GENERATING LIDAR DATA IN LABORATORY: LIDAR SIMULATOR

Bharat Lohani* and R K Mishra

Department of Civil Engineering, Indian Institute of Technology Kanpur, Kanpur 208016, INDIA – blohani@iitk.ac.in

Commission VI, WG VI/4

KEY WORDS: Altimetric LiDAR, Simulator, Modelling of sensor, Software, Research and Education tool

ABSTRACT:

This paper describes the methodology adopted for developing a simulator for airborne altimetric LiDAR. The goal is to model LiDAR sensor functioning so LiDAR data can be generated for a user specified terrain with given parameters of the sensor and aerial platform. The simulator is conceived having three components: 1) Terrain component, 2) Sensor component and 3) Platform component. Terrain component is formed using multiple mathematical surfaces for bare terrain and for objects on top of the surface. Further, the terrain can be represented using a raster. The sensor component permits a user to opt for the commercially available sensors or a generic sensor and accordingly generates data. The third component attempts to model the platform parameters, viz., velocity, roll, pitch, yaw and accelerations. LiDAR data are generated by first finding the equation of laser vector that changes with each pulse and then determining the point of intersection of this vector with the mathematical surface or the raster representing terrain. This GUI based simulator, developed in JAVA, is an ideal tool for research and education.

1. INTRODUCTION

The last decade has seen manifold growth in the use of airborne altimetric 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).

1.1 What is a 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 simulator for LiDAR instrument. These efforts are limited in their scope as either these consider effect of only single parameter on one kind of object (Holmgren et al., 2003) or inaccurate scanning pattern (Beinat and Crosilla, 2002). More focused and comprehensive efforts have been made to simulate the return waveform from a footprint (Sun and Ranson, 2000; Tulldahl and Steinvall, 1999).

1.2 Why a simulator?

LiDAR data with varying specifications are fundamental for success of a research. To judge the optimality of algorithms or suitability of data for an application one needs to work with data with varying characteristics. Collecting these data in field is not feasible in view of extensive time and resource involvement. Further, for success of a research (e.g., building extraction from LiDAR data), availability of accurate and complete ground truth is crucial, which is difficult and expensive to collect in field. LiDAR simulator can generate data with all user specified specifications at no cost. Data can be generated even with those specifications that are not available in commercially available sensors. Also, in the case of simulator complete and 100 per cent accurate ground truth is available. Simulated data can help in evaluation of the effect of noise and/or systematic error in final outcome.

A simulator is also a useful tool for education, as data generation process and the effect of error and various flight parameters can be understood in laboratory. In view of the cost and sometime the security/proprietary concerns associated with LiDAR data, the same are not cheaply and readily available for classroom activities. Simulator can help by producing data for various laboratory exercises aimed at understanding LiDAR data, their errors and information extraction algorithms.

2. DESIGN BENCHMARKS FOR THE SIMULATOR

The following benchmarks are set for an ideal simulator:

1. Simulator should employ a user-friendly GUI (Graphical User Interface.)
2. Simulator should be designed for wider distribution over various computational platforms.
3. The simulator should come along with a help/tutorial system which can explain concepts of LiDAR using user-friendly multimedia techniques.
4. It should simulate a generic LiDAR sensor and some other sensors available in market.
5. 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 LiDAR.
6. Simulator should facilitate data generation for actual earth-like surfaces.

7. The output data should be available in commonly used LiDAR formats.

3. METHODOLOGY

3.1 Coordinate systems used

As shown in Figure 1 two coordinate systems are considered. The first coordinate system is (X, Y, Z), which is absolute. All trajectory and terrain coordinates are determined in this system. A system which translates with platform and remains parallel to absolute coordinate system is considered at the laser head and is henceforth referred to as gyro coordinate system. The second coordinate system is the body coordinate system(x,y,z), which has its origin at laser head and is affected by roll, pitch and yaw rotations. Scanning takes place in this coordinate system, i.e., in y-z plane. The laser vector at any instance is defined using direction cosines and coordinates of laser head in gyro coordinate system.

