Comparing Computer Resource Usage Through Interpolating G.E.D.I. LiDAR Waveform Data

Andrew C. Baker, Arjun Raj, Sara I. Ansari
Under the mentorship of:
Brandon J. Woodard, Department of Computer Science, Brown University
Shinpei S. Nakamura, Department of Statistics and Data Science, Yale University

Abstract

In our study we aim to compare the computer resources utilised by cubic and linear splines. We will access and use publicly available data collected by Nasa's Global Ecosystem Dynamic Investigation's (GEDI) LiDAR technology to model waveforms using cubic and linear b-spline interpolation. We will compare the viability and efficiency of these two interpolation methods to determine which is more applicable when working with GEDI data, and when working with datasets similar in size, scope, and variability specifically as it relates to artificial intelligence creation. This study will assist researchers working with GEDI data and data like it to make more informed decisions about creating models and the efficiency of those models as it relates to large scale artificial intelligence deployment.

Keywords

Artificial Intelligence, GEDI Data, LiDAR Waveforms, Interpolation, Linear and Cubic Splines, Computer Resources

1. Introduction

The study we propose will compare the computer resource usage of the two interpolation model's ability to accurately and efficiently model the GEDI data. GEDI or NASA's Global Ecosystem Dynamics Investigation was scheduled for a two year deployment on the International Space Station (ISS) in 2018, but the deployment was extended to retrieve more data. GEDI produces high resolution observations of the three dimensional structure of the Earth's topography using its full form LiDAR instrument. LiDAR is a technology that uses laser pulses to measure structures, spaces, and surfaces. Henceforth we will refer to LiDAR as it relates to GEDI data in two instances: beams and shots. Beams are a series of shots that contain thousands of shots. Shots are individual measurements and points where the LiDAR machine took a measure. Hancock et al. (2019) and Simard et al. (2011) provided useful background information and research when we first began to learn about the data, technology, and uses for GEDI as well as LiDAR. The research these groups did have helped to focus our own research and knowledge surrounding this project. GEDI data is of great value for studies about forest and water resource management, carbon life cycle science, and weather prediction.

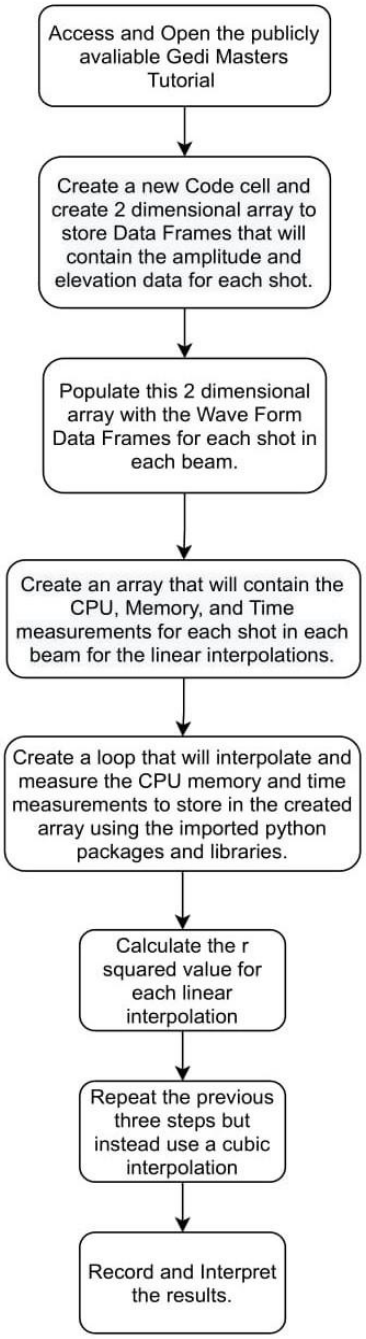
We will be interpolating GEDI data, and using Hancock (2018)'s GEDI Tutorial as well as other publicly available tutorials in our work. We took inspiration from the work done by Chen et al. (2008), but applied their ideas and findings on a much smaller scale and in a slightly different

way. We compared ways of creating and training artificial intelligence systems as they relate to environmental ones instead of testing different artificial intelligence systems on modeling environments. Our process and results also help relate to machine learning because of the intense computer usage and cost related to the training and testing of machine learning. Without weighing the benefits of different interpolations, calculations, or models, companies could easily lose large amounts of money by not properly addressing the value of different methods of training algorithms.

