Cluster Comput (2017) 20:741–748
DOI 10.1007/s10586-017-0759-x

Geometric primitive extraction from LiDAR-scanned point clouds

Nakhoon Baek[1] · **Woo-seok Shin**[1] · **Kuinam J. Kim**[2]

Received: 5 November 2016 / Accepted: 24 January 2017 / Published online: 4 March 2017

Abstract Recently, we have lots of LiDAR (light detection and ranging) data, for applications of high-resolution maps including geography, geology, forestry, and others. One of the current research and industrial issues is efficient ways of storing the LiDAR data itself, and also elegant ways of extracting geometric primitives from those LiDAR-scanned 3D point clouds. In this paper, we first analyze the characteristics of LiDAR data and tis storage schemes. Additionally, we present an efficient method to extract geometric primitives from those point clouds. Its implementation and results are also presented.

Keywords LiDAR · Light detection and ranging · Efficient processing · Geometric primitive · Point clouds

1 Introduction

LiDAR (light detection and ranging) is a surveying technology that measures distance by illuminating a target with a laser light [1–4]. Lidar is popularly used as a technology to make various kinds of geometric and geological data. In some cases, it is simply referred to as **laser scanning** or **3D scanning**, with terrestrial, airborne and mobile applications.

Typical LiDAR systems produce large-scale data sets, with lots of 3D sampling points. Sometimes, they refer these data point sets as **point clouds**. Figure 1 shows a typical airborne LiDAR acquisition system for large landscape-scale data sets. Figure 2 shows another terrestrial LiDAR acquisition system, for ground structures like buildings, rock formations, and others.

One of the fundamental problems with LiDAR data is that there are still no general-purpose, open standard for storing and analyzing LiDAR data. Thus, the LiDAR system developers should make decisions on the storing file formats and fundamental analysis methods.

Another problem occurs with the enormous number of points captured with LiDAR equipment. A typical large-scale point cloud consists of millions of points, which makes it hard to be loaded onto the main memory of commercial PC's. To efficiently use these LiDAR data with other applications including **GIS** (geographic information system) and others, extracting abstract information from the LiDAR data is strongly required.

In this paper, we first present a set of solutions for the LiDAR data handling. The widely used LiDAR file formats are also explained. Then, we show our method to extract geometric representation from cloud points, acquired from airborne LiDAR devices and also from terrestrial LiDAR devices. We also show efficient way of using existing general-purpose geometric algorithms and their implementation.

Our prototype implementation of the LiDAR framework was accomplished through combining the LiDAR file formats, our method of extracting geometric primitives, and the generic versions of geometric algorithms. We finally show the results of our LiDAR framework.

✉ Kuinam J. Kim
kuinamj@gmail.com

Nakhoon Baek
oceancru@gmail.com

Woo-seok Shin
mell03@naver.com

[1] School of Computer Science and Engineering, Kyungpook National University, Taegu 41566, Republic of Korea

[2] Department of Convergence Security, Kyonggi University, Suwon 16227, Republic of Korea

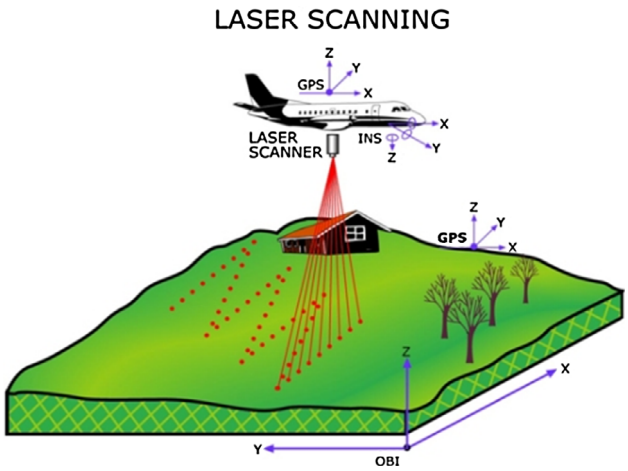


Fig. 1 A landscape-scale airborne LiDAR data acquisition system (courtesy of ASPRS)

