Towards 10^{15}-level point clouds management - a nD PointCloud structure

Haicheng Liu
TU Delft
Delft, Netherlands
H.Liu-6@tudelft.nl

Peter van Oosterom
TU Delft
Delft, Netherlands
P.J.M.vanOosterom@tudelft.nl

Martijn Meijers
TU Delft
Delft, Netherlands
b.m.meijers@tudelft.nl

Edward Verbree
TU Delft
Delft, Netherlands
E.Verbree@tudelft.nl

Abstract

Drastically increasing production of point clouds as well as modern application fields like robotics and virtual reality raises essential demand for smart and highly efficient data management. Effective tools for the managing and direct use of large point clouds are missing. Current state-of-the-art database management systems (DBMS) present critical problems such as inefficient loading/indexing, lack of support of continuous Level of Detail (cLoD) and limited functionalities. Previous research has suggested and demonstrated the importance of converting property dimensions such as time and classification to organizing dimensions for efficient data management at the storage level. However, a thorough validation and theory are still missing. Besides, how new computational platforms such as the cloud technology may support data management also needs further exploration. These problems motivate the PhD research with the focus on a new data structure (nD PointCloud) which is dedicated for smartly and flexibly organizing information of large point clouds for different use cases.
Keywords: point cloud, data management, data structure, database, dimension

1 Introduction

Point clouds form big data nowadays. A typical point cloud can contain billions, even trillions (10^{12}) of points. For instance, AHN is a detailed elevation model of the whole Netherlands measured with Airborne Laser Scanner (ALS). Fugro among other firms had been working on the second version, i.e. AHN2 from 2008 to 2013 and acquired 640 billion points in the end. It is also pre-calculated that the Dutch national Mobile Lidar Scanning (MLS) dataset (street view) collected by CycloMedia could achieve an amount of 35 trillion points eventually. Moreover, in January 2018, the National Institute of Standards and Technology (NIST) of US initialized a public safety research program in which the topic was the collection of indoor point clouds. The intension was to build a standard prototype for indoor point cloud models through the initiative, as point clouds may become the basis for indoor applications for the next generation.

However, current state-of-the-art solutions like Oracle, PostgreSQL, PDAL, Lastools and HDF present problems including cumbersome and inefficient loading, indexing and querying processes, much development work for subsequent applications, etc. If more attributes like intensity, classification and color information were elaborated, the burden of efficient point cloud management would become even heavier. The innovations of existing data storage approaches and architectures are no doubt imperative.

This paper starts with specific problems concerned with massive point clouds management from practical experience and literature study. Then novel methods are proposed based on a nD PointCloud data structure which will be verified by use cases. In the end, initial results including primary benchmarking and software testing are presented.

2 Problems

Motivation also comes from technical issues unsolved from several aspects.

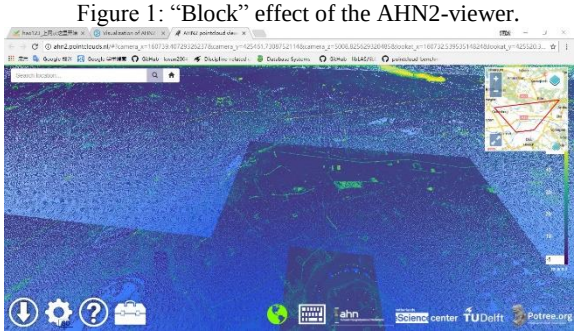
2.1 Visualization

Both industry and academia have already experienced bottlenecks of visualizing large point clouds. Specifically, Fugro BV has established an inspiration center to introduce Virtual Reality (VR) services to users and the scene inside is built directly on point clouds. However, due to limited memory capacity, the VR equipment can only store small amount of point, which confines the size of the scene. And if more points were inserted, the headset would crash.

With respect to viewers, a major problem of the AHN2-viewer (Van Oosterom et al., 2016) is that point clouds of discrete densities shown in the same scene (Figure 1). This is caused by the octree data structure used to organize the data storage. Octree is basically a *discrete* LoD (dLoD) structure in a data pyramid organization. So during the rendering, due to different distance, blocks with different densities, i.e. layers are shown with sharp boundaries. This effect should be avoided for a smooth visualization, which is even more critical for the VR environment which poses high requirements of vivid simulation, i.e. real-time, extremely high accuracy and semantic enrichment.

