LIBRARY CONCEPT AND DESIGN FOR LIDAR DATA PROCESSING

Nicolas David, Clément Mallet, Frédéric Bretar

Institut Géographique National (IGN) – MATIS laboratory
2-4 avenue Pasteur 94165 Saint-Mandé, FRANCE
http://recherche.ign.fr/labos/matis
firstname.lastname@ign.fr

KEY WORDS: lidar data, Point Cloud, Design, Data Management System, Processing

ABSTRACT:

Airborne Laser Scanning (ALS) is nowadays a very popular technology providing accurate altimetric data for remote sensing and mapping purposes. Therefore, many algorithms have been developed so far to process these data, depending on the application. Nevertheless, for researchers, it is still a challenging task to handle large amount of heterogeneous data and adapt them for their specific aim and processes. This paper reports the thoughts and the strategy developed by the MATIS laboratory of the Institut Géographique National (IGN) about an efficient lidar library design in order to tackle these issues. The specification of an efficient and versatile lidar file format is first discussed. The standard and current lidar file formats are first reviewed and a new one, dedicated to raw data processing with high feature modularity is presented. Besides, existing code components and libraries are reviewed with regard to their compatibility for research development. Modularity, availability and license conditions are here the main selection criteria. Then, differents strategies for large data set handling are summarized and extended by a new solution, both based on lidar strip and raw sensor topology. These workflows are illustrated through a UML activity diagram dedicated to 2D spatial query. Finally, the current status of the implementation of this federative software as well as the perspectives of development are sketched.

1 INTRODUCTION

In the last decade, airborne lidar systems have become an alternative source for acquiring altimetric data. Such devices deliver a reliable, fast and accurate representation of terrestrial landscapes through unstructured 3D point clouds. These systems are based on the recording of the time-of-flight distance between an emitted laser pulse and its response after a reflection on the ground. Points cloud are georeferenced with an integrated GPS/INS system and the amplitude of the reflected laser pulse give intensity information. Moreover, a single laser pulse can provide multiple returns (echoes) which correspond to different measured altitudes.
Lidar data management and processing are today well-known tasks; many softwares and tools are now available for that purpose. Thus, lidar data filtering and Digital Terrain Model (DTM) generation on large areas and data sets can be carried out with good performance with many commercial (Soininen, 1999; AppliedImagery, 2008) or research softwares (Hug et al., 2004; Inpho and TU-Vienna, 2008). If such solutions fit to ”end user” expectations, it is no longer the case for research purposes when scientists expect customisable tools or source code to test their own algorithms (Beinat and Sepic, 2005).
On the one hand, most of these solutions are no freely available and the description of the implementation is subsequently never provided. Different strategies exist concerning the lidar data management, and many questions remain: which spatial indexing is supported by lidar file database –raw laser strips, pretiled files (Chen, 2007), both? Which data is used for lidar processing – point cloud or interpolated raster? It is possible to add or remove features from the data when new information is available (intensity, classification. . .)? Which neighborhood topology has be chosen for RAM lidar data structure –quadtree, octree, cells or database?
On the other hand, the conception of an adaptive, efficient, easy-to-use lidar library with good interoperability toward well-known lidar softwares is not obvious. Few articles have been dedicated to the design of lidar libraries, with appropriate data structure and workflow (Hug et al., 2004; Chen, 2007) and even less to their implementation.
Since no existing software can fit the research aims of the MATIS laboratory and in order to not to depend on commercial develoment toolkit – GIS or remote sensing– it has been decided to elaborate a versatible lidar library, as a basis of lidar processing sofware. The file format is flexible, large lidar data sets can be efficiently handled, one can select its workflow and even switch between existing ones and finally researchers can easily “plug” their algorithms. A raw sensor file format has been designed for the lidar sensor data georeferencing and intensity calibration issues.

The specification of an efficient and versatile lidar file format is first discussed in Section 2. The common lidar formats are analysed, and a new format, dedicated to raw data processing with high feature modularity is presented. The global architecture of the software, designed to solve most of the raised issues, is detailed and several additional useful *open-source* libraries are introduced in Section 3. The relevant workflows and their application to efficient queries on lidar data are outlined in Section 4. Finally, the current status of the software implementation is presented and the perspectives of development are drawn.

