The reality behind a LiDAR simulator

Rakesh Kumar Mishra, Bharat Lohani
Geoinformatics division, IIT, Kanpur

Here's a user-friendly GUI-based LiDAR simulator using object oriented software engineering approach. The simulator is conceived with three components - terrain component, sensor component and trajectory component - designing each component independently. Numerical methods are used to solve complex problems for generating LiDAR data from simulated terrain and flights. Considering the fact that LiDAR data files are very large, special data structures and file formats are designed to improve the performance and to solve memory problems. Developed in JAVA, this simulator is an ideal research tool and is scalable, easily maintainable and reliable.

1. Introduction

1.1 Background

The last decade has seen manifold growth in the use of aerial survey LiDAR (Light Detection and Ranging) technology. Due to the main advantage of measuring topography through highly dense and accurate data points which are captured at high speed, the LiDAR technology has found several interesting applications (Lohani, 2001; Queija, et al., 2005).

1.2 Object-oriented software engineering

Software engineering has traditionally been an expensive and time-intensive process. Object-oriented analysis and design is the principal industry-proven methodology that answers the call for a more cost-effective, faster way to develop software and systems. Object-oriented technology cuts development time and overheads, giving time to market and thus significant competitive advantage, enabling software engineers to produce more flexible and easily maintainable applications.

1.3 Characteristics of LiDAR simulator

A LiDAR simulator is aimed at faithfully emulating the LiDAR data capture process with the use of mathematical models under a computational environment. Basically, data generated by simulator should exhibit all characteristics of data acquired by an actual LiDAR sensor. Literature reveals that only a few attempts have been made by researchers to develop a simulator for LiDAR. These efforts are limited in their scope as either these consider the effect of only single parameter on one kind of object (Holmgren et al., 2003) or inaccurate scanning pattern (Beinat and Crosilla, 2002). Another attempt is made (Kukko and Hyyppa, 2007) using MATLAB however the simulator has limitations as it is designed only for test purpose and does not offer flexibility and completeness. More focussed and comprehensive efforts have been made to simulate the return waveform from a footprint (Sun and Ranson, 2000; Tulldahl and Steinvall, 1999).

1.4 Requirement of LiDAR simulator software

Considering the significance of LiDAR, there is a need to introduce LiDAR technology to students at undergraduate and postgraduate level. LiDAR instrument and data are costly. Collecting LiDAR data with varied specifications, as are desired for classroom teaching and laboratory exercises, may not be viable considering the cost and availability of the instrument. A simulator can generate various kinds of data, as and when desired, at minimal or no cost. This data could be very useful for conducting laboratory exercises. User control over the entire data generation process in simulator can also help students in understanding the functioning and limitation of LiDAR instrument. Further, error sources and their effect on LiDAR data can be understood. In any laboratory exercise, availability of ground truth is fundamental. While it may be difficult and expensive to collect ground truth in case of actual LiDAR data, for simulated data, the ground

truth is readily available. In addition to education, there are several other applications where data generated by simulator can be employed. In particular, for testing the information extraction algorithms for their performance over a wide variety of data is conveniently possible with simulated data. It can also be used for LiDAR flight planning.

2. Benchmark for LiDAR simulator

Considering the aforesaid applications, the following benchmarks are set for the simulator:

- Simulator should employ a user-friendly GUI (Graphical User Interface.)
- The simulator should be easily extendable and maintainable.
- The simulator should be designed for wider distribution over various computational platforms.
- The simulator should come along with a help/tutorial system which can explain concepts of LiDAR using user-friendly multimedia techniques.
- It should simulate a generic LiDAR sensor and some widely available sensors in the market.
- The simulator should facilitate selection of trajectory and sensor parameters as in actual case along with the facility of introducing errors in various component systems of the LiDAR.
- The simulator should facilitate data generation for actual earth-like surfaces.
- The output data should be available in commonly used LiDAR formats.

3. Software development

3.1 Software components

In view of the aforesaid requirements, object-oriented programming (OOP) has been used for developing the simulator. This technology offers a new and powerful model for writing software. OOP allows decomposition of a problem into a number of entities called objects and then builds data and functions around these objects. A major advantage of OOP is the code reusability. The simulator is composed of three basic components in addition to input and output modules as shown in *Figure 1*.