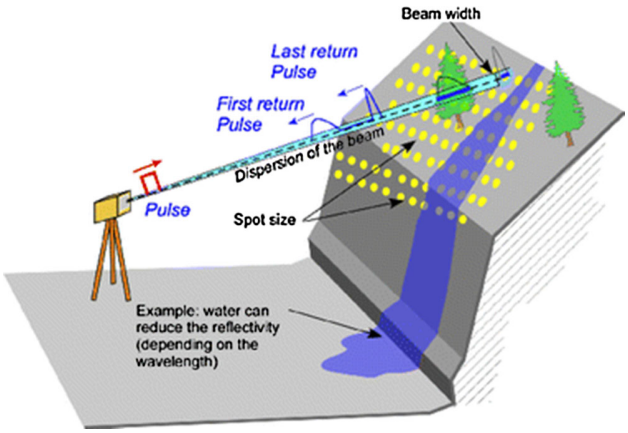


Fig. 2 Terrestrial LiDAR data acquisition system

2 LiDAR file formats

Since there are no dominant and/or open standard specifications for storing LiDAR data, there are various kinds of file formats such as XYZ, PTS, PTX, PCD, and others. Among them, we will show two of the most widely used LiDAR data file formats: **LAS** and **E57**. Our practical choices of E57 handling library is also explained.

2.1 LAS file format

The **LAS** file format (short for LASer) is a data format designed to store 3D point cloud data, especially for airborne LiDAR scanning devices. It was developed by the American Society for Photogrammetry and Remote Sensing (ASPRS) [5,6]. The LAS format is a sequential binary format used to store data from sensors and as intermediate processing storage by some applications.

Public Header Block
Variable Length Records (VLR)
Point Data Records
Extended Variable Length Records (EVLR)

Fig. 3 LAS file format

X, Y, Z (4 bytes for each)
Intensity (4 bytes)
Return Number (3 bits)
…
Scan Angle (1 byte)
…
Red, Green, Blue (2 bytes for each)

Fig. 4 An example point data record format stored in the LAS file

Figure 3 shows the global view of a LAS file format. Figure 4 shows an example format of single point data record stored in the LAS file, containing intensity, scan angle, color value. Those data are typically specialized to detect and extract features from the area-based LiDAR data.

The **libLAS** is a library for reading and writing geospatial data encoded in the LAS file format [7]. It consists of base library with multiple application programming interfaces available for programming languages like C, C++, Python as well as languages available in .NET Framework. Also, a variety of useful command-line utilities is provided for translating LAS files from one version of the LAS format to another, inspecting header information, and translating LAS data to and from text.

2.2 E57 file format

The **E57** file format is a compact, vendor-neutral format for storing point clouds, images, and metadata produced by 3D imaging systems, such as laser scanners [8,9]. This file format is specified by the ASTM, an international standards organization, and it is documented in the ASTM E2807 standard. The E57 format is intended to be a more general format that is well-suited for storing data across a variety of application domains.

At a high level, the structure of an E57 file is a hierarchical tree structure, as shown in Fig. 5. The format of the hierarchy is based on the XML data format. At a low level, the actual point data is represented using a compressed binary format. Other large data blocks, such as images, are also represented efficiently in binary. In this way, the format supports flexibility and extensibility using text-based XML, while enabling efficient input/output and storage using compressed streams of binary data.

An E57 file is divided into three parts: a header, a set of optional binary sections, and an XML section. The header is a small, 48-byte binary structure that contains critical file-level information, such as the version number and the location of

Fig. 5 E57 file format

Header
Binary Section (points)
Binary Section (points)
…
Binary Section (points)
XML Section

the XML section. The XML section contains the hierarchical tree structure describe above. If the file contains point data or images, these portions of the hierarchy are referenced by the XML section, and the actual data is stored in the binary sections, with a separate section for each set of points or image.

2.3 libE57: an existing implementation

The **libE57** system consists of a library, supporting utilities and example programs, and documentation. The software includes two separate application programming interfaces (APIs) for reading, writing, and manipulating E57 files—the foundation API and the Simple API [9].

The foundation API is a full-featured interface that operates at a relatively low-level, allowing control over all aspects of an E57 file, including custom extensions.

The simple API is a simplified interface (built on top of the foundation API) that supports the most common use cases for reading and writing E57 files. The library also includes tools for converting LAS format files into E57 files, and a tool for validating E57 file correctness is under development.