2 FILE FORMAT

A major issue when designing a library as a basis for managing and processing task is to clearly define inputs and outputs. For ALS data, the specification of the input file format is therefore a crucial task and has been first tackled in this study to design the library. On the one hand, such format has to be compatible with existing standard file formats. On the other hand, it must be flexible and must take into account more information than the typical $\{(x, y, z), intensity, classification\}$ quintuplet.

2.1 Standard and current lidar file formats

The definition of laser data is still a work in progress. The different solutions strongly depend on lidar manufacturers, software

companies, data providers and, end users. Most of the existing laser data file formats are summarized in (Samberg, 2007).For that purpose, the LAS data exchange format of the American Society for Photogrammetry and Remote Sensing (ASPRS) has been designed in order to make the exchange of lidar data more convenient (widely used, weak storage volume, acquisition metadata). The last version of the ASPRS LAS standard is 2.0 (ASPRS, 2007), improving the last 1.1 version (ASPRS, 2005), and has become a *de facto* standard. It is now widely used by ALS community.
Even if such format is suitable for storing and handling Terrestrial Laser Scanning (TLS) and other 3D digitalization lidar data, the PLY format is used by the Computer Vision community, interested in 3D point clouds for graphical models. PLY has been defined by the Standford University and is one of the most current interoperable format for 3D model (Turk and Mullins, 2006). It aims at providing a simple format but enough general to be useful for a wide range of models. Models could be based on point clouds or on faced items, (*e.g.*, Triangular Irregular Network).
Nevertheless, there are still many lidar file formats which differ from standards. Numerous ASCII formats exist, depending on the information available. It could be classical XYZ file or an extended version with intensity, XYZI. Other features could be added as ASCII columns and in any possible order. The most frequent features are the classification field, the GPS acquisition time and the pulse number. In addition, a file can represent different things, a strip (raw acquisition geometry) or a tile of lidar data. And finally, as detailled in section 4, some formats also propose spatial indexing (McGaughey, 2007).
Finally, it becomes clear that a modern lidar library should be able to manage this two standards but also to supply extended ASCII import and export functionnalities for ensuring compatibility with toolkits and current software packages.

2.2 Raw file description

A common point of every lidar format is the use of cartesian coordinates expressed in a given datum, with minor exception for TLS in LAS 2.0 specification which accepts range point encoding (r, θ, ϕ). Several lidar processing tasks (intensity calibration, fine strip registration, full-waveform echo extraction) could be carried out more easily easily performed when applied directly on the raw sensor geometry. Therefore, it has been decided to investigate a file format related to the sensor geometry. In addition, a XML metadata file is used to describe the file as well as sensor information. The format is then splitted into two specific parts (figure 1).

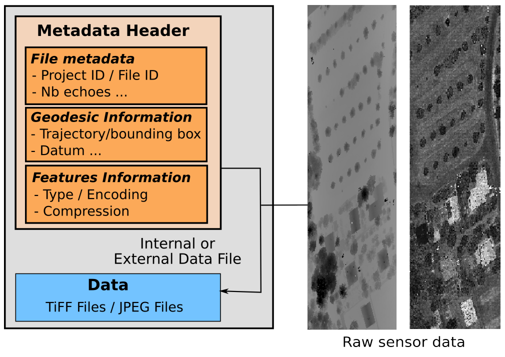

Figure 1: Lidar data file structure divided on two specific parts. **Left**: metadata file. **Right**: examples of data within raw sensor topology.

2.2.1 XML metadata file :

XML is a general-purpose specification, widely supported and extensible since it allows users to define their own data markups. Thus, it permits to add new metadata, when available or computed (*e.g.*, the real point density of a survey is not known beforehand). Moreover, XML facilitates the sharing of lidar metadata between different information systems. XML files can be displayed within a simple text file, Web browsers, GIS softwares. . . and even can be formatted for a Webcatalogue purpose (Consortium, 2008). Indeed, the XSL (eXtensible Stylesheet Language) can be used to describe how XML files or parts of a XML document are to be transformed and visually formatted.
As for the LAS file header, some metadata fields included in the file are: file ID, project ID, number of echoes, strip bounding box. For the registration of raw data, the trajectory (INS and GPS) and the reference coordinate system are required. Several specific fields as the simplified convex hull and the trajectory of the strip, are consequently added to improve query efficiency (see Section 4). Besides, it has been decided to adopt a modular attribut format. Therefore, each feature intensity, classification or some others owns its specific XML markup, with a complete feature description: which type (*e.g.*, intensity)? which encoding (*e.g.*, `char/float`)? Which file compression is used (*e.g.*, LZW/ZIP)? . . . .