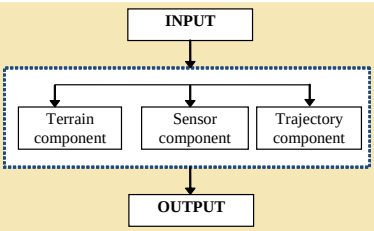

Figure 1. Basic components and their integration in software

Each component is divided into their sub modules and designed accordingly. Each module takes input from the user, while their integration generates LiDAR data as desired by the user. Each of these components is realised by developing algorithms and mathematical models. The following paragraphs describe the development of the present version of software.

3.2 Terrain component

Terrain component *(Figure 2)* is a combination of three modules, mathematical surface, raster and fractal. These three modules are designed to simulate bare earth surface and above ground objects. Each module has the facility to mark area of interest by using GUI. The marked area becomes available for LiDAR data generation. Further, the user has option to change parameters of each module to create the terrain of his choice. The GUI is shown in *Figure 3*.

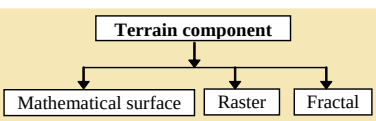

Figure 2. Terrain component

3.2.1 Mathematical surface

In this, terrain is modeled by using mathematical surface equations, which yield earth like surfaces. The software permits selection of these surfaces and their parameters. A few of these are

$$AX+BY+CZ+D=0$$
$$Z=A\,[\sin(X/B) - \sin(XY/C)] \quad (1)$$
$$Z=A\,[\sin(X/B) - \cos(XY/C)]$$

Figure 3. GUI for terrain component.

3.2.2 Raster

In this, a terrain using the above mathematical surfaces and different objects over the surfaces are rasterised. The developed GUI permits to choose the surface type and different types of objects on it. Further, it permits to drag and resize the placed objects. The generated raster data is saved in specially designed file format to improve the performance of the software. In addition, it is possible to import an existing raster file for which LiDAR data can be generated.

3.2.3 Fractal

Here, a natural-looking terrain is generated by using fractal algorithms. This module generates random fractal for the chosen area by using GUI. Different types of fractals can be generated by varying input parameters.

3.3 Sensor component

Two commercially available sensors (ALTM and ALS 50) and a generic sensor *(Figure 4)* are designed as independent modules. Developed GUI *(Figure 5)* facilitates selection of any of the above sensors and set their

parameters which are desired for LiDAR data generation, viz., scan angle, scan frequency, firing frequency, type of scanning *(Figure 6)* etc. The range of parameters is constrained in commercial sensors as per their specifications. The generic sensor permits selection of any range of parameters.

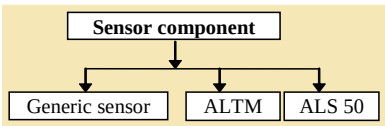

Figure 4. Sensor component


Figure 5. GUI for Sensor component

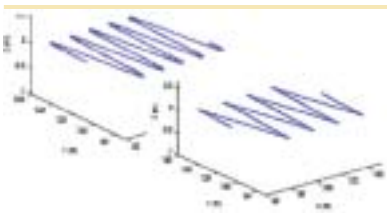
Figure 6. Sinusoidal and zigzag scan pattern

3.4 Trajectory component

This component is divided into two modules, location and attitude. Location module gives the position of the aeroplane at each instance on the basis of its sub-modules, velocity and acceleration. A nd the altitude module is responsible to generate semi random (controlled) Roll, Pitch, and Yaw. The integration of altitude and location modules, trajectory component generates trajectory coordinate (GPS coordinates) of aeroplane at each instance.

