Graduation Plan
SplitSFC: A database solution for massive point cloud data management

Yuduan Cai
5625483

1st supervisor: Dr.ir. B.M. Meijers
2nd supervisor: Prof.dr.ir. P.J.M. van Oosteromg

January, 2023

Contents

1 Introduction

1.1 Problem statement

A point cloud is a discrete set of data points in space, and is one of the most widely used formats for 3D shapes and objects. The most significant advantage is its simplicity and efficiency in representing 3D data. It stores only the coordinates of individual points in space, rather than the complex geometry of surfaces and shapes, and may contain other attributes like intensity. This makes it easy to process and analyze large amounts of 3D data, such as from LiDAR sensors or 3D scans. Additionally, point cloud data can be easily visualized, shared, and integrated with other software and systems. With the developments in the point cloud acquisition technologies, like terrestrial and airborne laser scanning, mobile mapping, image matching and multi-beam echo-sound techniques etc., it becomes easier and easier to acquire data. Therefore, it is widely used in industries such as surveying, architecture, engineering, construction, and robotics for applications such as building information modeling (BIM), 3D scanning, and autonomous navigation, and the popularity is expected to continue to grow in the coming years as the technology becomes more widely available and the cost of data acquisition decreases (Psomadaki, 2016).

The majority of the application of point cloud data is currently done at the file level, but an efficient database-based solution is still crucial, for database management systems (DBMS) has advantages in functionalities, optimized disk IO strategies and automatic parallelization in the query executions (Van Oosterom et al., 2015). There are many commercial or open-source databases available and suitable to tackle this issue, like Oracle, PostgreSQL, MonetDB, etc. Researchers and engineers have explored this field for many years. However, the database solutions are still not mature yet. And the specific approach will depend on the use case, the size and the structure of the data, and the type of queries that will be made. The most significant challenge is the size of the point cloud data. Point clouds can be very large, with billions of points, and traditional relational databases are not well suited for handling this amount of data. Other challenges include the need for fast querying and indexing, and the need to support multi-dimensional data. To improve the performance, common techniques include compression, sampling, indexing, partitioning, database optimization and so on. One approach is to use Space Filling Curve (SFC) to group and encode the points, but the size is still very large, and the query can be slow.

In this research, we will propose a storage model called SplitSFC to increase the efficiency of storage and manipulation of point clouds in databases. The approach is to split the SFC key in a head and tail part and make blocks of points inside the database based on the head, resulting in a more efficient storage. The effect on the manipulation of the data will also be investigated, like the efficiency and simplicity of retrieving and querying points. The expected outcome includes a storage model, a Python-based data importer and a scientific benchmark.

1.2 Scientific value

The size of the occupied memory space and the querying speed are two important factors of the user requirements of the existing point cloud database-based solutions. Since point cloud data is typically very large, performance has a big effect on the application. For instance, it is unable to offer a web application with bad performance steady service. Despite years of testing with many databases and data models, issues with occupied storage space and query speed occasionally arise. If the improved efficiency of this new data model is validated in some scenarios, it may offer societal and scientific advantages for better point cloud data management through storage and manipulation.

2 Theoratical background and related work

2.1 Point cloud

A point cloud is a set of points in three-dimensional space. Point clouds are multi-dimensional data. Each point usually contains XYZ coordinates and other attributes, like intensity, classification, RGB colour, etc. The major sources of the point clouds include the Light Detection and Ranging (LiDAR) systems, photogrammetry, 3D scanning, navigation systems and conversion from other types.

Van Oosterom et al. (2015) proposed that point clouds should be recognized as a distinct form of spatial data representation, alongside vectors and rasters. While point clouds exhibit similarities with vector-based structures due to their point-based nature, they also share characteristics with raster data owing to their sampling nature. However, point clouds are irregularly scattered, and differ from individual points or multi-points in vectors. Point clouds are gaining more popularity and importance in the industry and academia. Hence, it is necessary to acknowledge them as the third spatial data type and build standardization for wide usage.

