ISPRS Journal of Photogrammetry and Remote Sensing xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

journal homepage: www.elsevier.com/locate/isprsjprs

A scalable and multi-purpose point cloud server (PCS) for easier and faster point cloud data management and processing

Rémi Cura [a,b,*], Julien Perret [a], Nicolas Paparoditis [a]

[a] Universite Paris-Est, IGN, SRIG, COGIT & MATIS, 73 avenue de Paris, 94160 Saint Mande, France
[b] Thales Training & Simulation SAS, 1 rue du Général de Gaulle 95523 Cergy-Pontoise, France

ARTICLE INFO

Article history:
Received 15 January 2016
Received in revised form 13 June 2016
Accepted 13 June 2016
Available online xxxx

Keywords:
RDBMS
Point cloud management
Point cloud generalisation
Patch
Meta-data
Point cloud server

ABSTRACT

In addition to more traditional geographical data such as images (rasters) and vectors, point cloud data are becoming increasingly available. Such data are appreciated for their precision and true three-Dimensional (3D) nature. However, managing point clouds can be difficult due to scaling problems and specificities of this data type. Several methods exist but are usually fairly specialised and solve only one aspect of the management problem. In this work, we propose a comprehensive and efficient point cloud management system based on a database server that works on groups of points (patches) rather than individual points. This system is specifically designed to cover the basic needs of point cloud users: fast loading, compressed storage, powerful patch and point filtering, easy data access and exporting, and integrated processing. Moreover, the proposed system fully integrates metadata (like sensor position) and can conjointly use point clouds with other geospatial data, such as images, vectors, topology and other point clouds. Point cloud (parallel) processing can be done in-base with fast prototyping capabilities. Lastly, the system is built on open source technologies; therefore it can be easily extended and customised.

We test the proposed system with several *billion* points obtained from Lidar (aerial and terrestrial) and stereo-vision. We demonstrate loading speeds in the *~50 million pts/h per process* range, transparent-for-user and greater than *2 to 4:1* compression ratio, patch filtering in the *0.1 to 1* s range, and output in the *0.1 million pts/s per process* range, along with classical processing methods, such as object detection.

1. Introduction

The last decades have seen the rise of Geographical Information System (GIS) data availability, in particular through the open data movement. Along with traditional image (raster) and vector data, point clouds have recently gained increased availability (the site opentopography[1] is a good example) and usages (robotic, 3D, virtual reality). Sensors are increasingly cheap, precise and available. However, due to their massive un-organised nature (no neighbourhood information) and limited integration with other GIS data, the management of point clouds still remains challenging. This makes point cloud data barely accessible to non-expert users. Yet, many fields would benefit from point clouds, had they an easiest way to use them.

1.1. Problems

Point clouds data sets are commonly in the TeraByte (TByte) range and have very different usages; therefore, every aspect of their management is complex and has to scale.

Having such large data sets makes the compression an essential need. Not only is the compression necessary, but it also has to maintain a fast read and write access, and be transparent for the users. Indeed, we observe that, today, virtually all images and videos are compressed; most users not noticing it at all.

Similarly, so much data cannot (should not) be duplicated and must be shared, following a broader trend in the Information Technology world. Sharing data necessarily introduces concurrency issues (several users simultaneously reading/writing the same data).

Users usually need to access only a part of the data at once, thus efficiently extracting (filtering) a subset is important. With many varying usages, the criterae for choosing the subset may be volatile, and sometimes mixed.

* Corresponding author at: Universite Paris-Est, IGN, SRIG, COGIT & MATIS, 73 avenue de Paris, 94160 Saint Mande, France.
E-mail address: remi.cura@ign.fr (R. Cura).
[1] www.opentopography.org.

http://dx.doi.org/10.1016/j.isprsjprs.2016.06.012
0924-2716/

Visualising something helps understanding it. In the case of multi-billion point clouds, a Level Of Detail (LOD) strategy is necessary, because the data set cannot be displayed in its entirety at once.

Features of point clouds can be very different depending on their source (Lidar, stereovision, medical, etc.), regarding the number and type of attribute, the geometric precision and noise, etc. Yet, point clouds usually are geospatial data, which makes them akin to vectors and rasters from the GIS world. Thus, point clouds may be used conjointly to other data types, either directly or by converting point clouds to images or vectors.

Lastly, point clouds are processed in many different ways suiting each user's needs. These methods must be fast and easy to design, scale well, and be robust.

Another important problem is related to point cloud management. For various reasons, point clouds are often handled as sets of points. Yet, a point cloud (data set) is much more than just points, as it also includes meta-data and other information such as sensor geometry, etc. Managing such data sets is difficult; like knowing which data sets are available and where. Because data sets are heterogeneous, managing extended meta-data such as point cloud coverage, date of acquisition and so on is also difficult, especially without a standard data format. Treating point clouds as only points is especially problematic, as is illustrated by a very recent benchmark release,[2] which provides massive and very useful hand-labelled point clouds, yet does not provide any meta-data at all, neither extended meta-data nor contextual data.