2. Methods

We will be using two functions to interpolate the data we are modeling: cubic and linear. We will be using the cubic and linear functions found in the scipy libraries implemented through python. In the function we are using we use three parameters: One parameter to pass an array of X values that contain the Amplitude of a shot, a second parameter to pass an array of Y values that contain the Elevation of a shot, and the third parameter we passed is a string value which represents the order of the interpolation function. We compared the computer resources utilized by passing cubic and linear as the parameter for the order function. To create the interpolation models we needed data frames of the waveforms for each shot in each beam. We created loops and arrays to create a waveform data frame that could be used to make the interpolation models later on. We extracted all the waveforms and put it into a single array. We created two separate interpolations that model canopy height utilizing GEDI Level 1B data. One model used the linear interpolation function, and another will use the cubic interpolation function. We interpolated each waveform and recorded the following measurements:

1. Time the model takes to generate
    a. Measured in seconds (s).
    b. We will measure the time taken to complete the process of optimizing by using python's time module or the data training library's built-in functions.
    c. We run the interpolation 50 times and take the average for each waveform
2. Memory Usage
    a. Measured in GigaBytes (GB)
    b. We will measure the memory using the os and psutil libraries functions.
3. Accuracy
    a. Measured in an r-squared value



b. The accuracy will be measured using the data training library's built-in functions

Figure 1

We ran the tests on a computer which had the following specifications:
Operating System - Windows 10 Pro 64-bit operating system, x64-based processor
Central Processing Unit - Intel Xeon CPU E5-1620 0 @ 3.60GHz 3.60 GHz (8 Core)
Random Access Memory - 40.0 GB ddr3 1333 MHz RAM
Code language - Python 3.8.7

3. Results

To discuss our results we have compiled an array containing all relevant measurements and results from our project. Here is that array formatted into a dataframe to show some of the results. L.I. refers to Linear Interpolation and C.I. refers to Cubic Interpolation. MSE refers to mean squared error.

	Beam	Shot	L.I. Time (s)	L.I. RAM (bytes)	L.I. MSE	C.I. Time (s)	C.I. RAM (bytes)	C.I. MSE
0	1	1	0.00020	983.04	0.0	0.000180	245.76	4.053185e-23
1	1	2	0.00018	0.00	0.0	0.000120	0.00	4.384057e-23
2	1	3	0.00018	0.00	0.0	0.000140	0.00	3.639595e-23
3	1	4	0.00020	0.00	0.0	0.000148	0.00	3.391441e-23
4	1	5	0.00020	0.00	0.0	0.000140	0.00	2.564260e-23
...	...	...	...	...	...	...	...	...
23675	8	2174	0.00026	0.00	0.0	0.000220	0.00	1.976962e-22
23676	8	2175	0.00028	0.00	0.0	0.000280	163.84	2.067952e-22
23677	8	2176	0.00040	81.92	0.0	0.000420	0.00	2.010049e-22
23678	8	2177	0.00048	0.00	0.0	0.000220	0.00	2.067952e-22
23679	8	2178	0.00028	0.00	0.0	0.000240	0.00	2.332649e-22

23680 rows × 8 columns

Table 1

The totals of our measurements and results' most prudent information is as follows:

A) Totals:
Total shots without exception waveforms: 23680
Total shots with exception waveforms: 35472

B) Linear Interpolation
a) R-Squared Values
Linear Interpolation total r-squared value: 1.7904975090779743e-23
Linear Interpolation average r-squared value: 7.561222589011715e-28

b) RAM usage
Linear Interpolation entire process RAM usage (bytes): 98074624
Linear Interpolation total RAM usage (bytes): 1737850.8799999887
Linear Interpolation average RAM usage per shot (bytes): 73.3889729729725

c) Time Measurements
Linear Interpolation total time (seconds): 6.045272965431242
Linear Interpolation average time per shot (seconds): 0.0002552902434726031

