OPTIMIZING COMPUTATIONAL PERFORMANCE FOR REAL-TIME MAPPING WITH AIRBORNE LASER SCANNING

P. Schaer, J. Skaloud

TOPO Lab, Swiss Federal Institute of Technology Lausanne (EPFL), Station 18, CH-1015 Lausanne, Switzerland
(philipp.schaer, jan.skaloud)@epfl.ch

Commission I, WG I/5

KEY WORDS: **Mapping, Registration, Laser scanning, Algorithms, Georeferencing, Software, Real-time**

ABSTRACT:

This paper focuses on evaluating and optimizing the computation procedures for achieving real-time mapping with modern Airborne Laser Scanning (ALS) instrumentation. The computation steps under consideration include real-time point cloud generation at full resolution, strip-wise analyses of scanning density and coverage, error propagation, ground classification and digital terrain and surface model production and DTM quality map. Different approaches for each mentioned step are implemented within a custom system and evaluated under real mapping scenarios for speed and correctness. The investigation reveals that real-time generation of laser point cloud (georeferencing) is feasible with conventional laptop for scanning rates up to 200 kHz. Further it is shown that the analyses of point cloud density and coverage are less computationally demanding, so is the calculation of digital surface model (DSM) and generation of corresponding hillshade image. Therefore, these tasks can be completed shortly after each flight line. On the other hand, the most computationally demanding tasks are identified as point-cloud classification required for DTM production as well as the error propagation. The latter tasks are therefore examined for optimizing their execution speed while maintaining correctness. The retained solutions are presented in the paper together with recommendation for further development.

1. INTRODUCTION

Airborne Laser Scanning (ALS) is a very effective and accurate method for establishing digital elevation models (DEM) from airborne platforms. In some applications the requirements on point density and DTM accuracy can be as high as several points per m^2 and 0.1 m, respectively. Contrary to the terrestrial laser scanning (TLS), the conventional airborne laser scanning generates the point-cloud coordinates only after the mission. There, the laser data is merged with the trajectory in a process that is sometimes referred to as 'basic-processing'. The laser returns are then separated to those belonging to vegetation, buildings and terrain by a (semi-) automated classification procedure. The digital surface models (DSM) and digital terrain models (DTM) are established from the aggregated point cloud as explained in (El-Sheimy et al., 2005). The latter steps are referred to as 'advanced processing'. The major drawback of the basic and advance processing procedures is the latency between data collection and quality control for completeness and correctness of the resulting DEM.

In our previous investigations we have introduced methodologies for in-flight quality control (Schaer et al., 2008) and, more, recently Real-Time-Kinematic (RTK-)ALS (Skaloud et al., 2010). The latter is an extension of the former and strives to perform all basic and advanced processing steps in flight (Fig. 1) together with DEM quality control.

To cope with the time-constraints of a real-time computational environment, the execution efficiency of the algorithms is crucial. This contribution investigates the performance (in terms of computation-time) of the critical components of the formerly presented in-flight quality assessment tool. The structure of the paper is following.

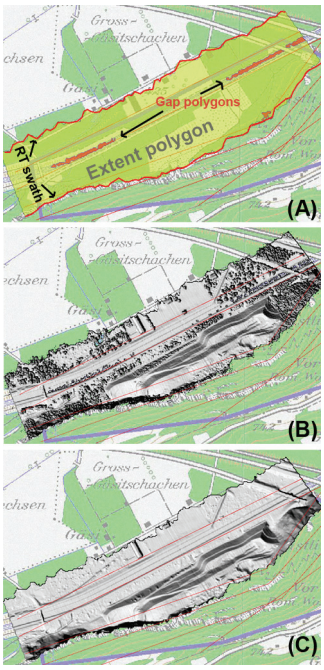

Figure 1: Display of different outcomes of the in-flight point-cloud processing: (A) RT swath borders, data extent and gap polygons, (B) DSM hillshade, (C) DTM hillshade.

After listing the processing steps in Sec. 2, we benchmark all real-time (Sec. 3) and time-delayed (Sec. 4) computational procedures on the same processor. Although the processors speed is evolving quickly the relative comparison is more important. Nevertheless, the absolute values will show that despite the large data quantities collected by modern LiDAR, the complete processing cycle is feasible with conventional hardware when some tasks are distributed among few

processors. Suggestions for future development are given in conclusion.