2.2.2 RAW data :

Concerning the raw data file architecture, it has been decided to use a multi-channel image for each feature (see figure 2). One axis of the image corresponds to the sensor acquisition angle θ and the others represents the scan line number (which can be considered as the acquisition time). Their bound values are specified in the XML metadata file.
The first image is the range image and the others correspond to feature images. For each image, the first channel is linked to the first echo of the current lidar pulse, the second one to the second echo and the last one to the greatest number of detected echoes. All images have exactly the same structure but can be encoded differently (*e.g.*, intensity with `char` or `short`, echo width with `int` or `float`).
Each first channel contains dense information, thus raw data could be preserved. However, other channels are bounded to hold sparse data. Therefore, compressed algorithm as LZW could be use to store this channel, depending on the feature (compression algorithms are not always adapted to all encoding types).

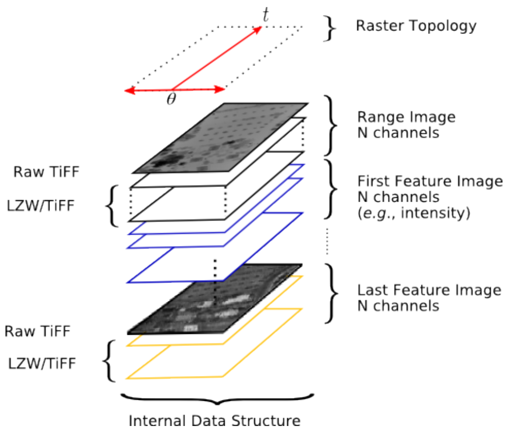

Figure 2: Lidar data within raw sensor structure. Each feature is linked to a N-channel image. Image indexes are scan line number (as time) and angle acquisition θ

3 GLOBAL LIBRARY DESIGN

3.1 Use case

The current design of the library aims at tackling some clearly identified use cases; some of them dedicated to specific tasks and other common to all tasks. The basic needs are described above:

- import and export data (Section 2.1);
- convert data from one geodesic coordinate system to another one;
- add/remove lidar feature;
- perform efficient search query on large data set;

Then, some specific components are data processing and display as:

- filter lidar point cloud;
- generate DSM/DTM (Bretar, 2007);
- display the data on 2D or 3D;
- perform cross-sections on point cloud jointly with DEM;

Theses specific tasks depend on the previous common tasks. They are directly "client" of the core library components.

3.2 Useful libraries overview

The first step when developing a software is to find open-source libraries matching with our needs. Hopefully, open-source geomatic community is active and many libraries are available for GIS, geodesic, geometric and display purposes.
First, concerning GIS components, the PROJ4, OGR, and GDAL libraries have been studied. They are used on the core of nearly all open-source GIS.

- PROJ4 is dedicated to cartographic projection and datum transform (PROJ team, 2007). A wide range of projections are supported in the PROJ4 library. However, some other advanced geodesic features are required: georeference raw data and transform coordinates between vertical datums (*e.g.*, to support quasi-geoide).
- GDAL and OGR are libraries for GIS raster and vector data handling respectively (GDAL team, 2008). GDAL can be used for raster DEM handling but large lidar point clouds do not fit very well with the abstraction data model. Both of them depend on PROJ4 for cartographic projections.

Taking the drawback of GDAL/OGR into account, it has been decided to develop a similar library but specifically dedicated to handle lidar data sets. PROJ4 is kept for coordinate transform tasks.
Besides, LAS and PLY formats have to be supported. Consequently, LAS and PLY librairies will be chosen (both have numerous dedicated open-source libraries).

- <u>PLY</u>:
  - plytools (Turk and Mullins, 2006) and RPly are both coded with C language;
  - libply (Lagae, 2007) is coded with a modern C++ design: more flexible than plytools but also more difficult to handle.
- <u>LAS</u>:
  - LAStools is an earlier open source code for LAS 1.1 format (Isenburg et al., 2006; Isenburg and Schewchuck, 2007).
  - LibLAS (Butler et al., 2008) supports the LAS 1.0 and 1.1 specifications but still needs improvement for the LAS 2.0 implementation. It is a refactoring of LAStools.