Besides, data rendering through the web service is intermittent with blocks popping up discontinuously, but the CPU and I/O of the server remains an easy mode. So the performance issue most likely lies in the data streaming and transmission processes. Consequently, either current protocols should be structured and combined in a more efficient way or new protocols for massive point data retrieving in a distributed environment should be developed.

Figure 1: "Block" effect of the AHN2-viewer.



2.2 File system and DBMS

Most current software for point clouds are based on files. Some are directly constructed on LAS/LAZ, HDF files, while other vendors adopt their own formats. Additional sorting and indexing to create block data structures for efficient querying have to be manually performed. With available APIs, developers may program using script languages to resolve specific tasks in a short time. However, when a new function arises, then redevelopment or additional development should be performed. Yet the scalability with large data cannot be guaranteed. Besides, when other data types such as vector and raster are involved, multiple formats, libraries and systems can hardly constitute a robust software.

DBMSs also experience bottlenecks. Oracle for example releases state-of-the-art solution for point clouds management. Two approaches are available, and one is based on SDO PC blocks while the other utilizes Index-Organized Tables (IOT). However, a key issue for the implementation is the complex process for data preparation. Basically for both approaches, the data has to be first loaded into a heap table before it can be converted into either IOT or blocks. Inefficient data loading and indexing are also the major reasons that prevent industrial engineers from adopting a DBMS solution. Besides, regarding storage, table approaches are not convenient for compression, while block approach is mainly implemented for efficient storage but fairly time consuming to construct due to sorting. Yet an updating operation entails the recreation of blocks. Also its logic is not as intuitive as tables, maximum selection for example.

Point Data Abstraction Library (PDAL) is a C/C++ open source library and applications for translating and processing point cloud data. Through PDAL, pre-sorted point clouds can be directly loaded into Oracle as blocks without the heap table process. But it also utilizes its own APIs to manipulate blocks in the databases and the query is encapsulated in a xml file together with native SQL statement, which is more complex.

2.3 Benchmark

To assess performance of different tools as well as advancement brought by new platforms, it is essential to define and perform a comprehensive benchmark test. Aspects including query types, data size, hardware parameters and setting of data structures (e.g. block size, compression) should be specifically addressed by the benchmark. Different sets of tests should be established and repeated several times to determine the influence of each of the factors, which is a very labor-intensive work (Van Oosterom et al., 2015; Liu et al., 2016). Additionally, in the benchmark tests, execution of queries is repeated but users have different query habits in reality. For example, a rectangle spatial selection may be then followed by a time series extraction. As a matter of fact, cache usage of discontinuously query execution varies from the repeated case (Liu et al., 2016). As a result, the best solution tested may actually not work efficiently as expected. Van Oosterom et al. (2016) performed initial experiments for a realistic benchmark by simulating simultaneously queries from multiple users. But the benchmark was mainly confined by xyz without more attributes. In addition, more complex query processes are missing in current benchmarks (Van Oosterom et al., 2015; Psomadaki, 2016) apart from spatio-temporal selections. Spatio-temporal join operation for example is a fundamental spatial computation and frequently occurs in spatial analysis based on vectors and rasters. Point clouds are also involved for this operation in many cases, the k-Nearest-Neighbour (kNN) search involved in the change detection for instance. Perspective view query for visualization is another essential type during the rendering process of 3D terrain models.

Besides benchmark data and query sets, the architecture for benchmark testing is a sophisticated issue. The massive point cloud management environment could be a distributed sever-client-protocol system where a query execution can pass through a multi-level hierarchy in a single machine and also communications between servers constitute a critical part. Additionally in a cloud environment, other threads run in parallel may occupy essential computational power, which hampers a convincing testing result.