2. PROCESSING STEPS

The processing steps can be divided into two categories as a function of the computation latency (Figure 2):

- On-line (real-time) processing: This includes the generation the point-cloud by merging the trajectory with the laser measurements and the integrated trajectory (Skaloud et al., 2010). The point-cloud coordinates and all information needed for subsequent processing (i.e. accuracy information for the error propagation) is saved to a file. Additionally, RT swath boundaries can directly be displayed (Figure 1.A) to inform the operator about the progress of the scan (Schaer et al., 2008).
- Time delayed or strip-wise processing: Once the point-cloud data for an entire flightline is available, the data are loaded and the processing and quality control (QC) is performed. Here we make the distinction between basic QC and advanced QC operations. The first includes point-cloud density computation, data extent and gap detection and DSM computation. The latter englobes point-cloud classification (search of bare-earth points), DTM generation, point-wise error propagation and the generation of point-cloud quality maps (Schaer et al, 2009). The outcome of each processing step can be displayed in a GIS-like environment (Figure 1). A detailed description of the above-mentioned processing steps can be found in (Schaer, 2009).

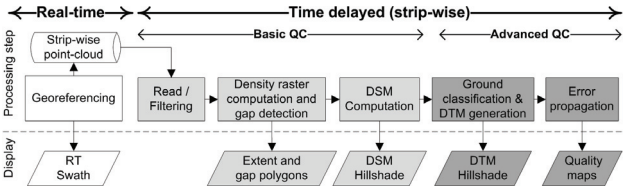

Figure 2: Generalized workflow for in-flight point-cloud processing.

The following sections investigate the performance (in terms of computation-time) of the critical components of the previously mentioned steps. All performance tests were carried out using a DELL Latitude 820 laptop (Intel(R) DualCore CPU T7600 @ 2.33 GHz, 2GB RAM) and were performed on data from the Scan2map system (Schaer et al., 2008) .

3. ON-LINE COMPUTATION

As explained in the previous section our computational strategy is different within and outside a flight line. When the surface is scanned over the area of interest only the execution of vital tasks is performed. These are: data storage, real-time GPS/INS integration with feedback to pilot guidance, real-time point-cloud georeferencing (all points) and display of the real swath.

3.1 Point cloud generation

Within the RT processing chain, the DG of laser data is the computationally most demanding task as it has to handle trajectory data at 400 Hz and in case of the Scan2map the laser data at 10 kHz. Fig. 3. Illustrates the performance of the georeferencing engine LIEOS for one million laser measurements in a local mapping frame. The raw binary data reading and the georeferencing task itself require about 5s (corresponds to a processing rate of approx. 200 kHz). For the tested dataset, the computation-time was nearly invariant to the different georeferencing methods (coarse, approximate, rigorous) implemented in LIEOS (Schaer et al., 2008). Simultaneous georeferencing and data logging increase the processing time. If the data is logged to an ASCII file, the computation-time exceeds 20s (~50 kHz). However, it the data is stored in some optimized binary format, as typically the case; the slowing-down of the process is moderate, resulting in a processing rate of approx. 150 kHz. This example shows that in the current configuration LIEOS is capable of performing RT DG for scanning rates between 150 to 200 kHz which is also the limit of current hardware.

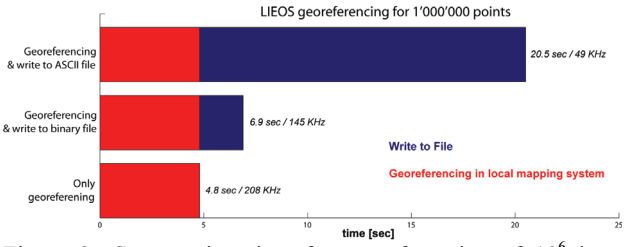

Figure 3: Computation time for georeferencing of 10^6 laser measurements.

3.2 Scanning progress presentation

The computation of RT quality indicators used for display (i.e. GPS positions and RT swath borders and laser range all at 1 Hz) requires very little computation-time. Within the whole process, the most important latency is introduced by the drawing and rendering of the data on the map (i.e. color-coded dots for GPS positions, polylines for swath, etc.). Although we use an optimized display refreshment algorithm where only the part of the map affected by changes is re-drawn, important time delays due to the graphical refreshment can occur. They can be as long as 1s, especially when pan or zoom function is used in parallel, thus requiring an entire screen update. Nevertheless, these latencies are principally hardware-dependent and could be reduced by using e.g. computers with more powerful graphic boards.

4. TIME-DELAYED COMPUTATION

During transition flights or between flight lines the algorithms evaluating data completeness (basic QC) and quality (advanced QC) are started. In the current implementation, these algorithms are executed sequentially within a separate thread called LIAN (LIdar Analyze module). The execution of this thread can continue in parallel with the on-line computation, however, its priority is reduced and the calculation slower. In the benchmarking we will therefore consider the 'fare case' when the thread runs under normal priority between the flight lines. Also, for the flight management it is preferable to provide all quality relevant information as fast as possible.