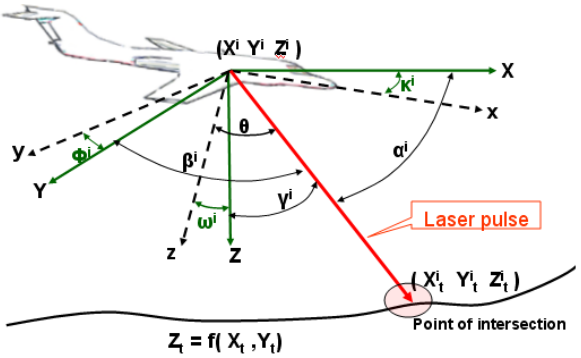


Figure 1. Schematic of laser vector intersection with a surface and coordinate systems

3.2 Simulator components and programming environment

Simulator components are shown in Figure 2. These components take form as per user input, while their integration generates LiDAR data. Following paragraphs describe development of these components as implemented in the latest version of simulator. The simulator has been improved substantially from its previous version (Lohani et al., 2006). This paper will focus more on description of these improvements. However, to make it complete a few parts are reproduced from Lohani et al. (2006) with modifications.

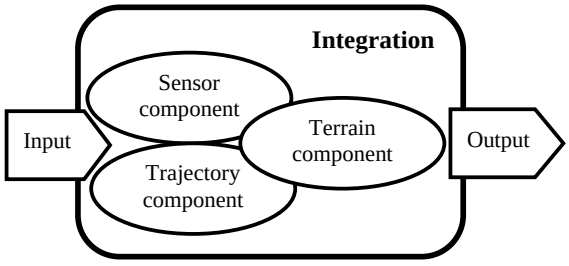


Figure 2. Basic components and their integration

Programming language JAVA has been chosen, as it offers good numerical and graphical programming besides, and most importantly, being platform independent. The parameters required to define three individual components and output data format are input through user-friendly GUIs.

3.3 Terrain component

Vector and raster approaches are chosen for simulating bare earth surface and above ground objects as described below. Through a GUI, as shown in Figure 3, a user is prompted to select an area of interest, by marking it using the mouse on screen. The area selected from the underlined mathematical surface or raster becomes available for LiDAR data generation.

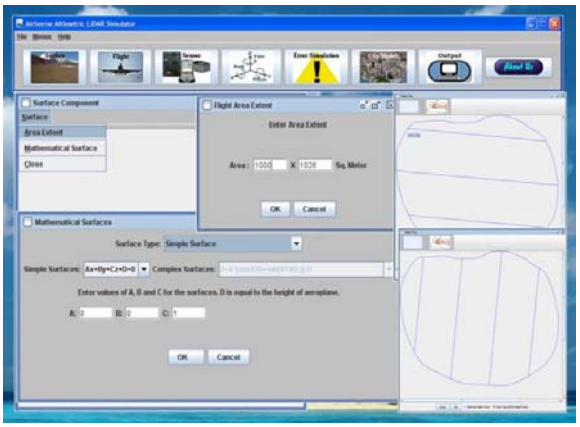


Figure 3. GUI showing selection of underlying mathematical surface, its extent and area of interest on it for LiDAR flight along with flight lines

3.3.1 Vector approach

In this, a terrain is represented using mathematical equations, which yield earth like surfaces. The GUI permits selection of these surfaces and their parameters. A few of these are:

$$Z = AX + BY + C$$
$$Z = A[\sin(X/B) - \sin(XY/C)] + D$$
$$Z = A[\sin(X/Y) - \sin(XY/B)] + C \quad \textbf{(1)}$$

3.3.2 Raster approach

In this the surfaces resulting from the above equations are rasterized. Most importantly, this approach permits populating the raster with above ground objects. Those cells, where an over ground object is desired to be placed, take new values as per the height and shape of object. In addition, it is possible to import an existing raster file (say DEM) for which LiDAR data can be simulated.

3.4 Sensor component

The GUI prompts user to select any one of the two commercially available sensors (ALTM3100 or ALS50) or a generic sensor. While the range of parameters is constrained in commercial sensors, as per their specifications, the generic sensor permits selection of any range of parameters. Having selected the sensor, the user is prompted to enter the sensor parameters which are desired for LiDAR data generation, viz., scan angle, scan frequency, firing frequency, type of scanning etc. (Figure 4.)