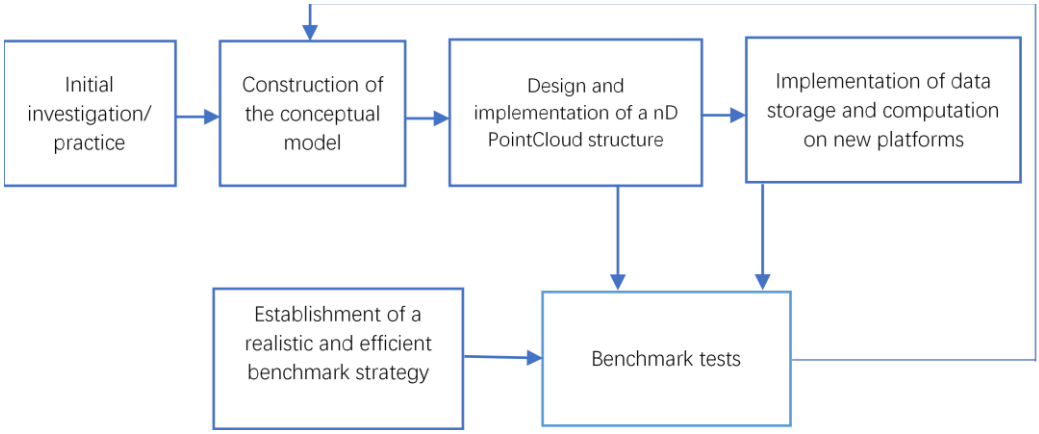
3 Methodology

The research aims to resolve issues mentioned above and it is further divided into 6 components (Figure 2).

3.1 Construction of the conceptual model

The conceptual model should support the visualization and the storage of large classified point clouds. Poux et al. (2017) proposed a SmartPC model that strived for point cloud management and processing. It is a 3-layer model consisting of a bottom data description layer, an upper domain adaption layer and a middle connection layer. The conceptual model in this research will be built on top of the SmartPC model. The data description will be connected with the nD PointCloud structure realized by adopting key organizing dimensions (e.g. space, time, scale, classification). The domain adaption layer will be interpreted with 3 representative use cases.

Figure 2: Hierarchy of sub-projects of the research.



3.2 Design and implementation of nD PointCloud

Novel data organization is the key to solve the problems of inefficient point cloud management and processing. In the research, a nD data structure will be designed and implemented. The concept of dimension should be distinguished at this stage where an organizing dimension is used to organize the data storage while a property dimension is a non-functional attribute including color, velocity, classification, etc. From prior work (Psomadaki, 2016; Van Oosterom et al., 2015) and requirements, following design constraints are proposed:

1. Clarifying different nature of dimensions to structure point cloud data. For example:
   - Traditional spatial dimensions xyz are still the fundamental dimensions to organize data as point clouds are mostly used for spatial analysis.
   - Temporal dimension can be of the same importance as spatial dimensions in certain applications. It is a dynamic dimension, i.e. the range can be changed and unlimited, and has close relationship with data updating.
   - Scale is a manually introduced dimension used for realization of the LoD structure. In the cLoD implementation, it refers to importance. Scale influences computing accuracy and efficiency significantly, e.g. classification at different scales.
   - Classification dimension should be established for semantic analytical purposes. It has limited number of integers, a major difference from others.
2. Efficient implementation of spatial operators such as spatio-temporal join and perspective view queries.
3. Interaction with users: providing flexibility to update organization of data.
4. Interfaces to support visualization and parallelization.
5. Other user requirements, e.g. storage space (compression), scalability.

Psomadaki (2016) proposed the possibility to map property dimensions to organizing dimensions for a unified structuring plan depending on the context. Experiments were conducted that either z or time was utilized as an organizing dimension for data management. In this research, spatio-temporal dimensions, scale and classification dimensions will be integrated into a key encoded using the Space Filling Curve (SFC) (Sagan, 2012). In the geospatial domain, SFC is commonly used to organize data in the multidimensional space into 1 dimensional space (Van Oosterom, 1999) and two main types are the Morton curve and the Hilbert curve. When encoding several dimensions into a SFC key, an essential issue is to employ the appropriate unit (e.g. meter, second) which controls the resolution and range of the data representation. This has direct influence on the computing efficiency based on the SFC scheme.