3 Geometric primitive extraction

From the point data in either of LAS and E57 file formats, we can reconstruct the LiDAR scanned point cloud, as shown in Fig. 6. In this section, we show two implementation schemes to extract geometric primitives from the point clouds. The existing geometric library of CGAL [12] are used to realize our schemes.

3.1 DEM extraction

In this section, we present our scheme to extract DEM (digital elevation model) data from airborne LiDAR data. *Digital elevation model* (DEM) represents bare ground surfaces without any objects like plants and buildings. To extract DEM data from the LiDAR point clouds, not only reconstruction of the point clouds but also object removals are needed [10]. However, this method takes a large amount of time to find neighboring points for each points. Thus, it is not appropriate for large-scale airborne LiDAR point clouds which typically consists of millions of points.

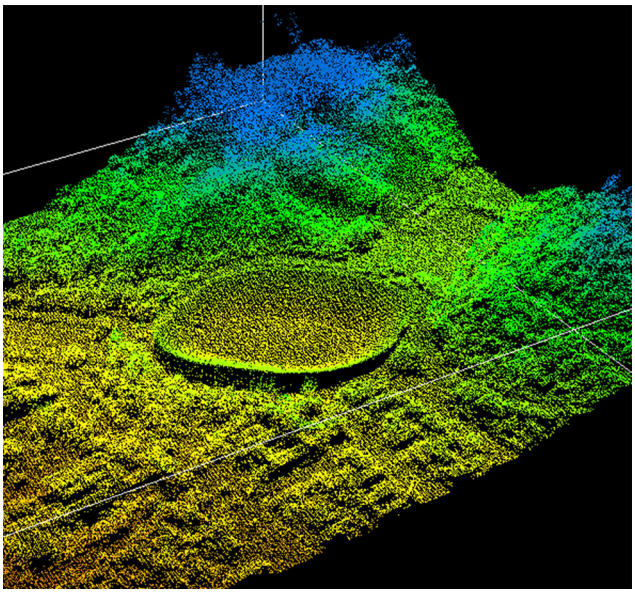


Fig. 6 An example LiDAR scanned point cloud (courtesy of UNAVCO)

To generate DEM's, we used height maps, which can be extracted from the height information of point clouds. Height maps are actually the projection of point clouds onto 2D planes. Initially, a height map is constructed as a two-dimensional array with zero values. For every points in the target point cloud, we project those points onto the 2D plane. During the projection, we can store the original height of those points as floating point numbers. In our system, we digitized the Z values to integer values between 0 and 255, for more simplicity and efficiency. Since we will use these height maps mainly for visualization purposes, we can accept these rough approximation.

If needed, we can use floating-point values for more accuracy. Since the Z values in our height map is restricted to integers between 0 and 255, the height map can be interpreted as an intensity map, as shown in Fig. 7.

We extracts the contours through connecting projected points on height map with the same height value as shown in Fig. 8. However, using these contours directly for the DEM's has some problems, since generated height map contains lots of small contours, which represents objects like trees and buildings, while DEM should represent the bare ground without any objects.

Our system can update generated height map to remove objects, instead of manipulating point cloud itself. Since height maps can be represented as 2D intensity maps, it is much concise to read, and it can be manipulated as bitmap images.

Height maps can not only be manipulated itself, but also can be applied image based filters. For example, we can apply median filter to remove noises captured during LiDAR data