Depending the type of scanning (which may be zig-zag or sinusoidal) the instantaneous scanning angle is determined by

the following model. Let time taken to complete 1/4th of a scan is T and there are P numbers of points in this. The maximum scan angle is θ_{max}. For the i^{th} point, which is fired at time t_i from the beginning of scan, the scan angle will be:

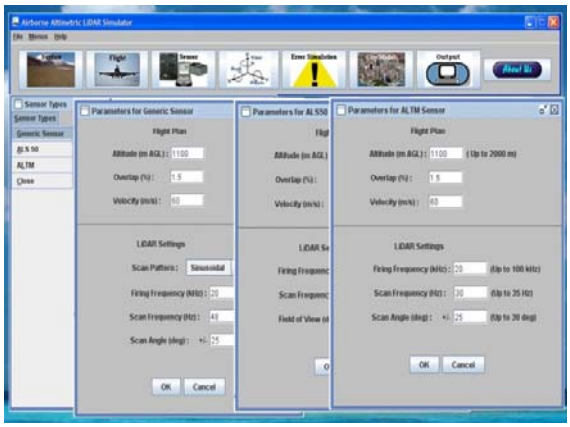


Figure 4. Parameter input for different sensors

$$\theta_i = \frac{\theta_{\max}}{P} i \quad \text{For zig-zag}$$
$$\theta_i = \theta_{\max} \sin\left(\frac{\pi}{2T} t_i\right) \quad \text{For sinusoidal} \qquad \textbf{(2)}$$
$$\text{where,} \quad t_i = \frac{T}{P} i$$

The resulting trajectories are shown in Figure 5. By changing the parameters listed above the spread of points within scan can be altered.

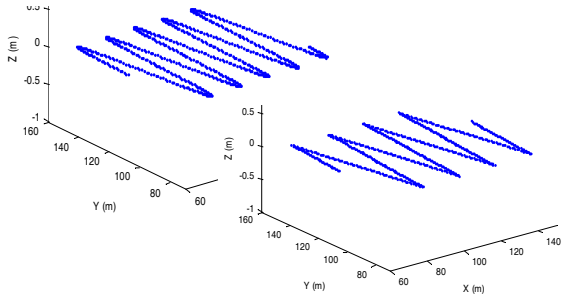


Figure 5. Sinusoidal and zig-zag scan generation

3.5 Trajectory component

Complete trajectory of aircraft is made of several flight lines as shown in Figure 6. Depending the area chosen for LiDAR data generation (gray area in Figure 6) and sensor parameters (i.e., swath width and per cent overlap) the direction of flight lines is either chosen by the user or an optimum direction is determined by the software. In latter case, the flight direction is determined by making use of the principal direction of area. For this, first using Douglas-Peucker algorithm (Douglas and Peucker, 1973) the number of vertices defining the area of interest marked by the user are reduced, which ensures that the area marked has no small kinks which are the artefacts of drawing on screen by hand. Covariance matrix is generated for the coordinates of all points forming the area of interest polygon. The first eigen vector of this is used to determine the principal direction of the polygon. The flight direction is oriented in the principal direction, which makes the total flight line length required to cover the area a minimum (in most of the cases.)

The software also determines the location of flight lines (thick lines in Figure 6) so as to cover entire area considering the overlap specified. The algorithm places the first flight line (top flight line in Figure 6) in such a way that the swath covers up to the edge of area. The last flight line is placed considering the spacing between flight lines for given overlap. It is shown in Figure 6 that in order to cover full area of interest some extra area (*union of all swath rectangles – area of interest*) is also scanned. Using the points of intersection of periphery of area and the flight line the starting and ending points of flights are determined. The following section describes computations for an individual flight line. The same procedure is followed for other flight lines also.

3.5.1 Location

A trajectory (referred as flight line henceforth to indicate a single flight) is 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. To simulate the flight line and to incorporate a possibility of introducing errors in parameters the following procedure is employed.

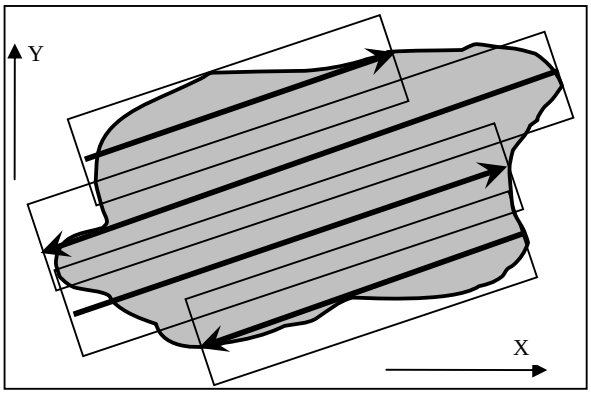


Figure 6. Area of interest (gray) , flight lines in optimum direction (thick arrows) and swaths with overlap (thin rectangles)

Let, time interval between firing of successive pulses is d_t , which is equal to *1/F,* where F is firing frequency. Total number of points on flight line wherefrom pulses are fired is *n,* which will depend upon the length of flight line. Velocity of platform in flight direction at i^{th} point on flight line is u^i. Let the laser head coordinates at i^{th} point on trajectory are (X^i, Y^i, Z^i).