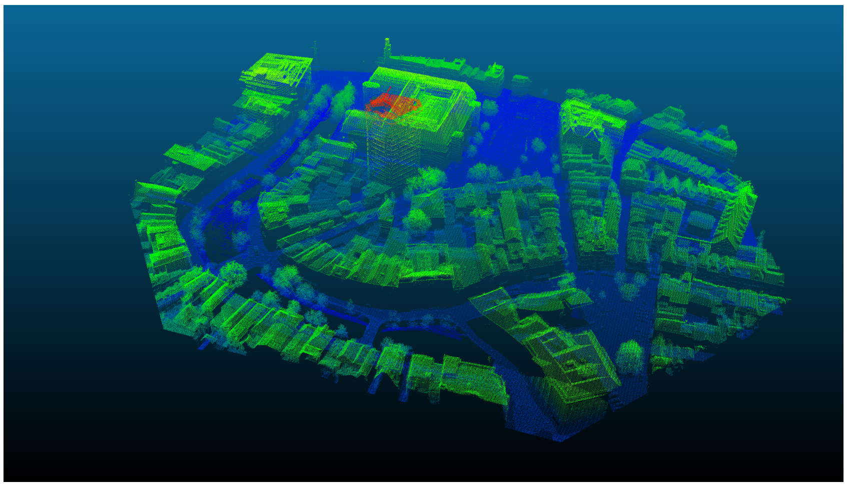


Figure 1: Part of the point cloud of the Dutch National Digital Surface Model

2.2 Point cloud data management

The data management system of point cloud data can be categorized into file-based systems or database management systems (DBMS). The key challenge is to handle massive data and offer standardized functionalities at the same time (Van Oosterom et al. (2015)).

File-based systems use a file or a set of files to store the point cloud data, and rely on softwares to edit, process, analyse and visualize them. File-based systems are flexible and efficient, and functionalities can be supported by the softwares. Therefore, currently most applications are based on file-based systems. However, it lacks native functionality support, and the data retrieval has to go over many files, which is not convenient.

The DBMS, on the other hand, manages the storage and the retrieval both based on the DBMS itself. Structured Query Language (SQL) can be used to query and update the data. External software and programming language are usually used for complex algorithms only. The storage and querying can be more efficient with native support, like parallel process techniques. Therefore, it is of high practical value to perform research on the DBMS solutions for massive point clouds.

2.2.1 File-based systems

The most commonly-used file formats is LAS and its lostless compression LAZ. LAS (LiDAR Aerial Survey) is a binary file format for storing LiDAR point cloud data. It is the industry standard for exchanging LiDAR data and is used by many commercial and open-source LiDAR processing software. The format includes a header section that contains information about the data, such as the coordinate system, the number of points, and the scale factors for the x, y, and z coordinates. The data section contains the actual LiDAR point data, which includes the x, y, and z coordinates of each point, as well as additional information such as the intensity, return number, and classifications. LAS files can also contain additional information such as waveform data and RGB colour information. The format also supports the use of variable-length records, which allows for the inclusion of additional information beyond the standard point data. The most common attributes of LAS files are shown in Table 1.

Attribute	**Format**	**Bits)**	**Description**
X	int	32	X coordinate.
Y	**int**	32	Y coordinate.
Z	int	32	Z coordinate.
Intensity	unsigned int	16	The pulse return amplitude.
Return number	unsigned int	3	The total pulse return number for a given output pulse.
Number of returns	unsigned int	3	Total number of returns for a given pulse.
Scan Direction Flag	boolean	1	Denotes the direction at which the scanner mirror was travelling at the time of the output pulse.
Edge of Flight Line	boolean	1	Denotes whether the point is at the end of a scan.
Classification	unsigned int	8	Classification code.
Scan Angle Rank	int	4	The angle at which th+e laser pulse was output from the scanner including the roll of the aircraft.

Table 1: The dimensions in LAS files
Source: Ledoux et al, 2022

2.2.2 DBMS

The ongoing discourse surrounding the applicability of Database Management Systems (DBMS) in handling point cloud data remains a topic of continuous investigation (Liu, 2022). While file-based solutions maintain popularity among users, database solutions offer distinct advantages such as enhanced functionalities, optimized disk IO strategies, and automated parallelization during query executions (Van Oosterom et al., 2015).