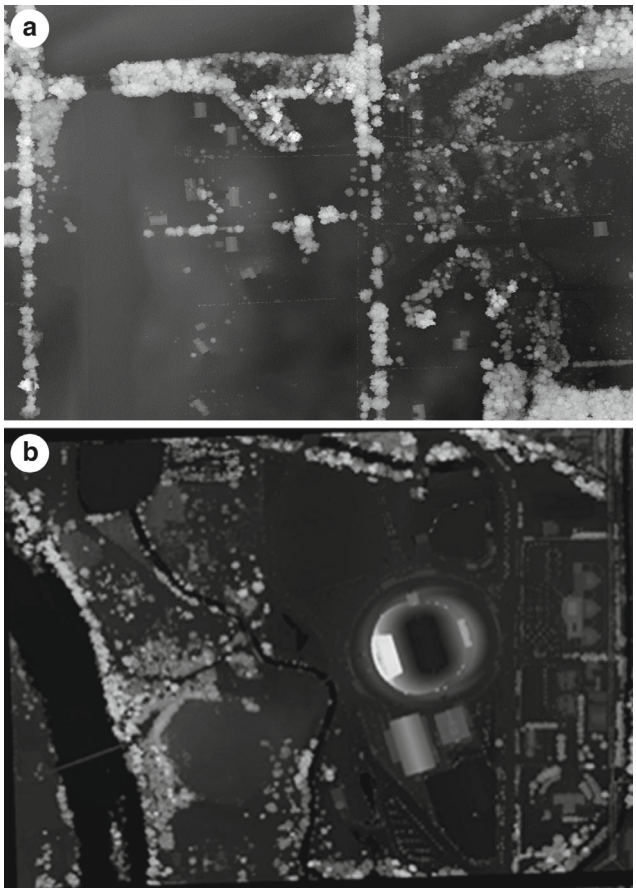


Fig. 7 Height maps presented as intensity maps. (**a**) An example height map of airborne LiDAR data in rural area. (**b**) An example height map of airborne LiDAR data

Fig. 8 Contour extraction from the height map of Fig. 7b.

acquisition, Gaussian filter to remove small object like trees and many another image based filters. Figure 9a shows the median filter applied version of height map. Figure 9b represents the generated contours using filtered height maps. Newly generated version of contour shows much smaller

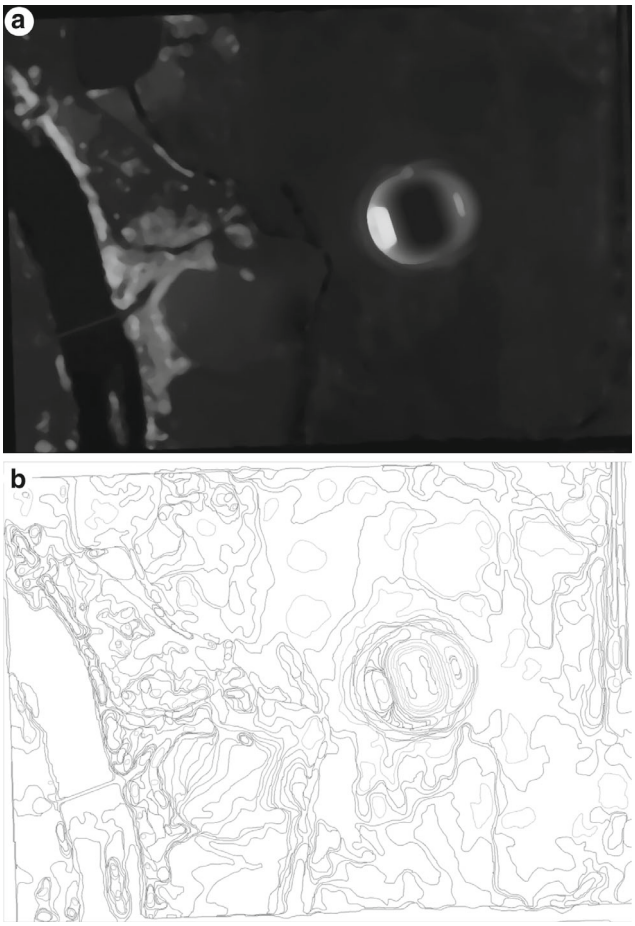


Fig. 9 Height maps and contours after filtering operations. **a** The median filtered height map. **b** Generated contours from the filtered height map

number of partial contours, which means small objects are automatically removed and in a proper way.

Additionally, for LiDAR data that have color information, we also construct texture maps from color information of point clouds. Texture maps are represented as two-dimensional array. To construct a texture map, we initially clear it with zero color values. Then, for every points in point clouds, we project that point to the 2D-plane with the size of the given texture map. The color values of those points are stored at the corresponding locations. We used this texture map as an aerial photograph to render precise terrain. An example of constructed texture map is shown in Fig. 10.