4.1 No optimization (case A)

Fig. 4 depicts the computation-time of LIAN for a full quality analysis (using the default LIAN settings) on a point-cloud with 500'000 points for different optimization scenarios. The lowest horizontal bar indicates the needed time when no algorithmic optimizations are performed. The upper bars indicate the processing time for the cumulated optimization steps. For the scenario without code optimization, Fig 4.A illustrates that the basic steps for quality control, such as data reading and

filtering, density grid computation, extent and gap detection and DSM computation, require only very little computation-time (ca. 5s). The predominant part of the computation-time is used for the more complex quality control, such as ground classification (ca. 65s) and error propagation (ca. 35s). In the sequel, several strategies to reduce the computational burden of the two latter processing steps are presented.

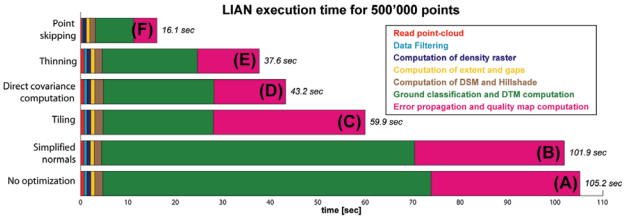

Figure 4: Computation-time for LIAN for point-cloud with 500'000 points.

4.2 Aggregated normal computation (case B)

Both the analysis of the scanning geometry and the ground classification algorithm require the computation of local normals for every point. Within LIAN, this is performed by computing the local covariance matrix for the points within a certain neighborhood N_p of size k and subsequent principal component analysis (PCA), (Schaer et al., 2007). If the computation is rigorous, the covariance matrix and the PCA have to be performed for every single laser point. However, in cases where the local curvature is low, the change in normal vector orientation for neighboring points remains minimal. Fig. 5 illustrates a method taking benefit of this property: If the local curvature Mcc of a point $\mathbf{p}_i$ is smaller than a certain threshold Mcc_{max} the computed normal $\mathbf{n}_{pi}$ can be assigned to the other points that are within a certain distance d_{max} to the original point. Thus, for all points within the small neighborhood no covariance computation is necessary anymore. For the example presented in Fig 4.B the reduction of computation-time is fairly low (time reduction of 3.2s or approximately 3%). However, for datasets with smooth topography the time reduction can reach up to 20% of the total processing time.

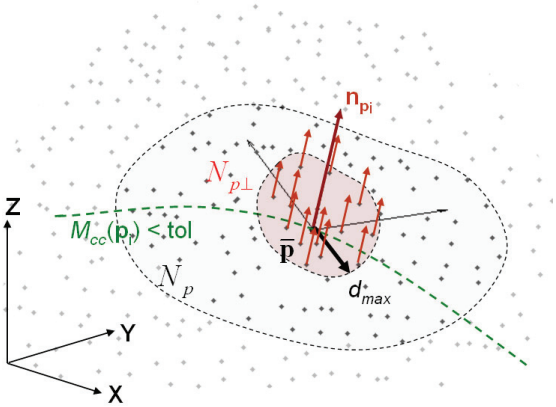

Figure 5: Local normal computation by aggregation.

4.3 Data tiling (case C)

Most algorithms used in the ground classification and error propagation step have a non-linear relationship between the size n of the dataset and the computation-time (see e.g. red line in Fig 6. For instance, the construction of a kd-tree for spatial indexing, necessary prior to any data query, has a logarithmic growth in computation-time (O (n log n)). In order to bound the time for spatial indexing and data querying, LIAN implements a dataset tiling procedure, where the point-cloud is subdivided into regular data blocks (typically 50'000 points) based on the timestamp (Fig. 7). To avoid incoherent results at the tile borders, the tiles are defined with a certain overlap (typically 5000 points). Subsequently, the different algorithms (i.e. ground classification) are applied tile-by-tile and the point-cloud is only merged at the end of the process. This procedure allows keeping the computation-time linearly proportional to the size of the dataset (see dotted line in Fig. 6). The therewith achievable increase in efficiency can also be seen in Fig 4.C, where the computation-time is reduced by more than 40% when implementing the tiling.