However, the development of a DBMS solution is consistently confronted with challenges. Among these challenges are the management of huge data volumes, high dimensionality of the data, and the imperative need for efficient querying and indexing mechanisms tailored to point cloud datasets. Furthermore, researchers and developers keep on seeking improved solutions, indexing structures, and query languages for point cloud data.

In Section 2.3, we explain the two design aspects of a DBMS solutions for point clouds. One is the type of storage model, whether use a flat table or a block-based model to organize the points. The other aspect is the Spatial Access Methods, which includes spatial indexing and clustering, and is crucial for an efficient storage. In Section 2.4, we give an overlook of the state-of-the-art DBMS solutions for point clouds.

2.3 Design aspects of a DBMS solution

2.3.1 Flat and block model

The storage model can be devided into two categories based on whether one row store one or multiple points:

- **Flat model** stores the points in a single table, with each row representing a point.

- **Block model** organizes the points into blocks, and each block contains a subset of the points. The grouping is usually based on spatial clustering.

While the flat model is simple, it can become impractical for large point clouds, as the storage size and the time complexity of queries increase. One of the main drawbacks is the poor performance of spatial queries, as it is required to scan the entire table to find the points that meet the query conditions, especially spatial queries such as finding all points within a certain bounding box or within a certain distance of a specified point.

Block model can be more efficient in terms of storage and retrieval, therefore it is more suitable for large point clouds, but it can be more complex to implement and use.

2.3.2 Spatial Access Methods

Spatial access methods are data structures and algorithms used in spatial databases to increase the efficiency of retrieving and querying spatial data. It can optimise the performance of the database.

There are two aspects of spatial access methods. One is spatial indexing (van Oosterom, 1999). It works by creating a separate data structure, called a spatial index, to organise the spatial data. A spatial index typically consists of a hierarchical tree-like structure, such as R-Tree, Quad-Tree, k-d Tree, etc. The other aspect is spatial clustering, which is to group data points that are spatially close to each other. The algorithms work by identifying clusters of data points based on their spatial properties, such as distance or density. The result of spatial clustering is a set of clusters, each containing a group of data points that are geographically close to each other.

Space filling curve is a continuous, self-repeating curve that traverses through every point in a given space. It can can map multidimensional data onto one dimension, and consequently reduce the dimensionality of data. The encoding will be their SFC keys. Space Filling Curve also have spatial clustering effect. Common Space Filling Curves include Morton curve and Hilbert curve, as they both have good spatial clustering effect.

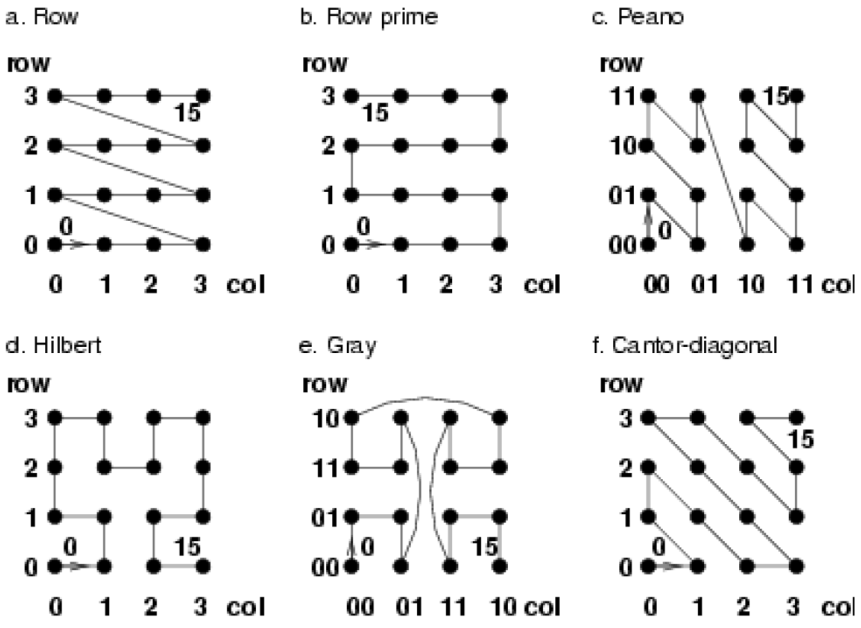


Figure 2: Commonly used Space Filling Curve
Source: van Oosterom (1999)