3.2 3D Model reconstruction

In this section, we present our implementation scheme to construct the 3D geometric models from cloud points acquired with terrestrial LiDAR equipment. Our final goal is to build 3D model of real-world constructions, from their scanned

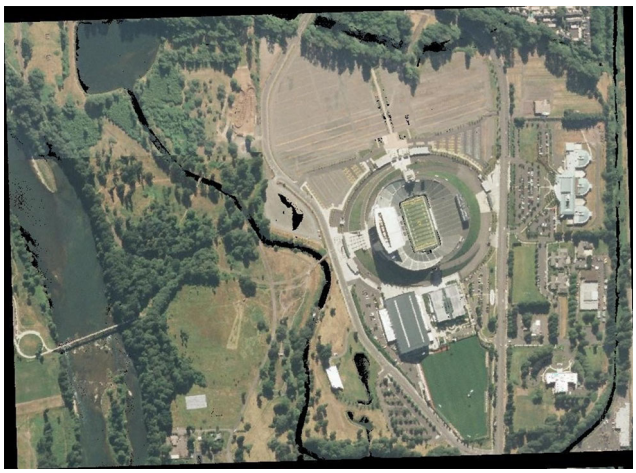


Fig. 10 A texture map extracted from the LiDAR data

LiDAR data. Those geometric models will be used on other application systems including 3DS Max, 3D GIS, and others.

Although there are lots of researches on 3D model reconstruction from point clouds [11], it is hard to adopt those methods to reconstruct surfaces for large-scale point clouds, which typically correspond to buildings and real-world constructions. According to [11], the biggest number of cloud points used for surface reconstruction is approximately 6 million, and reconstruction from these 6 million points took at least 30 seconds to get reasonable results. However, our captured data of building point clouds consist of more than 248 million points, as shown in fig 11a. Considering almost reconstruction algorithms show the time complexity of O(n^2), it would take tremendous time to reconstruct those point clouds.

Another problem we can't adopt precedent methods is they assume that point clouds would be in a single closed surface. Though buildings should be in closed surface like cube, LiDAR scanner can't scan bottom side of the building, and finally we have big holes at the bottom. Figure 11b shows a typical sample point cloud of a specific building. Figure fig11c shows the hole at the bottom of that building.

So, our scheme is to manually pick geometric primitives like triangles, squares, cylinders and/or cubes out of point clouds to generate 3D Models. After picking basic attributes, we grabbed framebuffer to generate texture of generated basic primitives. Figure 12 shows a sequence of semi-automatically extracting geometric primitives from a large-scale point cloud.

3.3 Geometric primitive extraction with CGAL

In this section, we introduce CGAL, the Computational Geometry Algorithms Library. CGAL is a software library of computational geometry algorithms [12]. CGAL is a software project that provides easy access to efficient and reliable geometric algorithms in the form of a C++ library. CGAL is used in various areas needing geometric computation, such as geographic information systems, computer aided design, molecular biology, medical imaging, computer graphics, and robotics.

a

Building Name	Number of points	Raw File size
Bangwha	318,046,967	15.9 GB
Seobook	248,031,609	12.4 GB
Hwaseo-moon	385,000,039	19.3 GB
Hwahong-moon	430,944,637	21.6 GB

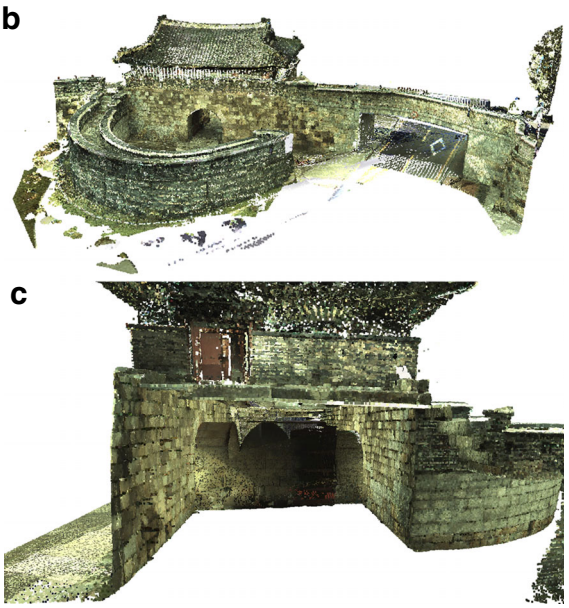


Fig. 11 Statistics and samples of our terrestrial LiDAR data. **a** LiDAR scanned data of buildings in Suwon, Korea. **b** A sample point cloud of a building. **c** The point cloud have a big hole in the bottom side

The CGAL library offers data structures and algorithms like triangulations, convex hull algorithms, point set processing, arrangements of curves, surface and volume mesh generation, geometry processing, alpha shapes, shape analysis, axis-aligned bounding boxes (AABB) and k-D trees.