Since upper flexibility is required, libply and LibLAS have been chosen for that purposes.
Concerning the raw sensor format (section 2.2), it is planned to use the common TiFF or JPEG libraries for managing raw data and a standard XML library (TinyXML, Xerces) for the metadata file.
Some geometrical capacities for neighborhood search and internal (RAM) data management are also required:

- (Refraction and GEOS team, 2007) library has numerous functionnalities but only in 2D .
- (CGAL, Computational Geometry Algorithms Library, 2008) is a comprehensive C++ library for geometrical computation, including both algorithms and data structures. The main drawbacks are a difficult source code handling and the fact that some components are only available under QPL (and not BSD or GPl/LGPL) license.

Despite its license term, CGAL has been chosen in order to increase geometrical capabilities of the lidar library.

3.3 Architecture Overview

The lidar library is splitted into distinct components: a "kernel" for low-level object modelling, an import/export (I/O) library, a topology module, a geodesy wrapper based on PROJ4 library, a processing unit including algorithms developed in the MATIS laboratory and, a display module (see figure 3). This order corresponds to the priority of each component of the library developments.

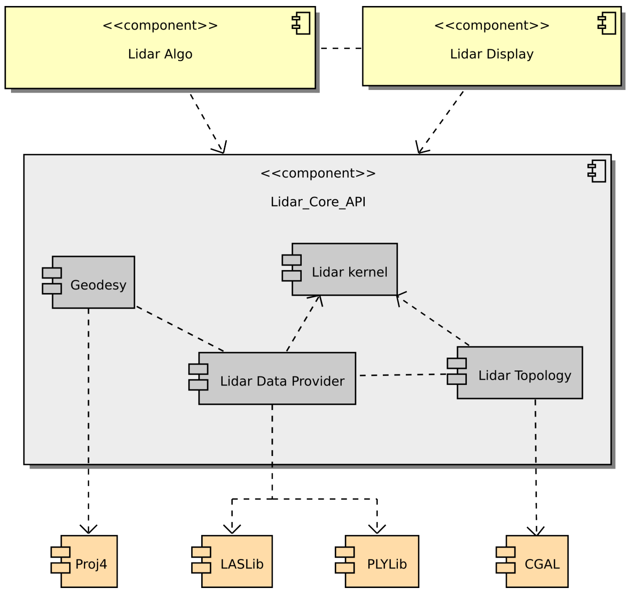

Figure 3: Library UML component view. **Top:** specific task component depending on core API. **Middle:** core component. **Bottom:** external component.

- The "kernel" component is dedicated to internal data structure. It contains an abstraction model for lidar data, more precisely a class for single lidar echo and a attribut structure, and also a container class for collection of lidar echoes (this can be view as unstructured point cloud). A query encapsulation class is also included in kernel.
- The I/O library has similar functionnalities as GDAL (but for lidar and not raster data), and master specific tools, like the metadata computation or the conversion and reprojection. It is a client of the kernel component and uses its abstraction model as a bridge between different file formats.

- The topology module implements the internal ordering and search data structure *e.g* the quadtree and cell topology structure as well as an interface class for an efficient and simple neigborhood search.
- The geodetic component extends PROJ4 functionnalities. It is linked to PROJ4 and contains module for coordinate transform, from height to altitude and from cartesian to sensor datum.
- The processing unit contains lidar filtering algorithms and DEM/DTM utilities developed in the MATIS laboratory: the generation and resampling of raster DEM and, the DTM regularization. For more details see for instance (Bretar, 2007)
- The display module is divided in three parts. A 3D viewer to easily visualize the data, a 2D (planimetric) interactive interface for the survey visualization and query.The point clouds can be displayed in 2D or 3D with a specific feature selected by the user (classification, intensity... and height is available). Finally, it is also possible to plot cross-sections within the point cloud, in conjunction with the DEMs or DTMs in order to assess the quality of the algorithms (*e.g.*, filtering or DEM/DTM generation).