At each successive d_t interval one needs to compute the location of laser head. The aerial platform is subject to internal and external force fields with the net effect that the platform is subject to random accelerations in three axes directions. The following system is employed to simulate accelerations. This system ensures a pseudo-random generation of acceleration values.

$$a_x^i = \sum_{j=1}^{J} A_j \sin(B_j(\frac{2\pi}{T}(id_t))) + \sum_{k=1}^{K} C_k \cos(D_k(\frac{2\pi}{T}(id_t))) \qquad \textbf{(3)}$$

Where a_x^i is the acceleration at i^{th} point in X direction. T is the total duration of a flight line. The parameters of this equation J, K, A, B, C and D control the direction and quantum of acceleration. Developed software permits selection of these parameter values within ranges that generate accelerations as may be observed in a normal flight. Similarly, a_y^i and a_z^i are also generated with different values of parameters in above equation. Using the acceleration values at i^{th} point the new location of laser head (i.e., X^{i+1}, Y^{i+1}, Z^{i+1}) after d_t interval is computed using equations similar to:

$$X^{i+1} = X^i + u_x^i d_t + \frac{1}{2} a_x^i d_t^2 \quad \textbf{(4)}$$

Where, u_x^i is the velocity in X direction.

3.5.2 Attitude

As in case of acceleration, due to internal and external force fields, the attitude will change within certain limits and may exhibit a random behaviour. To realise this, the attitude values (i.e., ω^i, φ^i, κ^i) at any i^{th} point are determined using the equation (3). Similar to the case of acceleration, the developed simulator permits selection of these parameters in the ranges which generate attitude values as in case of a normal flight.

The outcome of aforesaid is that at each point wherefrom a laser pulse is fired the attitude values and coordinates of point are known in the absolute coordinate system.

3.6 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. As shown in Figure 1, for any i^{th} point on trajectory there exists a laser vector. Equation of laser vector is given as:

$$\frac{X - X^i}{a^i} = \frac{Y - Y^i}{b^i} = \frac{Z - Z^i}{c^i} \quad \textbf{(5)}$$

Where a^i, b^i, and c^i are direction cosines ($cos\alpha^i$, $cos\beta^i$, and $cos\gamma^i$, respectively) of laser vector with respect to gyro coordinate system at i^{th} point. The values of α^i, β^i, and γ^i are determined from known values of attitude (ω^i, φ^i, κ^i) and instantaneous scan angle (θ).

The point where laser hits the terrain, following the above laser vector, is computed by solving for intersection of equation (5) and equation (1) or the rasterized terrain. Solution is realised using specially formulated numerical methods. These methods differ for vector and raster terrain and also depend upon the basic equations employed to create terrain. The raster data size becomes very large (raster cell is taken 10 cm). Therefore, to solve the intersection it is not feasible to store entire data in memory. Special data structuring is adopted by tiling the raster and reading the data only from those tiles which fall under the swath of flight line. This is ensured by using the "point in polygon" algorithm which determines whether a tile intersection is within the area covered by flight line (i.e. *Swath x Flight length*). Full description of these methods is beyond the scope of this paper. At this stage coordinates of all points of intersection (X_t^i, Y_t^i, Z_t^i) are known.

3.7 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 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:

$$X_T^i = X_t^i + N(\mu_X, \sigma_X^2) \quad \textbf{(6)}$$

Where X_T^i 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.

3.8 Output generation

The software facilitates output of LiDAR coordinates in simple ASCII format or standard LAS format. Further, a variety of other reports are generated, e.g., sensor parameters, flight parameters and parameters used to generate terrain (Figure 7). These reports are output in textual format which can be employed by user for further study.

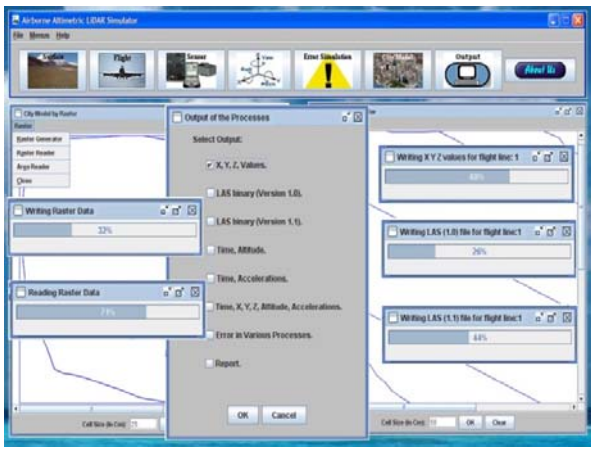


Figure 7. LiDAR data output menu and options

4. RESULT AND DISCUSSION

Simulated trajectory and attitude parameters are shown for a duration of 5 seconds (Figure 8). 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.