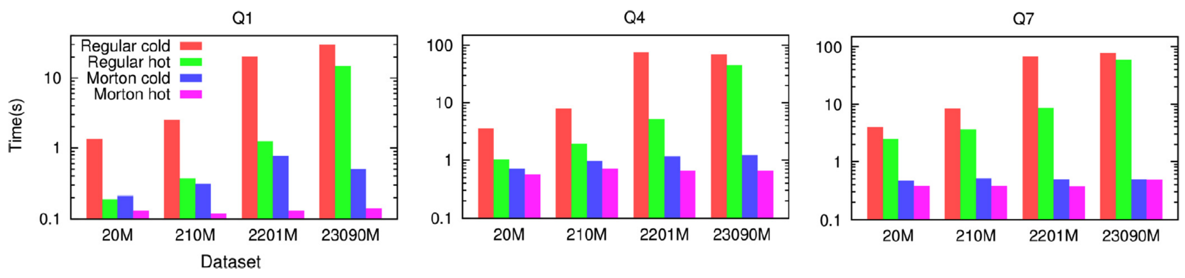


Figure 3: Comparison of the response times of three queries in regular PostgreSQL flat table and the Morton-based flat table
Source: Van Oosterom et al. (2015)

The dimensions can be categorized as organizing dimensions and associate dimensions, depending whether they are encoded into the SFC keys or not. The selection of the organizing dimensions depends on the use case. For example, for a point cloud that represents the digital surface model, we usually perform 2D geometry queries, then we can encode x and y dimension into the SFC keys. All organizing dimensions should be transformed into integer data type before the encoding.

2.4 Existing DBMS solutions

The most popular DBMS to handle point cloud data are PostgreSQL and Oracle. PostgreSQL is a powerful open-source DBMS, and it has a very powerful spatial extension called PostGIS. Oracle, on the other hand, is a commerical software, and it has native support for spatial data.

In this section, we only mention the existing DBMS solutions using PostgreSQL and Oracle. Depending on whether a row stores one or multiple points, the solutions can be categorized into flat and block model, as mentioned in Section 2.3.

2.4.1 Flat model

Normal flat table

The points are stored in a normal flat table, where each row stores the dimensions of a point. A B-tree index can be built on X and Y coordinates to improve the querying efficiency. The geometry of the points can be stored with normal numberal values, or using geometry data type, such as the Point data type from PostGIS.

SFC-based flat table

Van Oosterom et al. (2015) proposed an approach that uses Space Filling Curve to fasten the querying. In this approach, points are stored in a flat table with a Space Filling Curve encoding. The main table contains columns of x, y, z and their SFC key. A B-tree index is created on the Morton code and the points are reordered by the index. The query algorithms are based on the relationship between Morton curve and 2^n-tree.

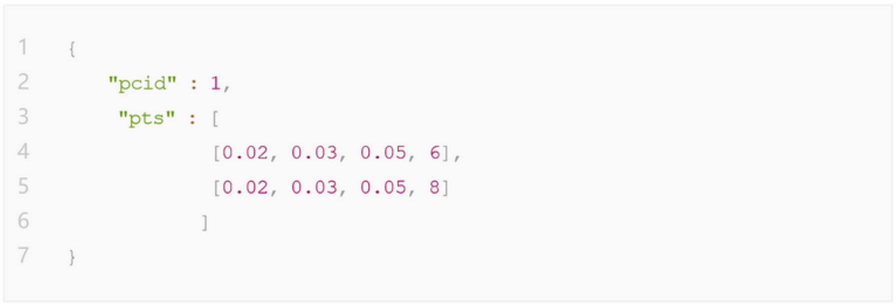
The benchmark results in Figure 3 show that the queries in the Morton-based flat table are much faster and more scalable thanks to the better spatial data organization and the smarter spatial accessing strategy.

2.4.2 Block model

pgPointCloud

pgPointCloud is an extension for storing and querying point cloud data in PostgreSQL (Ramsey, 2013). This extension allows the user to create point cloud tables and indexes, and provides functions like filtering and slicing. It is efficient and robust. The actual loading of the point cloud can take place either by making use of well-known binary (WKB) objects or with the PDAL driver for pgpointcloud.