4 WORFLOW AND QUERY APPLICATIONS

In order to efficiently process large lidar data sets, a data workflow strategy must be defined. A significant and recurrent need when implementing lidar algorithms is to perform spatial queries. A spatial query could be a coarse 2D/3D polygonal selection within the point cloud or a fine neihgborhood search. Therefore, a lidar workflow have to tackle this issue. Both external (file based) and internal (RAM data) topology structures are used for spatial query strategies.
The management of the external topology depends on the files processed by the chosen algorithm. A first spatial indexing could be made on the file object level. Each file contains its bounding box or is a pretiled file with exactly known convex hull (Chen, 2007). By testing for each file whether or not its bounding box or its convex hull intersects the 2D spatial query, this indexation speeds up the query. Besides this low level indexing, some more accurate spatial indexes exist which enable to load only regions of interest (ROI) within each file (McGaughey, 2007). Moreover, it should be noticed that using tiled files and current fine spatial indexing increases the pre-processing step.
Then, once the data are loaded in memory, the algorithms should still benefit from the internal data indexing structure. Quatree and 2D cells point indexing are the more common used on 2D spatial indexing (and could be extended in 3D by respectively octree and 3D cells). These dynamic structures increase the efficiency of the neigborhood search algorithm and the data management for adding and removing echoes. A database architecture could also be used both to manage fine external topology and internal data structure (Höfle, 2007).

4.1 Raster based strategy

One of the main common strategy solution to manage lidar data is to rasterize raw data. This lidar workflow processing can be detailled in two major steps. The raw data are resampling on a range image and image processings are made with this raster data. Major advantages of this workflow are its simplicity and the fact that all the classical image processing algorithms can be used. Current raster data are first and last echo DEM and eventually some density or intensity images. With only raster data, spatial queries become easier and are now well-known processes. This workflow also benefit from very efficient raster cache data management and streaming algorithms used by modern image processing toolkits. Such functionnalities are not yet avaible for point cloud processing but are increasingly studied (Wand et al., 2007).

4.2 Standard tile workflow

Nowadays, most of lidar softwares prefer handling directly point cloud data. Nevertheless, some key ideas of raster based strategy has to be retained, especially inside the tiled workflow (Chen, 2007; Lausten, 2007). Indeed a strategy extended from raster to 3D point cloud is the tiled file organisation. Raw data are recomputed from lidar strips to predefined squared file data, *e.g.*, tiles. A tile includes all echoes of a given lidar survey into a precisely defined bounding box. Consequently, for a spatial query, a simple test on bounding boxes quickly enables to know which files have to be processed in order to find the required part of the lidar point.
An extension of this strategy is to spatially index the points included in a tiled file on a gridded topology. Such workflow has been used and well explained in (McGaughey, 2007), with the LDA format dedicated to the FUSION/LDV software.

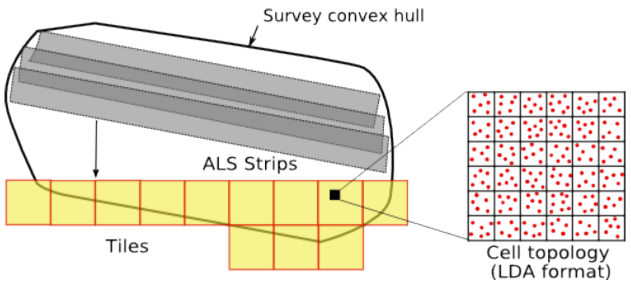

Figure 4: Tile workflow strategy. **Left:** raw strips (above) are tiled (below). **Right:** tiles are spatially reorganised.

To summarize the pre-processing step:

1. All lidar strips (1 strip=1file) are reorganized on tiled data files with well defined bounding box.
2. Each tile is gridded on square cells with fixed resolution. Each cell is indexed by an (i, j) position like pixels on image.
3. Lidar points of each tiles are reordered within the file. A new tile indexing begins with the point inside cell $(0, 0)$, then cell $(0, 1) \dots (N, M)$.
4. An image of index pointers is generated. Pointer at index (i, j) leads to the tile index of the first lidar points contained on cell (i, j).