In this article, we propose to use a point cloud server (PCS) to solve some of these problems. The proposed server architecture provides perspectives for meta-data, scalability, concurrency, standard interfaces, co-use with other GIS data, and fast design of processing methods. We create an abstraction layer over points by dealing with groups of points rather than individual points. This results in easy compression, filtering, LOD, coverage, and efficient processing and conversion.

1.2. Related work

1.2.1. File system

Historically, point clouds have been stored in files. In order to manage large volumes of these files, a common solution is to build a hierarchy of files (a tree structure, like a quad tree for instance) and access the data through a dedicated set of softwares. This approach is continuously improved (Hug et al., 2004; Otepka et al., 2012; Richter and Döllner, 2014) and a detailed survey of the features of such systems can be found in Otepka et al. (2013). This approach is simple, and scaling is relatively easy (provided the Operating System (OS) maximum file number is not reached). However, using a file-based system has severe limitations. These systems are usually built around one file format, and are not necessarily compatible with other formats. Recent efforts have been made towards format conversion.[3] The features of such systems are very limited (limited meta-data handling, lack of integration with other GIS data, difficulty to use several point clouds together). Moreover, these systems are not adapted to sharing data and multi-users environments (concurrency).

1.2.2. DBMS for points

Hofle (2007) proposed to use a Data Base Management System (DBMS) to cope with the concurrency issue. The DBMS creates a layer of abstraction over the file-system, with a dedicated data retrieval language (SQL), native concurrency capabilities (supporting several users reading/writing data at the same time), and the wrapping of user interactions into transactions that can be cancelled in case of errors. DBMSs have also been used with raster and vector data for a long time, and the possibility to define relations in the RDBMSs (Relational DBMS) offers a simple way to create robust data models, and deal with meta-data. Adding the capability to create point clouds as services, DBMSs solve almost all the problems we face when dealing with point clouds. Usually, the database stores a great number of tables, and each table storing a point per row (Lewis et al., 2012; Rieg et al., 2014). Such a database can easily reach billions of rows. Nevertheless, storing these many rows is problematic because DBMSs may have a non-negligible overhead per row. Moreover, indexing such a number of row is slow and takes a lot of space, and the possibilities for compression are limited.

1.2.3. Column store database and No-SQL

These limitations are more generic than point clouds, and apply to any massive amount of data which is weakly relational and does not get updated often. As such, they have been researched and inspired the concept of column-oriented databases, such as MonetDB.[4] This database system is used to store individual points (Martinez-Rubi et al., 2014; Martinez-Rubi et al., 2015; van Oosterom et al., 2015). This approach is effective to store large amounts of row without much overhead, and also solves most of the indexing issues. However, points are not compressed, integration with other GIS data is weak, and scaling to multiple computers is not straightforward. In parallel, stripped down column stores were proposed, having been specially tailored for massive and weakly relational data, forming the No-SQL databases. They scale extremely well to many computers and can deal with large amounts of data (Wang et al., 2014 and SpatialHadoop[5]). However, this comes at a price. Indeed, NoSQL databases must drop a few guarantees on data. At the moment, they are not integrated with other GIS data and have much less functionalities. Indeed, NoSQL databases are closer to being a file-system distributed on many computers (with efficient indexing) than being DBMSs. Thus, massive scaling still necessitates specialised hardware, and the people to maintain it.

1.2.4. Cloud computing

A recent possible workaround for this issue is to use Cloud Computing facilities[6] to store the points, like Amazon S3. In this solution, data storage is offered as a service and externalised. This may provide the ultimate scaling, but suffers from the same limitations as the NoSQL, with open issues on indexing.

1.2.5. DBMS for patch

All the previous data management systems try to solve a very difficult problem: managing a massive quantity of individual points. Solutions that scale well must focus on data storage and retrieval, and drop the rest of the management problem (feature, meta-data, integration, processing). Another recent approach is being explored in pgPointCloud (2014) and other commercial RDBMS. The key idea is to manage groups of points (called patches), rather than points, in a RDBMS. Creating this abstraction layer over points allows retention of all the advantages of a RDBMS, but keeps the number of rows low, thus avoiding the associated scaling difficulties (index, compression). Moreover, the proposed abstraction offers new theoretical possibilities, because it creates a generalisation of the groups of points. The price is that, in order to access a point, its whole group has to be accessed first, and so

[2] www.semantic3d.net.
[3] http://www.pdal.io/.
[4] http://www.monetdb.org.
[5] http://spatialhadoop.cs.umn.edu/.
[6] https://github.com/hobu/greyhound.

Please cite this article in press as: Cura, R., et al. A scalable and multi-purpose point cloud server (PCS) for easier and faster point cloud data management and processing. ISPRS J. Photogram. Remote Sensing (2016), http://dx.doi.org/10.1016/j.isprsjprs.2016.06.012