The PC_Patch data type in pgPointCloud stores points in blocks. A PC_Patch record stores the ID of the point block and a list of points with all the dimensions (Figure 4). The records are stored in a TOAST (The Oversized-Attribute Storage Technique) table.



```
{
    "pcid" : 1,
     "pts" : [
              [0.02, 0.03, 0.05, 6],
              [0.02, 0.03, 0.05, 8]
             ]
}
```

Figure 4: An example of PC_PATCH schema in pgPointCloud

Oracle SDO_PC

The block-based model in Oracle usually stores point blocks as Binary Large OBject (BLOB) types in a table. The most widely used approach is based on SDC_PC data type (Oracle, 2019).

There are two tables in the schema. One is the Base PC Table. It contains SDO_PC objects (Oracle, 2019) and their identifiers. The other table is the Block Table. Its columns contain SDO_PC_BLK object (point blocks), block ID, the number of points in the block, etc. A R-tree or a Hilbert-R tree can be built as an index. However, the organization of blocks only supports XY dimensions, and all other dimensions can only be stored as property dimensions.

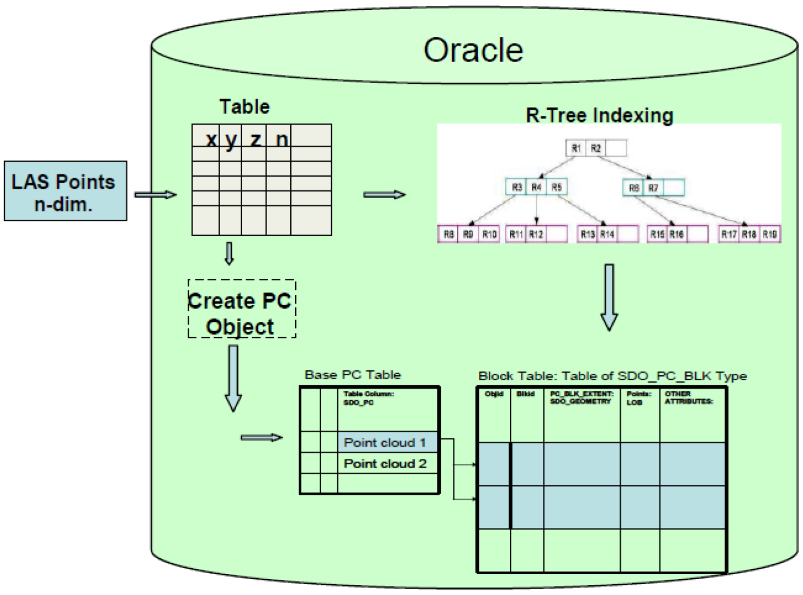


Figure 5: The schema for Oracle SDO_PC object
Source: Alfarrarjeh et al. (2020); Liu (2022)

3 Research objectives

3.1 Research questions

In the research of literature and existing methods, we found that one key challenge of DBMS solutions is to reduce the occupied storage space because of the large volume of data. Existing solutions often employ space filling curves to improve performance. Noticing that SFC keys have many characters, and the keys of similar points usually have repeated parts, we can consider designing a new storage model, split SFC keys into two, and store them in blocks. We call the approach SplitSFC. We will study its impact on improvements in storage and operational efficiency. Therefore, the main research question is addressed as follows:

What is an appropriate block-based storage model based on split SFC keys to efficiently store and manipulate point clouds in the relational database?

In order to answer the main question, the following sub-questions are proposed:

- How to encode and decode SFC keys with the organising attributes?
- How to split the SFC keys, i.e. by a fixed ratio?
- How to group the points with the same SFC key head?
- How to perform querying based on split SFC keys?
- How does the suggested method work compared to the current way of working?

3.2 Research scopes

The main focus of the research is to optimise the performance of the database solution of point cloud data with an improved storage model called SplitSFC. The methodology suggested will focus on splitting SFC keys and grouping points into blocks based on it. For example, we will explore how to split the SFC keys, how to split the SFC keys (e.g. by half), and whether the strategy may vary with different organising attributes and use cases. The benchmark will show the performance of the model. Morton curve will be used, but comparison of the effect of different space filling curves is not involved in this research. Eventually, a scientific benchmark and comparison of different strategies will be made to validate the practicality of the new methodology.