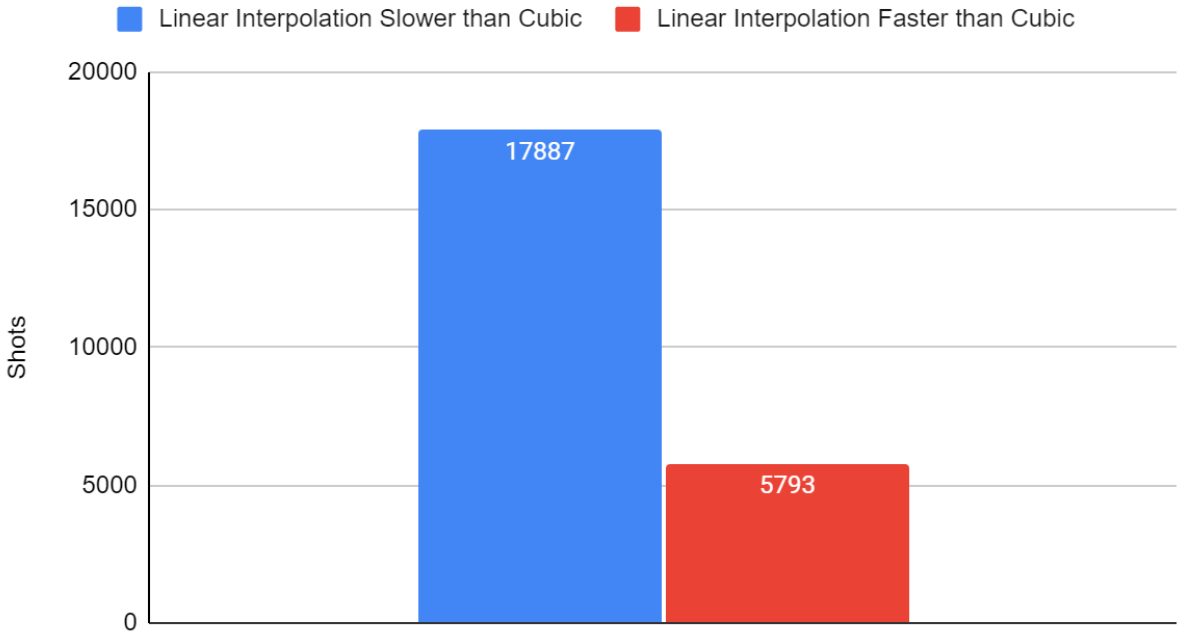
C) Cubic Interpolation

a) R-Squared Values
Cubic Interpolation total r-squared value: 1.3882197041634871e-18
Cubic Interpolation average r-squared value: 5.862414291230942e-23

b) RAM usage
Cubic Interpolation entire process RAM usage (bytes): 108204032
Cubic Interpolation total RAM usage (bytes): 236503.04000000388
Cubic Interpolation average RAM usage per shot (bytes): 9.987459459459624

c) Time Measurements
Cubic Interpolation total time (seconds): 4.896415433883705
Cubic Interpolation average time per shot (seconds): 0.0002067743004173862

Figure 2: Time Comparison Chart



All other information and individual pieces of data that we collected will be available in the appendix of the paper (Section 8)

4. Discussion

We used the `interp1d` class from the `interpolate` package from the `scipy` library to create our interpolations.

```
from scipy.interpolate import interp1d
```

A description of this class:
"x and y are arrays of values used to approximate some function f: y = f(x). This class returns a function whose call method uses interpolation to find the value of new points."
[https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.interp1d.html]

The class takes 8 parameters as follows:

```
class scipy.interpolate.interp1d(x, y, kind='linear',
axis=- 1, copy=True, bounds_error=None, fill_value=nan,
assume_sorted=False)
```

The linear interpolation was created by passing the value `'linear'` to the `kind` parameter, and the cubic interpolation was created by passing `'cubic'`. The amplitude (DN) of the waveform shot was passed as the value for 'x' and the Elevation (m) from the waveform shot was passed as the value for 'y'. We then interpolated it and took the measurements. This was repeated for each waveform using loops.