A hypothetical terrain (2km by 2km) is created over a flat surface and is populated with building like shapes (length and breadth ranging from 150 m to 300 m) along with 6 cylindrical objects (50 m diameter and 90m height). Large objects are chosen here to fill the 2km by 2km area. The flat surface and

objects on top of it are rasterized. The raster takes values as per the underlying surface or the object. A view of this is shown in Figure 9, which is generated using surface feature of Surfer. This view also shows the location and direction of the four flight lines. This view will help understanding the results presented later. LiDAR data were generated for this terrain with the parameters: Flight velocity: 60 m/s; Altitude: 600m; Firing frequency: 20000 Hz; Scan frequency: 48 Hz.; Scan angle: 50°; No of flight lines: 4; Overlap 1.5 %

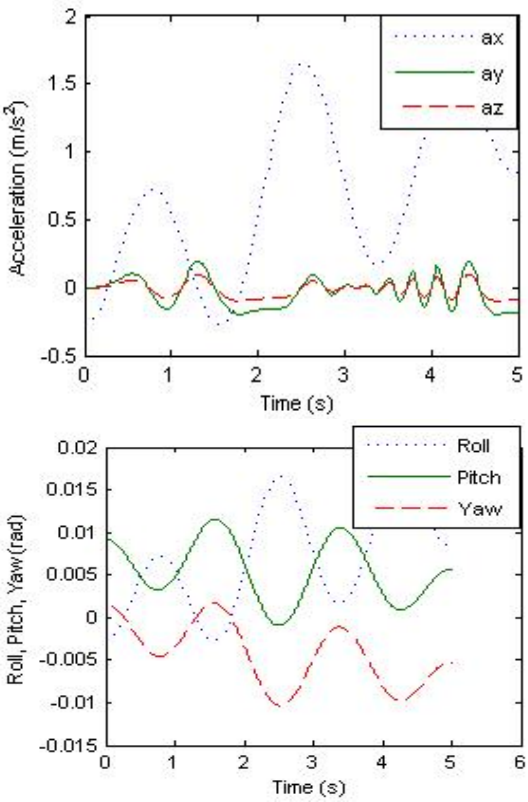


Figure 8. Acceleration (top) and attitude (bottom) values

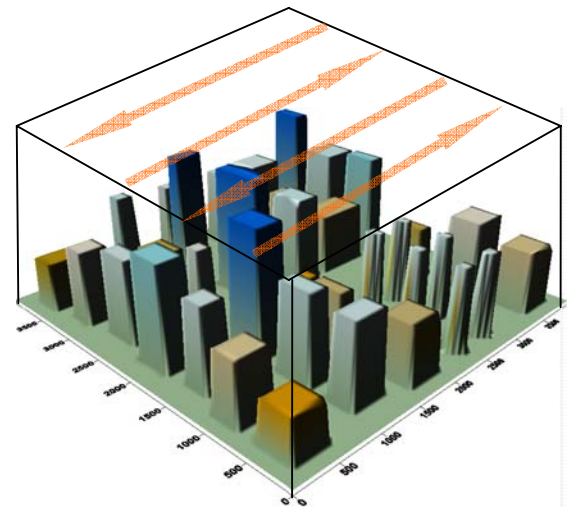
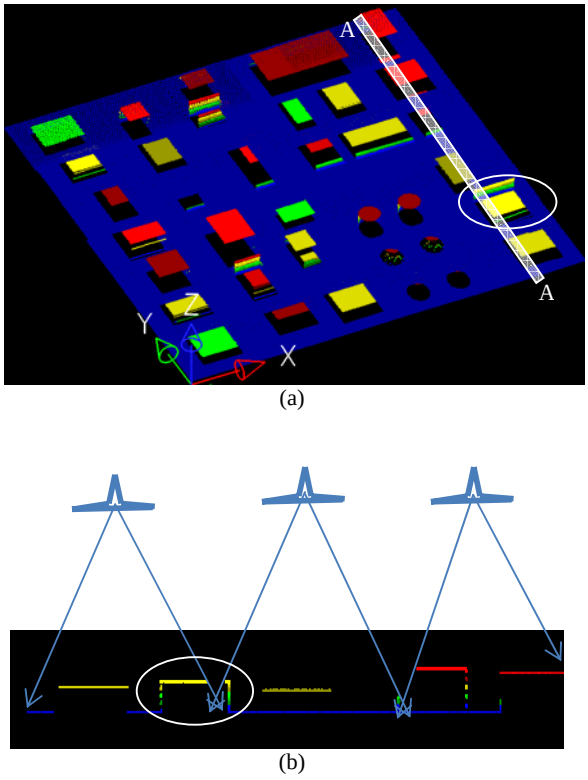