To take advantage of the intuitive table model, as well as an index integrated data structure of Oracle IOT, in the second step, the SFC key together with all property dimensions will be directly stored as leaf nodes. This will be accompanied by the development of interfaces to encode and decode the SFC. Mathematical axioms on spatial operations with the SFC encoding will be developed. A critical example is how to transform ranges in original dimensions into the span of the SFC key.

The aim of the data structure is to elaborate full possible organizing dimensions that currently concerned in quires into the design process. PDAL solution provides such a list (https://www.pdal.io/dimensions.html) for example. However, a dimension transformation implies a tremendous reorganizing process (e.g. sorting and blocking) for the computer. And as current knowledge and theory for an efficient solution are still insufficient, the implementation of the nD PointCloud structure will be decomposed into a 3-step procedure with each of them tested by a use case. The complexity of these applications increases gradually as more dimensions are involved:

Case 1 (4D): Only spatial dimensions xyz and the scale dimension are defined, other information is treated as property dimensions. As this is the most common case, both ALS and MLS data will be considered. The data management of the AHN2-viewer and VR rendering will be the use cases. Besides, normal spatial selections and spatial join computations like change detection will also be incorporated. They constitute main applications for this research.

Case 2 (5D): Spatio-temporal dimensions xyzt and the scale dimension are defined, together with other affiliated properties. Feature extraction from GPS tracks will be applied (Van Winden et al., 2016). Data can be collected from mobile devices or vessels.

Case 3 (6D): Spatio-temporal dimensions, the scale and the class dimension are realized. In this use case, computation becomes more complex, and semantics is involved intensively. An option is the visibility detection (Figure 3) inside buildings, where rays can pass through windows and space, but would be blocked by walls. Such a process can be integrated into a comprehensive VR simulation.

Figure 3: Faculty of architecture in TU Delft. Through one window, the square and another window, people and furniture in the other part of the building can be seen.

3.3 Implementation on the cloud

During the data collecting process, point clouds are normally stored and managed in local data centers, which complexes the implementation of large-scale applications. To conquer the issues of massive data volumes as well as huge number of user queries, employment of the cluster-based High-Performance Computing (HPC) paradigm is imperative. Current cloud technology has generalized the complex HPC processing model and is accessible by large numbers of users. So the local implementation of use cases will be transplanted on a cloud system (Figure 4) to handle big data practically.

In the data layer, data stored are airborne laser scanning data, indoor point cloud models and GPS tracks. These datasets will be managed using the nD PointCloud where cLoD computation takes place. However, data storage with original block and table approaches will also be created for benchmarking later.

In the functional layer, the database management module is mainly used by administrators and developers to store, maintain and benchmark data stores. The preprocess is concerned with noise removal and metadata management. Both dimension managing and cLoD computation are used to construct the nD PointCloud structure where optimization of storage can be made. Besides, system/query logs from the monitoring module can also be utilized to better organize the data. The analysis module contains all functionalities enrolled in the use cases proposed in this research. Spatial-temporal computation refers to property extraction from GPS tracking data, while generalized spatio-temporal queries and analysis functions are also included. Complete AHN2 and AHN3 datasets are planned to be compared for the change detection. Image rendering (Richter and Döllner, 2014) is mainly reserved for protocol testing. As in visualization applications, view perspective constantly changes, so smart caching plan is a key component to improve efficiency and smoothness. Parameters received from the map renderer which records motion of the user in the process layer will be utilized for extracting scenes as well as buffering neighboring data.

Current protocols present limitations for such a relatively new data type. Visualization supported by W3DS can only be synthesized images, which implies information lost during data retrieving. 3D visualization based on raw point clouds does not exist. Also there is no specific compression and LoD definitions in the protocols available. Communities (Kodde, 2010) have proposed to establish a new OGC standard, the Web Point Cloud Service (WPCS) which is expected to take account of the special aspects of point clouds. In this research, key points and bottlenecks that need to be improved or resolved will be developed for standardized WPCS.