3.4.1 Location

A trajectory (referred as flight line henceforth to indicate a single flight) is

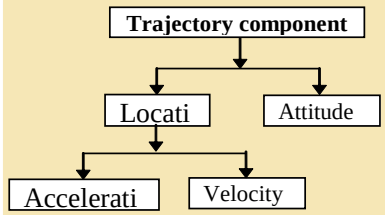

Figure 7. Trajectory component

defined by the location of laser mirror centre (point of origin of laser vector) in the absolute coordinate system at each instance of firing of laser pulse. The separate modules, velocity and acceleration are developed. The acceleration module generates pseudo-random acceleration in x, y and z direction within the permissible limit. It can be controlled by changing the parameters given in the software.

Velocity module calculates velocity of the aeroplane in x, y and z direction at each instance based on the input of the acceleration module. In software, it is also possible to do flight with constant velocity i.e. without acceleration variation. On the basis of velocity, location of the aeroplane can be determined.

3.4.2 Attitude

Due to internal and external forces, the attitude changes within certain limits and may exhibit random behaviour. To realise this, attitude model is designed. This module generates pseudo random attitude. The attitude can be changed within the limit by changing the parameters given in the software. Attitude can also be set to zero. Further, attitude values of actual flight can also be imported form file.

3.5 Integration of components

The components discussed above are integrated by generation of the laser vector and its intersection with simulated terrain. The point of intersection yields the coordinate of terrain point.

Depending on the area chosen for the LiDAR data generation, the direction of flight lines (Figure 8) is either


Figure 7. GUI for trajectory component

chosen by the user or the optimum direction is determined by the software. In later case, the flight direction is determined by using the principal direction of the chosen area. For algorithmic detail, please refer to Lohani and Mishra, 2007.

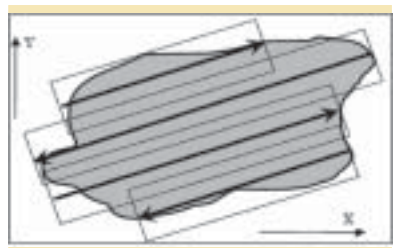
Figure 8. Area of interest (gray), flight lines in optimal direction (arrows) and swaths with overlap (thin rectangles)

The number of flight lines depends on the direction of flight and sensor parameters (i.e. swath width and percentage overlap). The initial and final points of each flight are calculated by the software. The total numbers of flight lines cover the whole area chosen for LiDAR data generation.

Trajectory component generates trajectory coordinates for each flight line depending on the distance of flight line, firing frequency and velocity of the aeroplane. The trajectory is rotated as per the direction of the flight lines. The laser vector is generated by using trajectory coordinate and corresponding instantaneous scan angle and attitude.

The point where laser hits the terrain, following the above laser vector, is computed by solving the intersection of laser vector and the mathematical surface or rasterised terrain. Solution

is realised by using specially formulated numerical methods. These methods differ for mathematical surface and raster terrain and also depend upon the basic equation employed to create terrain. These methods are incorporated in separate modules.

The size of raster data becomes very large, in particular when raster cell is small. Therefore to solve intersection, it is not feasible to store the entire data in memory. Special data structure and algorithms are designed to logically tile the data. The tiles which fall under the swath of the flight line can be known by the algorithm. It reads only those tiles which fall under the swath of a particular flight line.

3.6 Error introduction in data

LiDAR data suffer from systematic and random errors of different kinds (Huising and Pereira, 1998). Errors in position and orientation of platform and in angle and range measurement by sensor propagate in final coordinates. It is proposed to provide facility for introduction of these errors in the future version of simulator. In the present version, a normal error is introduced in the terrain coordinates computed in the above step in X, Y and Z directions separately. The system for introducing error in X direction is shown below:

$$X_T^i = X_t^i + N(\mu_X, \sigma_X{}^2) \qquad (2)$$

Where XTi is the X coordinate value with error. Similar systems with different values of parameters are used for Y and Z coordinates. It is assumed that errors in X, Y and Z directions follow normal distribution. Further, when introducing these errors, it is ensured in algorithm that there is no spatial auto-correlation of error. The parameters of this distribution are known from field experience and are reported by the vendors of sensors. The simulator facilitates variation of these parameters.

3.7 Output

This module has several sub-modules which generate different output files as in case of an actual LiDAR survey. The description about the modules is given as follows.

