal. Median filter allows humans and algorithms to focus attention on important small-scale features in the images. In the case of our lidar data, there are at least two known systematic features in the data that we have reason to eliminate with high pass median filtering: (1) the effects of attenuation and (2) random variations in the laser pulse energy that cause streaks in the unfiltered images. The main idea of the median filter is to run through the signal entry by entry, replacing each entry with the median of neighboring entries. The pattern of neighbors is called the "window", which slides, entry by entry, over the entire signal. The principle of the median filter is to replace the gray level of each pixel by the median of the gray levels in a neighborhood of the pixels, instead of using the average operation. For median filtering, we specify the kernel\Window size, list the pixel values covered by the kernel, and determine the median level. For a better understanding of Median Filter, you can refer to Figure 1 attached to the next page.

1. Define your dem file as your input
2. Define an output directory and name it to what you would like your result to have
3. Use the MedianFilter from the lidar library and use your dem file as your input and define a kernel size of 3 and choose an output directory to save your result
   Example:
   in_dem = os.path.join(data_dir, 'dem.tif')
   out_dem = os.path.join(out_dir, "median.tif")
   in_dem = lidar.MedianFilter(in_dem, kernel_size=3, out_file=out_dem)

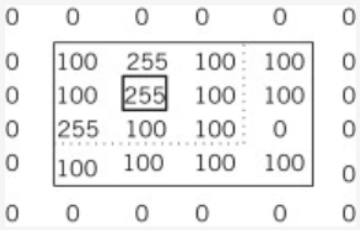
4. You can now read this filtered image as a NumPy array using imageio module
5. And plot it using the Matplotlib library
6. Report the runtime of Median filtering in the box below

Step 1: To process the first element, we cover the 3 × 3 kernel with the center pointing to the first element to be processed. The sorted data within the kernel are listed in terms of thier value as

$$0,\ 0,\ 0,\ 0,\ 0,\ 100,\ 100,\ 255,\ 255$$

The median value = median(0, 0, 0, 0, 0, 100, 100, 255, 255) = 0. Zero will replace 100.

Step 2: Continue for each element until the last is replaced. Let us review the element at location (1,1):



The values covered by the kernel are

$$100,\ 100,\ 100,\ 100,\ 100,\ 100,\ 255,\ 255,\ 255$$

The median value = median(100, 100, 100, 100, 100, 100, 255, 255, 255) = 100. The final processed image is

$$\begin{bmatrix} 0 & 100 & 100 & 0 \\ 100 & 100 & 100 & 100 \\ 0 & 100 & 100 & 0 \\ 100 & 100 & 100 & 100 \end{bmatrix}$$

Some boundary pixels are distorted due to the zero padding effect. However, for a large image, the portion of the boundary pixels (outmost image edges) is significant small so that their distortion can be omitted versus the overall quality of the image. The 2 × 2 middle portion matches the original image exactly. The effectiveness of the median filter is verified via this example.

Figure 1. Median Filter

Part8: Noise Removal\Gaussian Filtering

Gaussian filtering is used to blur images and remove noise and detail. Gaussian filters have the properties of having no overshoot to a step function input while minimizing the rise and fall time. The 2-D Gaussian smoothing filter is a convolution operator that is often used to remove details and noises in images. It has been shown that under some conditions, the only possible scale-space kernel is the Gaussian function, which is useful for feature localization. Furthermore, Gaussian filter provides "weighted" smoothing and preserves edges better than average filter of similar size. The 2-D Gaussian distribution has the form bellow:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where σ is the standard deviation. The distribution of 2-D Gaussian function with σ =1 is illustrated in this project. For a better understanding of Gaussian Filter, you can refer to Figure 2 attached bellow.

1. Define your dem file as your input
2. Define an output directory and name it to what you would like your result to have
3. Use the GaussianFilter from the lidar library and use your dem file as your input and define a sigma of 1 and choose your output directory to save your result
   Example:
   in_dem2 = os.path.join(data_dir, 'dem.tif')
   out_dem2 = os.path.join(out_dir, "gaussian.tif")
   in_dem2 = lidar.GaussianFilter(in_dem, sigma = 1, out_file=out_dem2)

4. You can now read this filtered image as a NumPy array using imageio module
5. And plot it using the Matplotlib library
6. Try changing the Sigma value to something very high like 10 or 15 to see how it effects the image in practice
7. Now try to take a plot of both Median and Gaussian filtering applied to the DEM image to compare them both