3.4 Benchmark design

To begin with, the factors affect the query (including complex computations) performance are defined (Table 1). Large amounts of tests have to be carried out to learn the impact of each factor and the performance on scalability. To decrease the workload, the benchmark will be decomposed into a primary version and a full version.

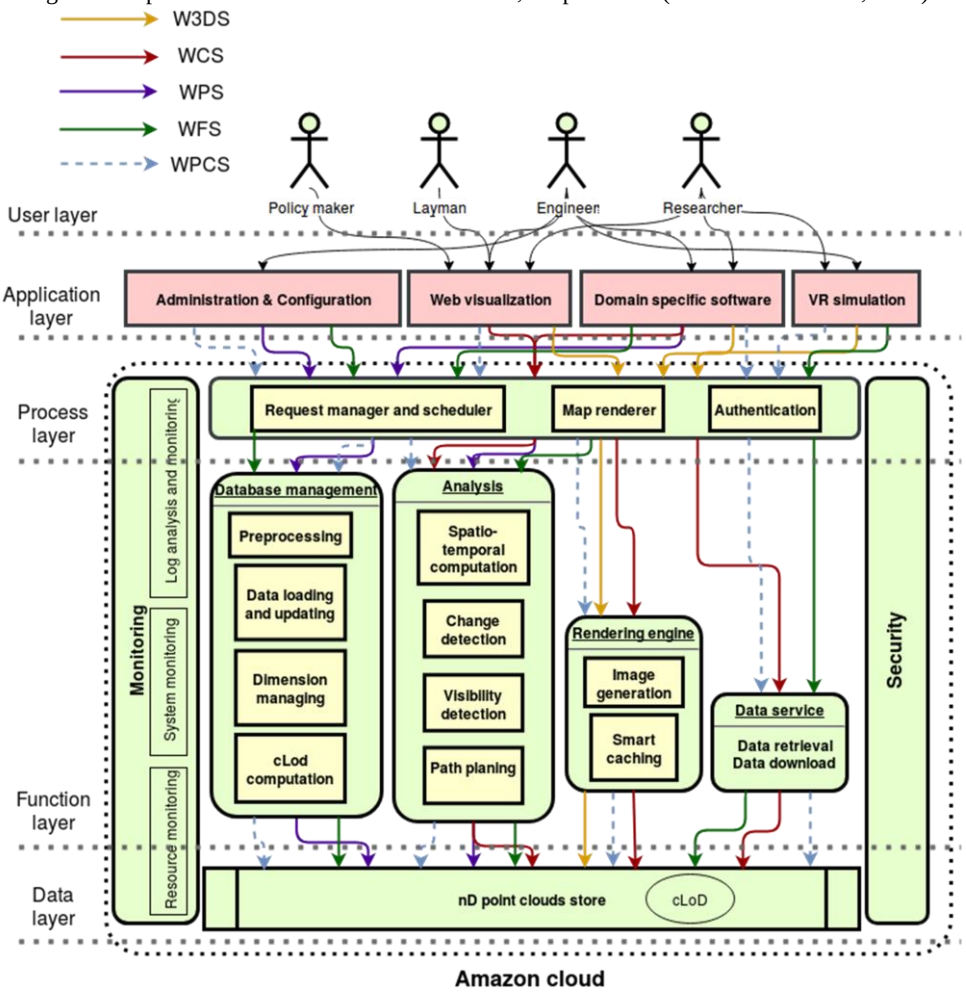
Table 1: List of factors influencing the query performance.

Category	Parameter
Data	Volume
Data structure	SFC type, dimensions, attributes, index, block size, compression
Query	Type (i.e. which dimensions or attributes are concerned), dimension span (e.g.1 x 100 or 10 x 10), output size
Software	Type, parallelization
Computer hardware	I/O, memory size, CPU
Environment	Number of users, network speed, protocol type

The primary benchmark is a local benchmark which is performed in the developing phase of the nD PointCloud structure, and also for the initial comparison among different solutions. Some of factors listed in Table 1 are sensitive, which means by altering the value slightly, the query performance will fluctuate dramatically, while others' impact may not be so significant. Through literature study and expert consultancy, relevant and controllable parameters will be selected for benchmark tests. The primary benchmark set will include spatial-temporal selections at different scales, certain aggregations and the kNN search. Every time only one factor changes while others remain their values. It is expected that after this step, sensitive factors will be identified.