While in the process of creating our interpolation models we ran across a unique and unexpected runtime error. The exception is below.

```
ValueError: Expect x to not have duplicates
```

To overcome this error, we removed the waveforms that gave this error from the array that contained the waveform data frames and continued on with the testing of interpolations. We removed 11792 waveforms out of the total 35472 waveforms from the dataset. That is, approximately 33% of the waveforms were removed.

Linear interpolation took longer than cubic interpolation for 75.53% of the shots we tested, but was much more accurate. However, the cubic interpolation method used much less RAM than the linear method and ran quicker. These different interpolation methods have their strengths and weaknesses, but both are viable and useful in their different scenarios and situations.

4.1 Continuation

If we are able to continue this project we would like to see how the research we have already completed would be applicable to creating artificial intelligence programs ourselves and how the results and conclusions we have drawn would change the outcome of the algorithms we design.

It would be an interesting application of our research and would shed light on areas of computer science that we are interested in.

4.2 Future Applications

The future applications of our research and research similar to ours mainly pertains to the design and deployment of artificial intelligence and machine learning algorithms. Our research will help algorithm designers choose when to implement different interpolation methods, the viability of those methods with GEDI data and data similar to it, and how these methods use computer resources differently. It will help cut costs and save time when it is needed, and it will allow others to understand the differences between these two methods as they pertain to the data that we used.

In a broader sense the usage of computer resources is one of the most important topics in computer science. Aside from theories and ideas, computer resources are one of the most limiting factors in computer science today. The ability for computers to process and calculate large amounts of data continues to increase; however, it will always be an obstacle. Especially as computer science shifts its focus into more complex and larger webs of algorithms and continues to increase its ambitions and abilities in the realm of artificial intelligence and machine learning algorithms, every second, every minute of electricity, every byte of RAM counts. That is why our research will be beneficial to others in their processes of designing algorithms.

5. Conclusion

The deployment of these methods into your algorithm or artificial intelligence is completely dependent upon the results you are looking to achieve and the ability and mobility you have while running intensive code with large datasets. When running an algorithm on a less advanced machine it might be more beneficial to implement the linear interpolation method because of its less intensive nature. However, if you are looking for the best results no matter the technological cost, then you should deploy multiple methods and see which one achieves the best results for the data that you are working with. GEDI Level 1B data is a very unique and abnormal set of data, and what worked best for this data might not work best for yours.

6. Acknowledgements

We acknowledge and thank Brandon J. Woodard and Shinpei S. Nakamura for their mentorship and guidance in this process. We would also like to thank Sara I. Ansari for her contributions to the beginnings of our research process. Additionally, we would like to thank the staff of Lumiere research and our families for allowing us the opportunity to meet and work with one another.

7. References

Chen, S., Jakeman, A., Norton, J. (2008). Artificial Intelligence techniques: An introduction to their use for modelling environmental systems. *Mathematics and Computers in Simulation*, Vol. 78, Issues 2–3, Pages 379-400, ISSN 0378-4754.
https://doi.org/10.1016/j.matcom.2008.01.028

Hancock, S. (2018). GEDI simulator.
https://bitbucket.org/StevenHancock/gedisimulator

Hancock, S., Armston, J., Hofton, M., Sun, X., Tang, H., Duncanson, L., et al. (2019). The GEDI simulator: A large-footprint waveform lidar simulator for calibration and validation of spaceborne missions. *Earth and Space Science*, 6, 294– 310.
https://doi.org/10.1029/2018EA000506

Simard, M., Pinto, N., Fisher, J. B., and Baccini, A. (2011), Mapping forest canopy height globally with spaceborne lidar, *J. Geophys. Res.*, 116, G04021.
https://doi.org/10.1029/2011JG001708

8. Appendix

Please find all data, code, and work from our research in this github:
https://github.com/AR4152/GEDI-Project.git