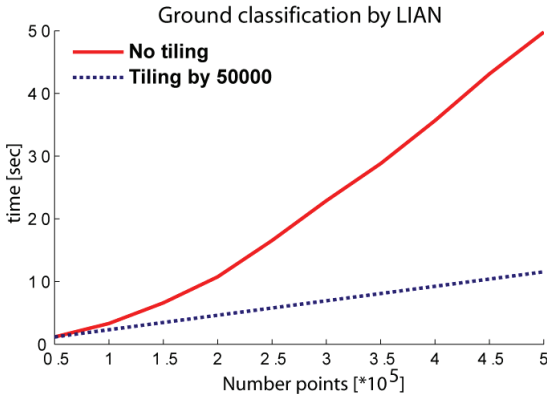

Figure 6: Computation time for ground classification without and with tiling.

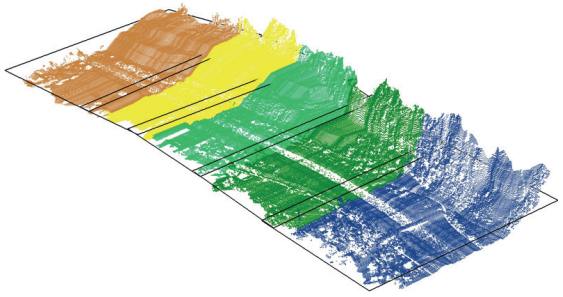

Figure 7: Example of point-cloud tiling by a timestamp.

4.4 Factorization of covariance computation (case D)

For the error propagation, LIAN applies functional covariance propagation as described in (Schaer et al., 2007) to compute the *3x3* covariance matrix for every laser point. This requires performing matrix multiplication per point of complexity $O(2n^2p)$, where n=3 and p=14. As explained in the same publication, the construction of the quality q_i-indicator is based on the diagonal elements of the covariance matrix. Hence, the off-diagonal elements are of no use and their computation is not necessary. This strategy is achieved by performing a complete factorization and aggregation of repetitive terms for the matrix multiplication. Expressing these terms directly (as simple multiplications) for the matrix elements of interest reduces the computational burden for the covariance matrix estimate by more than 90%. For the total processing chain, this result in further reduction of the computation-time of almost 30% (see Fig 4.D).

4.5 Selective data thinning (case E)

Within LIAN, the individual quality indicators q_i combining the influence of navigation ($\sigma^{nav}{}_{xy}$, $\sigma^{nav}{}_{z}$) and scanner geometry ($\sigma^{geo}{}_{xy}$, $\sigma^{geo}{}_{z}$) are computed for every laser point (Schaer et al., 2007). Based on this information, a quality map of a given cell-size (typically 1-2 m) is generated. Hence, applying the full error propagation to a dataset with higher sampling rate than the output quality map is not optimal, because it does not get used. Furthermore, the density of the point-clouds is often non-homogeneous as depicted in Fig. 10.A, where the scan lines are either stretched apart or squeezed together due to the variation of attitude and forward velocity. The selective thinning algorithm implemented in LIAN overcomes this problem by removing points that are within a certain spherical neighbourhood of the query point (Fig. 10.B). This has several benefits: First, the amount of data that has to be fed into the propagation engine is strongly reduced. Second, the dataset is homogenized and has a sampling rate close to the desired cell-size for the quality map. For the example illustrated in Fig 4.E, the selective thinning reduces the computation-time by more than 5s, which represents an improvement of about 13%.

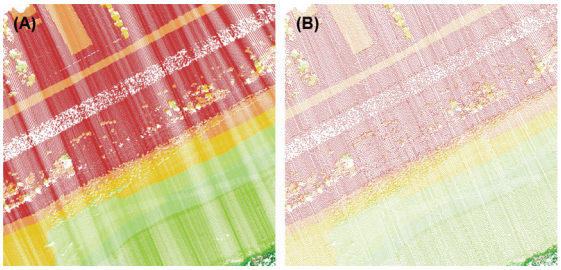

Figure 2: Homogenization of point-cloud density by selective thinning: (A) Original dataset, (B) Thinned dataset (color-coded by elevation).

4.6 Data skipping (case F)

For the purpose of in-flight quality assessment, processing the ALS data at the full data rate is not implicitly required. The data skipping can already occur directly in the georeferencing step. However, the performance evaluation presented in Sec. 3 has proven that point-cloud generation algorithm can handle the data rates of up to 100 kHz without problem. Thus, it is advisable to perform the data skipping only when reading the point-cloud data into LIAN. This has the substantial benefit that the point-cloud computed in-flight is complete and could directly be used in the PP-step when RTK-ALS method is used (Skaloud et al., 2010) . Fig. 4.F highlights the reduction in computation-time when reading only every third laser point (skip-factor 3) to LIAN from the 10 kHz laser data of Scan2map. In comparison to scenario E this speeds up the computation further by 21.5s or 57%, respectively. However, the reduction in computation-time via this approach comes together with a loss of spatial data resolution and probably also with the reduction of the assessment accuracy. Hence, the skipping factor has to be selected judiciously so that the computational efficiency is achieved without loosing the pertinence of the data accuracy analysis. If the flying height, carrier speed and PRF of the laser are known a priory, the achievable point density can be predicted. Coupling this information with the program's settings for mapping and evaluation (i.e. cell-size of density grid, DSM, DTM and quality map) the optimal skipping value can be determined prior to every mission. In the particular case of Scan2map, the typical skipping factor is between 2 and 4.