In the full benchmark, complete point cloud datasets will be loaded and tested. Besides, the query types as mentioned before are more comprehensive. Only sensitive factors from primary benchmarking will be experimented and tuned to locate the optimal solution.

Figure 4: Implementation of use cases on the cloud, adapted from (Richter and Döllner, 2014).



3.5 Benchmark test

Newly designed nD PointCloud structure will be constantly tested with the benchmark to learn the actual performance, until the structuring scheme achieves an "optimal" solution. Then a formal comparison among different solutions will be conducted by implementing the primary benchmark. After it, cloud technology will be utilized to process much larger datasets for the sake of scalability. The data structure in the cloud will be investigated and tuned through the full benchmark. It will also be assessed by referencing normal solutions without the cloud. The overall evaluation of performance of various solutions will be based on the wall-clock time where interventions of different queries executed by multiple users will be elaborated.

4 Initial results

In the phase of initial investigation, some initial experiments have been conducted. Three sample datasets were utilized.

- AHN2delft.laz: 13,346,502 points with only x, y, z attributes, 46.8 MB.
- AHN3delft.laz: 508,564,458 points with x, y, z, intensity, return number, GPS time and classification, 2.3GB.
- S3DIS (Armeni et al., 2016): indoor point clouds containing 273,608,340 points with x, y, z, LoD (an computed importance value between 0 and 1), RGB, classification, room type, roomID, objectID.

4.1 Initial testing of Oracle

The initial tests were performed on a HP DL380p Gen8 server with 2 × 8-core Intel Xeon processors, E5-2690 at 2.9 GHz, 128 GB of main memory. The disk storage is a 41 TB SATA 7200 rpm in RAID5 configuration. S3DIS is used for testing.

Two Oracle approaches are tested:

- Oracle IOT, with x/y/z/classification encoded in a Morton (SFC) key, other attributes including LoD, r/g/b, roomtype, roominsid, objinsid as property

dimensions. Organizing index is based on the key and LoD.

- Oracle flat spatial table, with x/y/z stored as SDO_POINT geometric data type. A spatial index is built on the geometry column together with another index on the LoD column.

Table 2: Schema of testing queries

Query type	Schema
Selection on LoD	select * from *S3DIStab* where *lod* > 0.95
Selection on roomID	select * from *S3DIStab* where *roomID* = 6009
Statistical computation	select COUNT(distinct *roomID*) from *S3DIStab* where *roomType* = 205

The IOT table occupies 14 GB space on the disk while the flat spatial table takes up 42 GB among which 23 GB is used for storage of indexes. The total data loading time for IOT is around 12 minutes, whereas the creation of the flat spatial table totally costs 37 minutes including 34 minutes for index creation and related sorting processes. As regard to querying (Table 2), on average, the IOT approach is 5~10 times faster than the flat spatial table for LoD selection. As to property dimension selection, IOT can be 1.5 times faster. However, both approaches present almost the same speed for statistical computations.

4.2 Comparison of viewers

As a practice to learn basic functions provided by current point cloud viewers, a brief comparison of viewers is conducted (Table 3) on a laptop computer using the AHN2 and AHN3 samples. These desktop viewers (except AHN2-viewer, a web viewer) present different problems of data management, mainly lack of the LoD support which could be an effective method to solve the loading crash issue. Besides, smart caching strategy plays a crucial role in smooth rendering.

5 Summary

Given the big data problem as well as processing bottlenecks concerned with point clouds, a bottom-up research focusing on the data structure will be performed. First a modified

Table 3: Comparison of prevalent point cloud viewers.