4 Methodology

In the research, a new storage model called SplitSFC to store massive point clouds in the database will be designed. To implement the storage model, a Python-based importer and querying tool are built to pre-process and load raw data and into the database, and perform queries. Additionally, a benchmark will be performed to analyse the performance of the new schema and provide guidance in the model design. The overview of the loading procedure is shown in Figure 6.

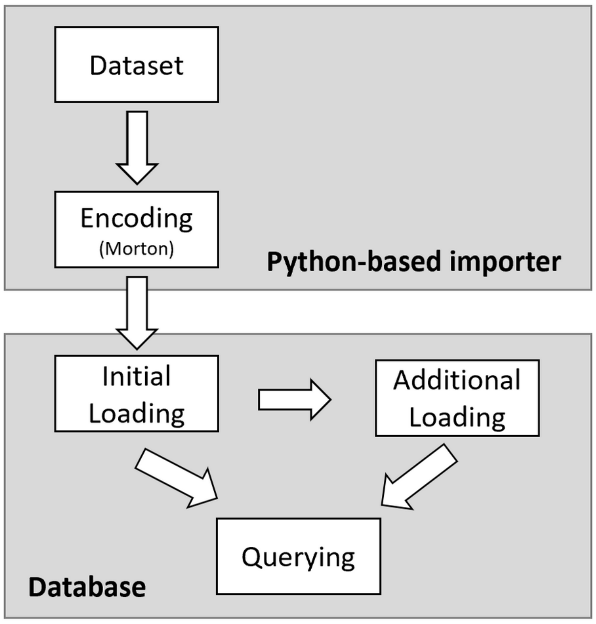


Figure 6: An overview of the SplitSFC loading procedure

4.1 User requirement

To design and benchmark the data model, the primary thing is to define the user requirement. It may vary per use case, but since the thesis aims to provide a general solution, a general user requirement will be applied to guide the research. The final product should support the following functionalities:

- **Loading**: It can (a) read LAS/LAZ files; (b) convert raw data into the designed format, and (c) load them into the database. The pre-processing procedures include Morton encoding, Morton key splitting and grouping the points based on SFC head.
- **Querying**: As it is a basic prototype, it only supports data selection based on 2D geometry and height. The possible 2D geometry query include simple 2D range / rectangle filters of various sizes, 2D polylines with different buffer sizes and 2D polygons.
- **Exporting**: It can export the selected data in the table to a LAS file.

4.2 Split Morton Curve

In this research, the Morton curve will be used, because its simplicity in algorithm and preservation of the spatial locality. which means that data points that are close to each other in the original multi-dimensional space will also be close to each other in the one-dimensional space.

The algorithm of encoding and decoding Morton key is bitwise interleaving, and it is easy to implement and has low overhead.

We can explain the Morton encoding algorithm with a case of encoding x, y and z coordinates. To encode a point in a 3D space using the Morton curve, x, y and z coordinates of the point are first split into binary digits. The digits are then interleaved to create a new binary number, which represents the position of the point on the curve. For example, if the x, y, and z coordinates of the points are 7 (binary 111), 3 (binary 011) and 5 (binary 101), the resulting Morton code would be 111011101. Furthermore, if we consider time, level of detail and other dimensions as the organising attributes, this encoding can be extended to higher dimensions by interleaving the binary digits of each dimension to form a single number.