Figure 9. Surfer surface view of the chosen terrain (heights are exaggerated)

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 10 and Figure 11. Data generation for objects of different shapes and sizes and as well as for objects situated at different locations w.r.t. the flight line can be understood from these figures. In Figure 10(a) a perspective view is shown, which 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. This can be understood from the profile drawn about A-A and shown in Figure 10 (b). For example, the roof of 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 lines. This also serves as an example of poor choice of scan angle and flying height, which can be understood by simulator.

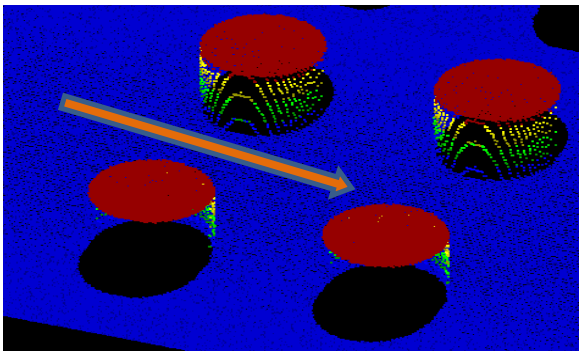


(a)

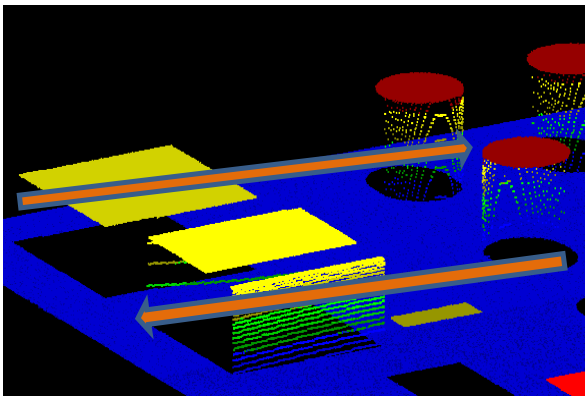
(b)

Figure 10. (a) LiDAR data display in Terrascan-perspective view; (b) Profile along A-A band shown by white colour in top image. The profile is shown along with the flight lines and swaths. The building within oval is not fully measured.

The zoomed out views (shown in Figure 11 (a) and (b)) of the same data show the point cloud as obtained for different objects. Location of flight lines is also shown. The spread of point cloud depends on the location of the object with respect to flight line and the parameters chosen for sensor. LiDAR data points are available on the vertical walls facing flight line, while no data points are captured on the other wall. Data in these examples are produced with error. This is evident as the points do not fall in smooth scan lines.



(a)



(b)

Figure 11. Zoomed out display of point cloud

5. CONCLUSION

The presented simulator emulates existing commercial sensors and models a generic LiDAR sensor to generate data over a user specified terrain. A user can alter the sensor and trajectory parameters with ease and generate the resulting LiDAR data. Error can also be incorporated in output. The simulator has a user friendly GUI designed in architecture independent JAVA language.

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.

Terrain representation using raster has solved to a large extent the problem of representation of bare earth and objects. However, this results in large data size which is managed through data structuring so the data are brought into the simulator in chunks as needed.

A MATLAB based system has been developed in parallel for simulating full waveform digitization for a Gaussian pulse. Efforts will be made in future to integrate this with the present simulator, thus to generate multiple return data and waveform digitization.

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.

Douglas, D., and Peucker, T., 1973, Algorithms for the reduction of the number of points required for represent a digitzed line or its caricature, *Canadian Cartographer*, 10(2), pp. 112–122.

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., Reddy, P., and Mishra, R., 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.

Tulldahl, H. M. and Steinvall, K. O., 1999, Analytical waveform generation from small objects in lidar bathymetry, *Applied optics*, 38(6), pp. 1021-1039.

5.1 Acknowledgements

This work is supported under RESPOND programme of ISRO. Authors are grateful to two anonymous reviewers and editor.