Viewer	Attribute support	LoD support	Data loading	Memory usage
Bentley pointools view	Color, intensity	No	Conversion to PLT format should be performed before loading. AHN3delft loading fails as it cannot fit into memory.	Normal, it depends on data loaded. When there is operation, e.g. rotation, zooming, memory usage increases.
AMC bridge viewer	Color, intensity	Octree	During loading, the memory usage is increasing steadily, and AHN3delft causes computer frozen. There is no memory crashing mechanism.	Several times larger than data itself, mainly due to additional octree data.
Fugro viewer	Color, class, intensity, sourceID, return number	No LoD, data is stored in both 2D and 3D	Data is loaded into memory gradually and slowly. It pops up a warning for shortage of memory for AHN3delft, and loading process is frozen after a while.	2D and 3D scenes both occupy memory space. Memory only changes when 2D and 3D scenes exist simultaneously, and there is a zooming operation in the 2D layer.
Cloud compare	Color, intensity	No LoD, but support octree computation	When data is very large, say, the AHN3delft, very slow loading, but no frozen, no crashing.	New operations do not change memory occupation (or very little change)
AHN2-viewer	Height, intensity, sourceID	Octree	It depends on network speed.	New operations raise memory usage which after a while decreases due to smart caching strategy.
Plasio	Color, intensity, class	No	Loading is very fast. But the AHN3delft fails immediately, and it seems that the web application can judge whether memory is capable for handling the data, i.e. pre-calculation	Data is totally buffered into memory without additional cost.
Scene mark	Color	No	Loading process is square tile based, i.e. loading one block after the other sequentially.	Hundreds of times smaller than the data itself. But the zooming operation can increase memory usage

smartPC model including cLoD should be devised considering new contextual environment. A process including 3-step use cases including visualization, GPS trajectory mining and visibility detection will be established regarding different number of dimensions. Then, implementation of the smart conceptual model inside a database is integrated with a nD PointCloud structure which particularly focuses on the efficiently organizing dimensions for use cases. The structure will then be verified against use cases to be tuned and improved. The final realization of the nD PointCloud structure will be deployed on a cloud system with specified compression, transmission and caching strategies. In the end, benchmark will be executed to tune the data storage as well as evaluate the performance of the new data structure.

Acknowledgments

The authors would like to thank the Chinese Scholarship Council (CSC) and Fugro for supporting this research.

References

Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M. and Savarese, S. (2016). 3D semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1534-1543).

Kodde, M. (2010). The art of collecting and disseminating point clouds. *NCG KNAW*, pp. 9–15.

Liu, H., van Oosterom, P., Hu, C., and Wang, W. (2016). Managing large multidimensional array hydrologic datasets: a case study comparing netCDF and SciDB. *Procedia Engineering*, 154:207–214.

Poux, F., Neuville, R., Hallot, P., and Billen, R. (2017). Model for reasoning from semantically rich point cloud data. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pp. 107–115.

Psomadaki, S. (2016). *Using a space filling curve for the management of dynamic point cloud data in a relational DBMS*. Master's thesis, Delft University of Technology, the Netherlands.

Richter, R. and Döllner, J. (2014). Concepts and techniques for integration, analysis and visualization of massive 3D point clouds. *Computers, Environment and Urban Systems*, 45, pp.114-124.

Sagan, H. (2012). Space-filling curves. *Springer Science & Business Media*.

Van Winden, K., Biljecki, F., and Van der Spek, S. (2016). Automatic update of road attributes by mining GPS tracks. *Transactions in GIS*, 20(5):664– 683.

Van Oosterom, P. (1999). Spatial access methods. *Geographical information systems*, 1, pp.385-400.

Van Oosterom, P., Martinez-Rubi, O., Ivanova, M., Horhammer, M., Geringer, D., Ravada, S., Tijssen, T., Kodde, M., and Goncalves, R. (2015). Massive point cloud data management: Design, implementation and execution of a point cloud benchmark. *Computers & Graphics*, 49:92–125.

Van Oosterom, P., Martinez-Rubi, O., Tijssen, T., and Goncalves, R. (2016). Realistic benchmarks for point cloud data management systems. In: *Advances in 3D Geoinformation*, pp. 1–30. Springer International Publishing.