8. Which of the filters that you have applied removes the noises better? Write down your observation in the box bellow:

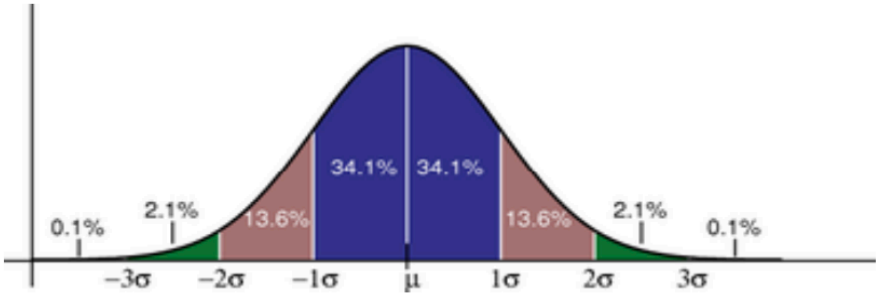
9. Report the runtime of your Gaussian filter in the box bellow:

Standard Deviation

The Standard deviation of the Gaussian function plays an important role in its behaviour.

The values located between +/- σ account for 68% of the set, while two standard deviations from the mean (blue and brown) account for 95%, and three standard deviations (blue, brown and green) account for 99.7%.

This is very important when designing a Gaussian kernel of fixed length.



Distribution of the Gaussian function values (Wikipedia)

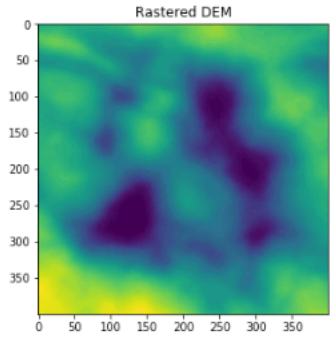
Figure 2. Gaussian Filter

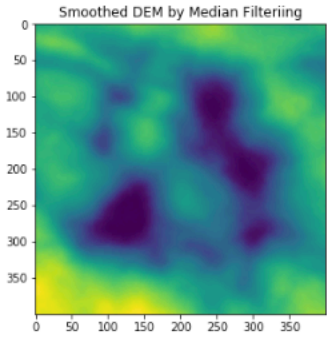
Critical Thinking:

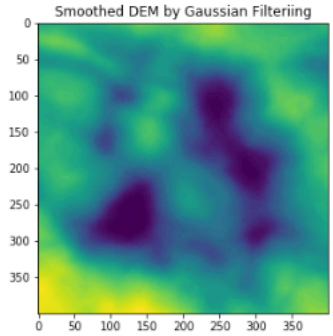
1. Which of the Filtering algorithms do you think fits better in analyzing LiDAR images?
2. What would happen if you count in a larger Kernel size in Median Filtering?
3. What would happen if you lower or higher the sigma value in Gaussian Filter?
4. How do you compare your original image with and without applying smoothing filtering?
5. For which of the filters you had a lower run time Median or Gaussian? What is the reason?
6. How can you plot all of your results in one figure?

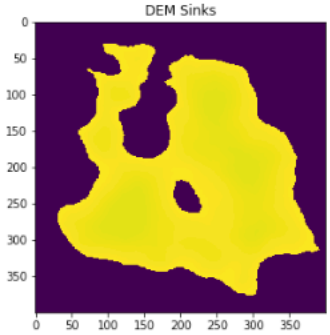
Final Report

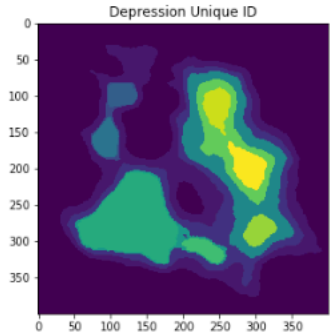
You can find the result for your project for your convenience to compare different filters that you implied. The source code for this project is also attached below.

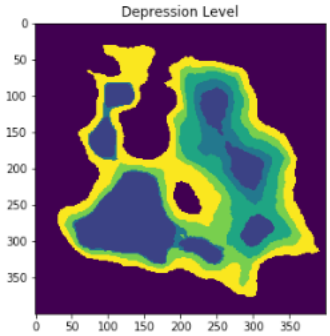












Items Discussed: Jupyter Notebook, LiDAR database, urllib python module, LAStools, os python module, las to dem, las to dsm, las to dtm, Python Lidar library, Extractsinks lidar function, DelineateDepressions lidar function, Median filtering, Gaussian filtering

Useful Resource: You can find the ipynb file of this project on my Github repository "https://github.com/MaryamMoghadam/Lidar/blob/master/My_Lidar.ipynb"