Tab. 1 assesses the compliance of three LIAN result grids (density grid, DTM grid and quality grid) computed once with configuration Fig. 4.A (no optimization applied) and once with configuration Fig. 4.F (cumulated optimizations). Although the computation-time between the two scenarios is reduced by a factor 6.5 (from 105.2s to 16.1s), the differences for the density grid and the quality grid are minimal and irrelevant for the purpose of the in-flight quality assessment. As the extent and gap computations are based on the density grid, their validity is not altered by initial point skipping.

On the other hand, the DTM grid is more affected by the quality degradation due to data skipping. As the amount of initial points is drastically reduced, the ground classification in sloped areas becomes problematic. Accordingly, the DTM accuracy principally suffers in those areas.

Datasets	Grid difference No optimization - Cumulated optimizations (B-F)			
	$\overline{\Delta Z}$ [m]	σ_Z [m]	RMS_Z [m]	$\lvert\Delta Z_{max}\rvert$ [m]
Density grid	-0.003	0.16	0.16	1.04
DTM grid	-0.11	0.35	0.37	15.7
Quality grid	0.002	0.09	0.09	9.1

Table 1: Comparison of LIAN results computed without and with all optimizations, including skip factor 3.

5. CONCLUSIONS AND PERSPECTIVES

The computational tasks related to real-time laser point cloud generation and processing were analyzed in terms of execution speed. Several strategies have been suggested to improve the computational efficiency in the bottle neck of the process. This includes methods like aggregated normal computation, tiling, covariance factorization and data thinning. Their successful implementation into a software module (LIAN) has been confirmed by considerable experimental testing. The performance evaluations have shown that real-time georeferencing can be performed for a LiDAR with a pulse repetition rate up to 150 kHz. By applying initial data skipping the computational performances of the software are scalable. The basic quality control operations, such as data extent and gap detection, are very fast (i.e. require less than 5 s for 0.5 million of points) and therefore can cope with higher data rates. The advanced quality control functionalities, such as ground classification and error propagation require more computation time (i.e. about 25 s for 0.5 million points). Their applicability may be limited for systems with data rates exceeding 50 kHz, although it could be argued that such computational rate is still sufficient to create relevant feedback to the operation when allowing certain, yet representative, aggregation/resolution in time and space.

Even tough the use of more powerful computers (i.e. industrial racks) could certainly drastically raise the speed of real-time processing; the computation-time remains the limiting factor for the applicability of the thorough in-fight quality-control concepts. Especially the point-cloud data querying and derivation of variance information (i.e. data classification, error propagation) require high computing power. Employment of Graphics Processing Units (GPUs) present in modern graphics

cards might be a solution as their parallel processing capabilities are tremendous. For example (Garcia et al., 2008) demonstrates that parallel processing using a standard graphic card accelerates the k-nearest neighbor search (one of the most frequent operations in LIAN) up to a factor of 120. Adapting the software to parallel processing would open the field to in-flight data analysis for laser systems with much higher pulse-repetition rates or even to those recording the full-waveform signal.

6. ACKNOWLEDGMENT

This work was mostly funded by the Swiss Commission for Innovation (CTI/KTI Project 7782 EPRP) in collaboration with Swissphoto AG.

REFERENCES

El-Sheimy, N., Valeo, C. and Habib, A., 2005. Digital terrain modeling. Artech House.

Garcia, V., Debreuve, E. and Barlaud, M., 2008. Fast k nearest neighbor search using gpu, CVPR Workshop on Computer Vision on GPU, Anchorage, Alaska, USA.

Schaer, P., Skaloud, J., Landtwig, S. and Legat, K., 2007. Accuracy Estimation for Laser Point Cloud Including Scanning Geometry. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 36(5): 8.

Schaer, P., Skaloud, J. and Tome, P., 2008. Towards in-flight quality assessment of airborne laser scanning. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 37(B5): 851-856.

Skaloud, J., Schaer, P., Stebler, Y. and Tome, P., 2010. Real-time registration of airborne laser data with sub-decimeter accuracy. ISPRS Journal of Photogrammetry and Remote Sensing, 65: 208-217.