- X, Y, Z: This module writes generated X, Y, Z data in the ASCII format. This can be displayed in many 3D display software e.g Surfer, Voxel, Terascan etc.
- LAS 1.0 format: LAS 1.0 is ASPRS LIDAR data exchange format standard. The intention of the data format is to provide an open format which allows different LIDAR vendors to output data into a format which a variety of LIDAR software vendors can use. The module writes data in this format so that the generated data is in a standard format.
- *LAS 1.1 format:* This format is the next version of LAS 1.0 format. This module writes generated data in this format.
- *Time, Attitude:* This writes time, Roll, Pitch and yaw data column wise in ASCII format.
- *Time, Acceleration:* This writes time and acceleration values in X, Y, Z direction column wise in ASCII format.
- *Time, Error:* This writes time and error introduced in X, Y, Z column wise in ASCII format.
- *Report:* This writes a report file in ASCII format which contains detailed information regarding LiDAR data generation process. e.g. velocity, altitude, number of flight lines, sensor type, total flight time, firing frequency, scan frequency, swath and terrain type etc.

4. Result and discussion

Simulated attitude and acceleration are shown for a duration of 14 seconds (Figure 10). Though, it is statistically difficult to compare simulated data with any set of actual flight data, as these two represent two different populations, the former amply exhibit the random nature of parameters as in any normal flight.

Figure 9. LiDAR data output menu and options

A hypothetical terrain (400 m X 600 m) is created over a flat surface and is populated with building like shapes. By using drag-drop facility of software, different types of buildings are created. The flat surface and objects on top of it are rasterised. A view of this is shown in Figure 11, which is displayed using surface feature of the surfer.

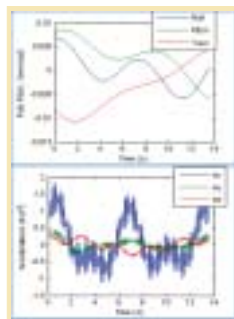

Figure 10. Attitude (top) and acceleration (bottom) values

Lidar data are generated for this terrain with the parameters: Flight velocity: 60 m/s; Altitude: 190m; Firing frequency: 20 KHz; Scan frequency: 48Hz; Scan angle 500; No. of flight lines: 3; Overlap 10%. The resulting LiDAR data are imported in Terrascan software and displayed. Only few views are being presented for the sake of space as shown in Figure 12 and Figure 13. To show the effect of multiple flight lines, small altitude value is taken.

Profile view of the generated LiDAR data is shown in Figure 13 which has three flight lines. This shows various buildings where data are captured, while the interplay of object and shadow is also evident. Not all black areas (i.e. where data are not captured) are shadows. For example, the roof of

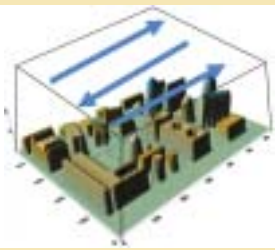

Figure 11. Surfer surface of the created terrain.

building marked by white oval is not fully captured. The reason for this can be understood in profile (also marked by white oval.). The black area is not being covered by either flight line. This also serves as an example of poor choice of scan angle and flying height. The zoomed out view of a complex building (shown in Figure 14 (a) and (b)) of the same place with no attitude and with high attitude variations. These figures give understanding that how attitude effects on the point cloud. This also shows that the spread of point cloud depends on the location of the object with respect to flight line and parameters chosen for sensor. LiDAR

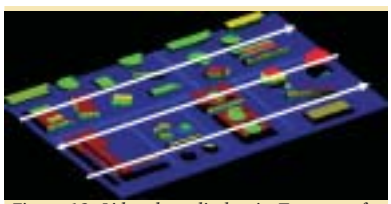

Figure 12. Lidar data display in Terrascan for the above surface.

Figure 13. LiDAR data display in Terrascan-profile view. The profile is shown along with the flight lines and swaths. The building within the oval is not measured.

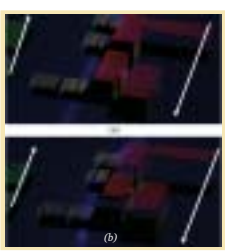

Figure 14. Effect of high attitude on complex building.