In this famous geometry library, we focused on the following three components:

- **Point set processing component:** This component implements methods to analyze and process unorganized point sets. The input is an unorganized point set (un-oriented or oriented). The point set can be analyzed to measure its average spacing, and processed through functions devoted to the simplification, outlier removal, smoothing, normal estimation, normal orientation and feature edges estimation.
- **Point set shape detection component:** This component implements an efficient RANSAC-based primitive shape detection algorithm for un-oriented point sets.

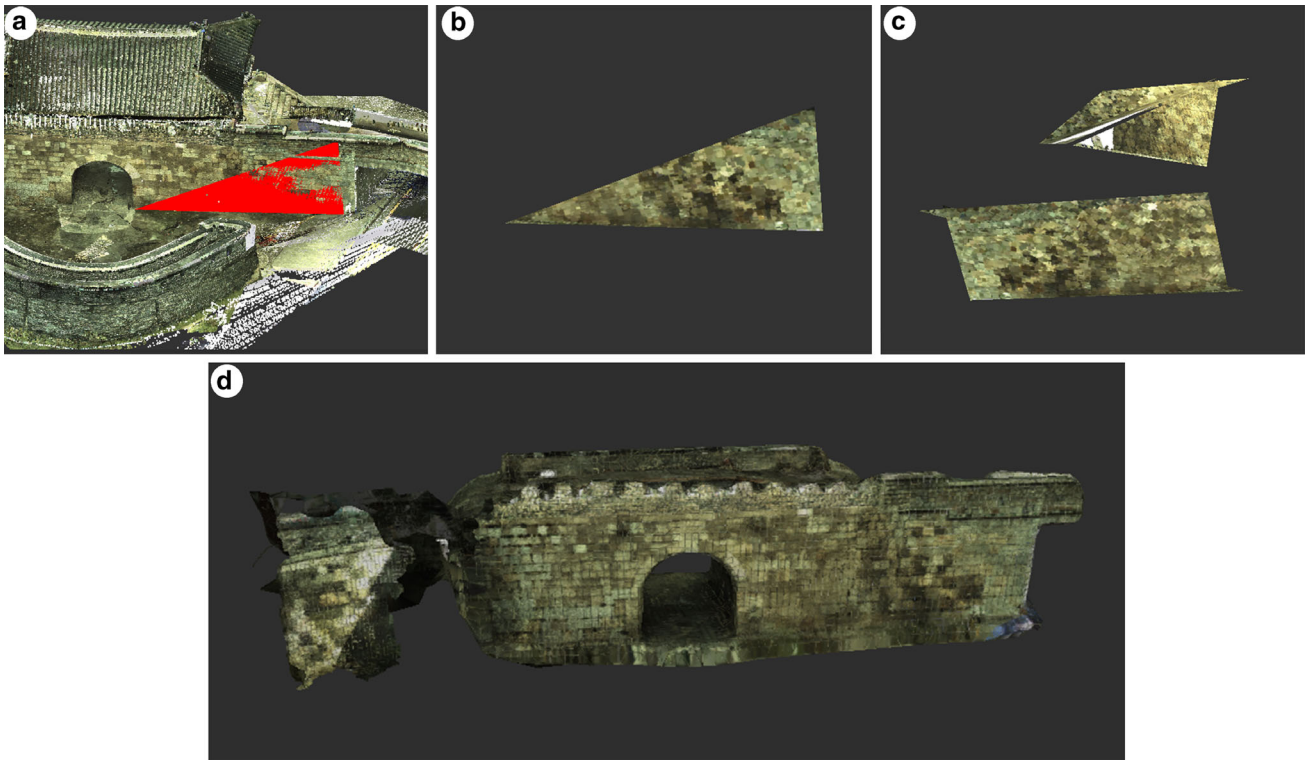


Fig. 12 Sequence of extracting geometric primitives from point clouds to construct the 3D geometric model. **a** selecting a specific geometric primitives from the point cloud. **b** a simply extracted geometric primitive from **a**. **c** Generating a set of geometric primitives from the point clouds. **d** The final result of geometric primitives

RANSAC (random sample consensus) is an iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers. It is a non-deterministic algorithm in the sense that it produces a reasonable result only with a certain probability, with this probability increasing as more iterations are allowed [13,14].