When lidar data are tiled and indexed with cells, the workflow for query point becomes (see figure 5 for details):

1. Get all tiles whose bounding box intersect the query bounding box.
2. For each selected tile, get cells inside the 2D query:
   - 2.1 compute the bounding box of the intersection between tile and query (BBoxI);
   - 2.2 get all index (i, j) of cells included in BBoxI;
   - 2.3 load all lidar points contained on selected cells using the image of index;

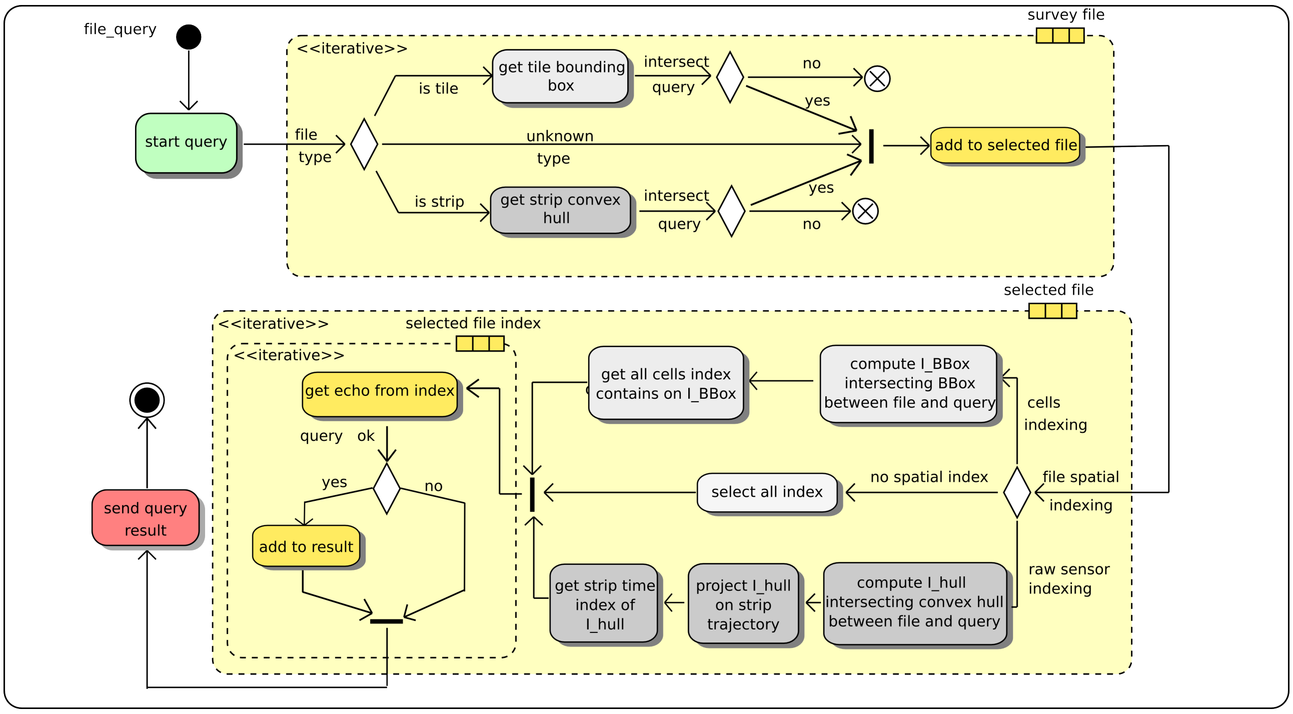

Figure 5: Global query activity diagram workflow with different files and echo spatial indexing.

4.3 Strip file management and raw sensor topology

If the strip information have to be kept on tiled workflow, it becomes necessary to add new feature to each point (strip ID, acquisition range and angle). As a consequence, issues related to sensor acquisition as intensity calibration or fine strip registration are made more difficult. To avoid such drawback, it has been decided to investigate in depth the raw sensor topology.
Since lidar strips are spatially long and oblique rectangles, their bounding boxes are not suitable to finely select which strips are concerned by a spatial query (figure **??**a). Therefore, a simplified convex hull is computed. Convex hull can be computed by some triangulation or α-shape geometrical algorithm (with CGAL) or simplier by detecting jumps or inflexion points between two lidar scan lines. In addition, the number of pulses by scan line is always the same along the strip and such information can be used to robustify the convex hull detection.
Besides, it has been considered that a raw sensor file has an inherent 2D topology. More precisely, raw measurements can be indexed by their scan line number (timestamped) and by the mirror angle at the acquisition time (figure **??**b).