(a) with no attitude variation.

(b) with high attitude variation.

data points are available on the vertical wall facing the flight line, while no data points are captured on the other wall. The flight lines are shown by an arrow. The figure also shows the overlap of two flight lines.

The generated raster for a fractal surface is shown in Figure 15 in the surfer. The LiDAR data is generated for this fractal (shown in Figure 16)

5. Conclusion

Figure 15. Fractal surface display in surfer.

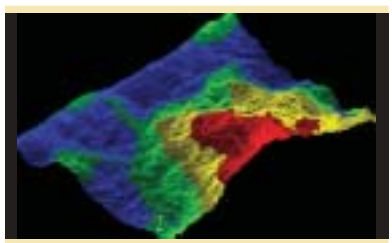

Figure 16. Lidar data of fractal surface display in Terrascan

The present version of software offers a user-friendly GUI based interface so that the user can use this software with ease. It simulates the process of LiDAR data collection for LIDAR sensor as well as the user has freedom to set the parameters according to his choice which is not possible in the case of actual LIDAR. Different data sets can be generated for the same terrain by setting different parameters. The simulator can be useful to generate LiDAR data for research to test algorithms. It is also useful in a classroom for demonstrating LiDAR data capture process and understanding the effect of flight and sensor parameters and their errors. Further, it can be used in LiDAR flight planning. Before actual flight, the effect of sensor parameters, flight direction etc can be seen. The software is developed using object-oriented methodology so it can be extended further very easily. JAVA programming language is used to develop the software as it supports object oriented programming as well as having excellent GUI features. The platform independency of JAVA made this software to run on multiple operating systems.

The present version of simulator is being modified to incorporate more faithfulness by introducing separate GPS and IMU data collection and their integration; by introducing concepts of separation of GPS antenna, laser head and INS; by incorporating atmospheric effects and by facilitating for multiple returns, etc.

References

Beinat, A., and Crosilla, F., 2002. A generalized factored stochastic model for optimal registration of LIDAR range images, International Archives of photogrammetry an remote sensing and spatial information sciences, 34(3/B), pp. 36-39.

Marciniak, J.J., 2001, Process models in software engineering, encyclopedia of software engineering. Walt Scacchi, Institute for Software Research, University of California, Irvine.

Holmgren, J., Nilsson, M., and Olsson, H., 2003. Simulating the effect of lidar scanning angle for estimation of mean tree height and canopy closure. Canadian Journal of Remote Sensing, 29(5), pp. 623-632.

Husing, E. J. and Pereira, L. M., 1998, Errors and accuracy estimates of laser data acquired by various laser scanning systems for topographic applications, ISPRS Journal of Photogrammetry & Remote Sensing, 53(5), pp. 245-261.

Lohani, B., Mishra, R. K., 2007, Generating LiDAR data in laboratory: LiDAR simulator, International Archive of Photogrammetry , and Remote Sensing XXXVI(3)W52 of Laser Scanning 2007 and SilviLaser 2007, Espoo, Finland, September 12-14.

Kukko, A. and Hyypa J., 2007, Laser scanner simulator for system analysis and algorithm development: A case with forest measurements, ISPRS Workshop on LASER Scanning 2007 and SiviLaser 2007, Espoo, September 12-14, 2007, Finland.

Lohani, B., Reddy, P., and Mishra, R.K., 2006, Airborne Altimetric LiDAR Simulator: An education tool, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science, XXXVI(6), Tokyo, Japan.

Lohani, B., 2001, Airborne altimetric LiDAR for topographic data collection: Issues and application., Proc. of International conference MAPINDIA-2001, 7-9 February 2001, New Delhi.

Queija, V. R., Stoker, J. M., and Kosovich, J. J., 2005, Recent U.S. geological survey applications of LiDAR, PE&RS, 71(1), pp. 5-9.

Sun, G. and Ranson, K. J., 2000. Modeling Lidar returns from forest canopies, IEEE trans. On geosciences and remote sensing, 38(6), pp. 2617-2626.

5.1 Acknowledgements

This work is supported under RESPOND programme of ISRO.