- **Estimation of local differential properties of point-sampled surfaces component:** For a surface discretized as a point cloud or a mesh, it is desirable to estimate pointwise differential quantities. This component allows the estimation of local differential quantities of a surface from a point sample.

We tried to use it for our geometric primitive extraction from cloud points that have closed surface. Figure 13 shows examples of surface reconstruction of point cloud with closed surface. Figure 13a shows a raw point cloud, and Fig. 13b–d is its geometric mesh representation, with respect to the difference resolution settings.

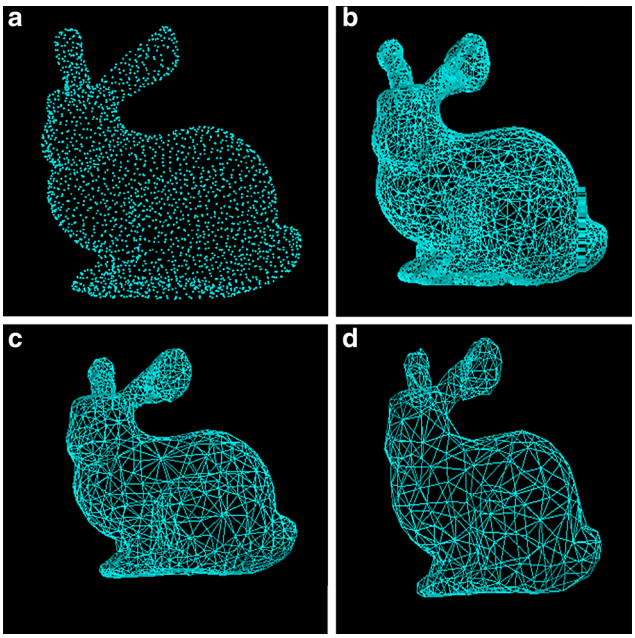


Fig. 13 Examples of surface reconstruction with closed surface. **a** Original point cloud, 2,503 points. **b** Mesh representation with 2648 points. **c** Mesh representation with 932 points. **d** Mesh representation with 432 points

4 Implementation and its results

Originally, our fundamental strategy for the geometric data extraction was to develop in-house versions of the well-

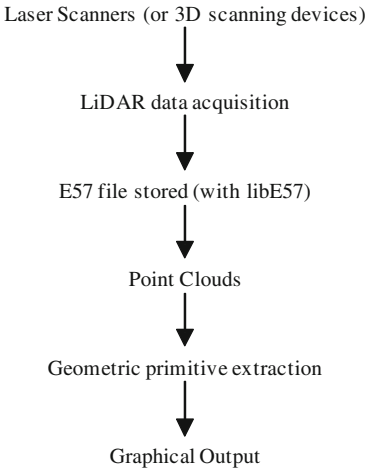


Fig. 14 Our over-all system design for LiDAR data handling

Fig. 15 Integrating the geometric extraction with an existing GIS application program

known registration algorithms. However, we rapidly recognized that preceding well-known algorithms are inefficient for both airborne and terrestrial LiDAR data.

We developed new schemes to extract geometric primitives from LiDAR data and adopted them to our prototype implementation. Additionally, we also tried to use CGAL which is one of widely-known well-known implementation, in our implementation. The basic steps in our prototype implementation are presented in Fig. 14. From the implementation of our prototype system, we acquired results in Figs. 9, 12, and 13 with reasonable performances.

The acquired geometric primitives are combined to construct the geometric representation of real-world constructions, from their corresponding point clouds. The geometric primitives can be used in various application programs. Figure 15 shows an example integration of our extracted geometric model with existing GIS navigation platform.

5 Conclusions and future work

Nowadays, we have many requirements for LiDAR data handling. We started to find an efficient way of handling LiDAR data and its related information. Actually, we focused on two technical issues:

- Storing the LiDAR data itself
- Extracting geometric primitives from those point clouds.

Our analysis shows that the E57 file formats and some registration algorithms are good solutions to these issues. Additionally, we found that some algorithms are already available in public domain, especially in the geometry handling libraries including CGAL.