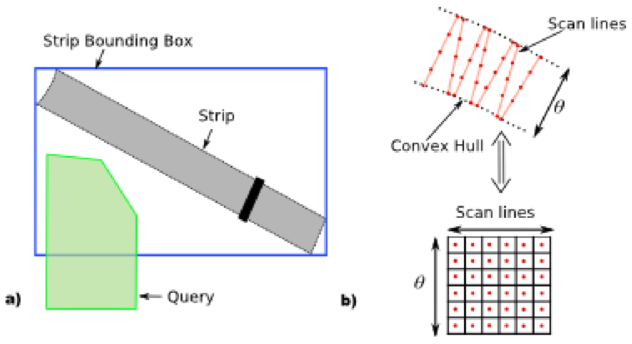

Figure 6: **a):** Query intersecting only strip bounding box and not convex hull. **b):** raw sensor data.

If the acquisition time and the approximate trajectory have been kept, the 2D indexes inside a spatial query can be approximated in raw sensor topology by the workflow below (figure**??**):

1. compute intersection of the strip convex hull with the query;
2. project intersection on the approximate trajectory;
3. order projected point according to their acquisition time. and extract maximum and minimum times (with some security margins due to approximations);
4. load all points contained between these two acquisitions time;

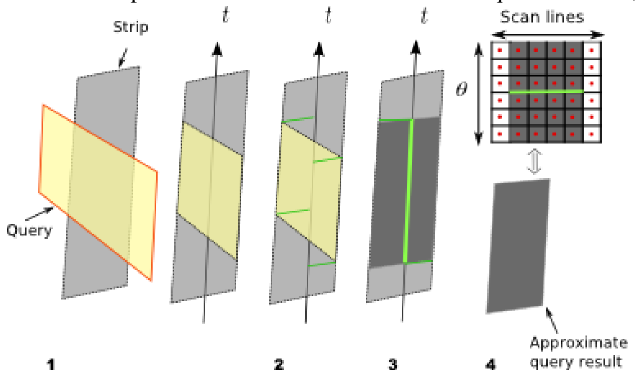

Figure 7: Steps of a spatial query processing with raw sensor topology

The first step on query processing is to intersect the query convex hull with the strip convex hull. To speed up this computation, the convex hull can be generalized. One possible generalization is to keep only the oriented bounding box of the strip (Höfle, 2007).

5 DEVELOPMENT STATUS AND FUTURE WORKS

The implementation of a library for lidar data handling and processing is an on-going process.
The kernel component has been fully developed since this is the basis of all the detailed library. The IO wrapper architecture is

tested with the ALS format wrapper and one of the previous format developed in the MATIS laboratory (for data and algorithm compatibility). The PLY wrapper is under development. Concerning the topology part, first implementation of spatial query and neigborhood search strucure have been made and are on a testing phase. First studies to add height transform within PROJ4 are finished and the final implementation is already planned. All other components are still prototypes.
Concerning future developments, adapt our algorithm on the new lidar library-API has been defined as next implementation priority.
Then, it is planned to develop and test our raw sensor topology strategy and, on parallel, to study whether or not a high abstraction level component for lidar survey management is needed. Some other open-source toolkits like QGIS and OSSIM should probably be investigated for this task.
Other modelling works are also well defined. First, automatic security margin computation will be deeply studied to improve the effiency and to robustify the raw sensor topology query workflow. Next, it could be of interest to adopt Geographic Markup Language (GML) normalisation from Open GIS Consortium (OGC) for our own XML metadata file.

6 CONCLUSION

Lidar data management is a challenging task with only young background compared to image processing and other remote sensing data. The increasing abilities of multi-echo data offer great opportunities to model a modern lidar data management architecture and implement a modular library, gathering efficient open-source libraries and taking previous lidar community experience into account.

References

AppliedImagery, 2008. QTModeler 6.0.5 : Quick Terrain Modeler. `http://www.appliedimagery.com/`.

ASPRS, 2005. ASPRS Lidar Data Exchange Format Standard version 1.1. Technical report.

ASPRS, 2007. ASPRS Lidar Data Exchange Format Standard version 2.0. Technical report.

Beinat, A. and Sepic, F., 2005. Un programma per l'elaborazione di dati LIDAR in ambiente Linux. In: 50^o Convegno Nazionale della Società Italiana di Fotogrammetria e Topografia, Palermo, Italy.