We developed two new implementation schemes to extract geometric primitives. With these new schemes, we achieved the final prototype of LiDAR data handling framework. Through combining these methods, we got the geometric models remarkably efficiently, even in a cost-effective way.

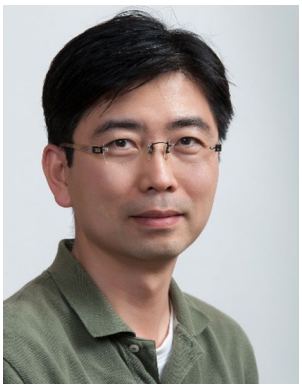
Our prototype system still need some more technical improvements in its practical implementations. In near future, we will accelerate our implementation with GPU-based parallel executions with CUDA and/or OpenCL.

Acknowledgements This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (Grant 2016R1D1A3B03935488). This study was also supported by the 2015 Technology Innovation Development program (project name: "Development of 3D GIS Platform for the LiDAR Data Utilization") funded by the Small and Medium Business Administration, Korea (project number: S2306348). The authors thank to Mr. Byeonguk Im and Mr. Joongin Lee for their contributions to the early-stage experiments on the prototype implementations of our schemes.

References

1. Heritage, G., & Large, A. Laser Scanning for the Environmental Sciences. Wiley, New York. ISBN 1-4051-5717-8. (2009)
2. Maltamo, M., Næsset, E., & Vauhkonen, J. Forestry Applications of Airborne Laser Scanning: Concepts and Case Studies (Vol. 27). Springer, Dordrecht. ISBN 9-4017-8662-3. (2014)
3. Shan, J., & Toth, C. K. Topographic Laser Ranging and Scanning: Principles and Processing. CRC press, Boca Raton. ISBN 1-4200-5142-3. (2008)

4. Vosselman, G., & Maas, H. G. Airborne and Terrestrial Laser Scanning. Whittles Publishing, Boca Raton. ISBN 1-4398-2798-2. (2010)
5. A. Samberg, "An implementation of the ASPRS LAS standard," IAPRS, vol. XXXVI, (2007)
6. The American Society for Photogrammetry & Remote Sensing, LAS Specification Version 1.4, ASPRS, (2011)
7. The LAS ASPRS LiDAR data translation toolset, http://www.liblas.org/. (2016)
8. Daniel Huber, "The ASTM E57 file format for 3D imaging data exchange", Proceedings of the SPIE Vol. 7864A, Electronics Imaging Science and Technology Conference (IS&T), 3D Imaging Metrology, January, (2011)
9. libE57: Software Tools for Managing E57 files. http://www.libe57.org/. (2016)
10. Yuan Feng, et. al. Urban DEM generation from airborne Lidar data. Urban Remote Sensing Event ISSN 2334-0932. (2009)
11. Matthew, Berger., et al. A Benchmark for Surface Reconstruction. ACM Transactions on Graphics, Vol.32 (2013). April 2013
12. CGAL homepage, http://www.cgal.org/. (2016)
13. Strutz, T.: Data Fitting and Uncertainty, 2nd edn. Springer Vieweg, Berlin (2016)
14. Hast, Anders, Nysjö, Johan, Marchetti, Andrea: Optimal RANSAC—towards a repeatable algorithm for finding the optimal set. J. WSCG **21**(1), 21–30 (2013)



Nakhoon Baek is currently a professor in the School of Computer Science and Engineering at Kyungpook National University, Korea. He received his B.A., M.S., and Ph.D. degrees in Computer Science from Korea Advanced Institute of Science and Technology (KAIST) in 1990, 1992, and 1997, respectively. His research interests include graphics standards, graphics algorithms and real-time rendering. He is now also the Chief Engineer of Dassomey.com Inc., Korea.

Woo-seok Shin is currently in M.S. course in the School of Computer Science and Engineering at Kyungpook National University, Korea. He received his BS in Computer Science and Engineering from Kyungpook National University (KNU) in 2016. His research interests include real-time rendering and large scale parallel processing.

Kuinam J. Kim received the B.S. degree from Mathematics, University of Kansas in 1989. He received the M.S. degree of Statistics and Ph.D. degree of Industrial Engineering from Colorado State University. He is currently Professor of Industrial Security Department, Kyonggi University, Korea. His research interests include industrial security.