Bretar, F., 2007. Processing fine Digital Terrain Models by Markovian Regularization from 3D Airborne Lidar Data. In: IEEE International Conference on Image Processing, Atlanta, USA, pp. 125–128.

Brovelli, M., Cannata, M. and Longoni, U., 2002. Managing and processing LiDAR data within GRASS. In: Open source GIS GRASS user conference, Trente, Italy.

Butler, H., Loskot, M. and Vachon, P., 2008. Liblas, a C++ library for reading and writing LAS LiDAR data. `http://liblas.org/`.

CGAL, Computational Geometry Algorithms Library, 2008. `http://www.cgal.org/`.

Chauve, A., Mallet, C., Bretar, F., Durrieu, S., Pierrot-Deseilligny, M. and Puech, W., 2007. Processing full-waveform lidar data: modelling raw signals. In: IAPRS, Vol. 39 (Part 3/W52), Espoo, Finland, pp. 102–107.

Chen, Q., 2007. Airborne Lidar Data Processing and Information Extraction. PE&RS 73(2), pp. 109–112.

Consortium, W. W. W. W., 2008. XML. `http://www.w3.org/XML/`.

GDAL team, 2008. GDAL 1.5 : Geospatial Data Abstraction Library. `http://www.gdal.org/`.

Graham, L., 2007. Las specification version 2.0 (final draft). Technical report.

Grass Development Team, 2006. Geographic Resources Analysis Support System GRASS Software. Technical report, Trente, Italy.

Höfle, B., 2007. Detection and Utilization of the Information Potential of Airborne Laser Scanning Point Cloud and Intensity Data by Developing a Management and Analysis System. PhD thesis, University of Innsbruck.

Hug, C., Krzystek, P. and Fuchs, W., 2004. Advanced lidar data processing with LasTools. In: IAPRS, Vol. 35 (Part B2), Istanbul, Turkey, pp. 832–837.

Inpho and TU-Vienna, 2008. SCOP++. `http://www.ipf.tuwien.ac.at/products/produktinfo/scop/scop_dtm_sheet.htm`.

Isenburg, M. and Schewchuck, J., 2007. LAStools : converting, viewing and compressing LiDAR data in LAS format. `http://www.cs.unc.edu/~isenburg/lastools/`.

Isenburg, M., Liu, Y., Shewchuk, J., Snoeyink, J. and Thirion, T., 2006. Generating raster dem from mass points via tin streaming. In: International Conference on Geographic Information Science, Münster, Germany, pp. 186–198.

J. Lüthy and H. Ingensand and R. Stengele, 2005. Production suite for airborne data. In: Proc. of the 7^{th} Conf. on Optical 3D Measurement Techniques, Vienna, Austria.

Lagae, A., 2007. libply :a modern C++ library for parsing the ply file format. `http://www`.

Lausten, K., 2007. ENVI LiDAR Toolkit. Technical report.

McGaughey, R., 2007. FUSION/LDV : software for LiDAR data analyse and visualization. `http://forsys.cfr.washington.edu/fusion/fusionlatest.html`.

PROJ team, 2007. proj-4.6 : Cartographic Projection Library. `http://proj.maptools.org`.

Refraction and GEOS team, 2007. GEOS 3.0 : Geometry Engine - Open Source. `http://trac.osgeo.org/geos`.

Samberg, A., 2007. An Implementation of the ASPRS LAS Standard. In: IAPRS, Vol. 39 (Part 3/W52), Espoo, Finland, pp. 363–372.

Soininen, A., 1999. Terra Scan for MicroStation, user's guide. Technical report, Helsinki, Finland.

Turk, G. and Mullins, B., 2006. Plytools. `http://www.cc.gatech.edu/projects/large_models/ply.html`.

Visual Learning System, 2008. Lidar Analyst 4.2. `http://www.vls-inc.com/lidar_analyst.htm`.

Wagner, W., Roncat, A., Melzer, T. and Ullrich, A., 2007. Waveform Analysis Techniques in Airborne Laser Scanning. In: IAPRS, Vol. 39 (Part 3/W52), Espoo, Finland, pp. 413–418.

Wand, M., Berner, A., Bokeloh, M., Fleck, A., Hoffmann, M., Jenke, P., Maier, B., Staneker, D. and Schilling, A., 2007. Interactive Editing of Large Point Clouds. In: PBG07, Prague, Czech Republic.