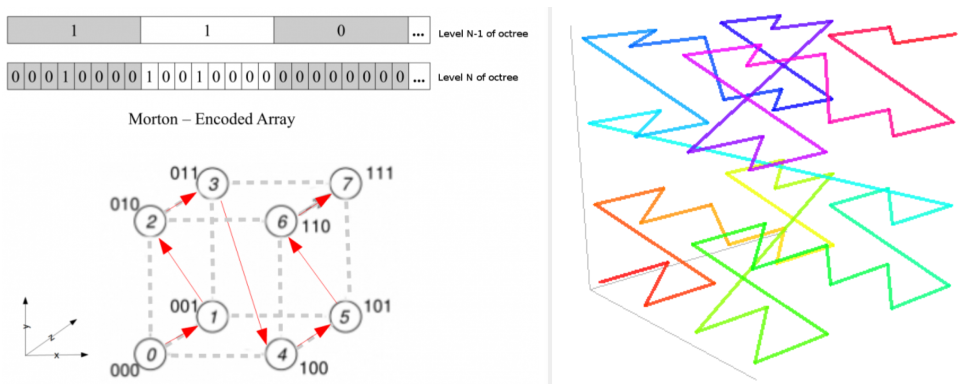


Figure 7: The 3D encoding using Morton curve
Source: Baert (2013)

In addition, Morton Curve can represent the hierarchical structure of 2^n-tree. Figure 8 illustrates the node and the range in 2D. All points have integer coordinates. By truncating the last n bits of the Morton codes of the points recursively, Morton codes at upper levels are derived. That is to say, the Morton codes of points implicitly have a hierarchy which is equivalent to a Quadtree structure. A branch node covers the nodes on the level below, and represents the spatial extent of a quadrant. Thus, the branch node also indicates a range of Morton codes starting from the lower-left corner to the upper-right. The property is extensible to higher dimensions because of the Quadrant recursive properties of Morton Curve.

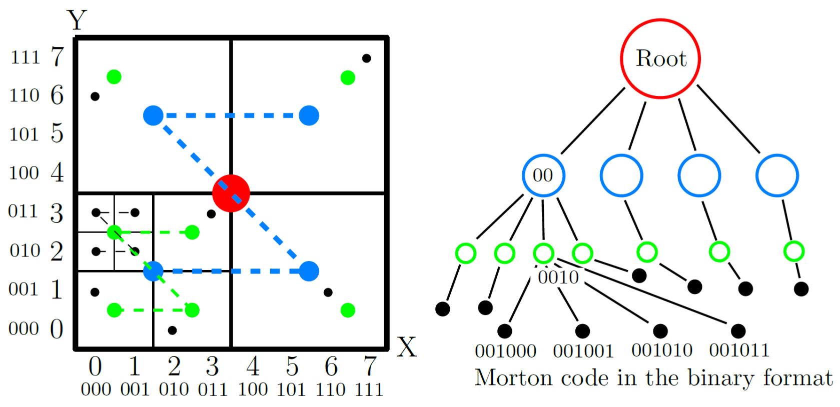
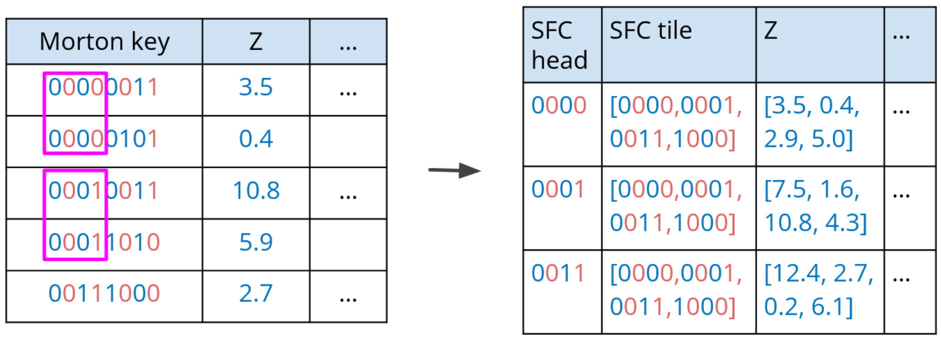


Figure 8: The hierarchical structure of Morton Curve and Quadtree
Source: Haicheng Liu (2020)

The main innovation is that the split of SFC keys. In Figure 9, we can see that in a Morton-based flat table, many points have same parts in the Morton key (in purple rectangle). We can consider to split each SFC key into a head and tile part, and we group the points with the same SFC head into blocks. It can potentially save storage space because each SFC head is stored once and shared with many points.



Morton key	Z	...
00000011	3.5	...
00000101	0.4	
00010011	10.8	...
00011010	5.9	
00111000	2.7	...

SFC head	SFC tile	Z	...
0000	[0000,0001, 0011,1000]	[3.5, 0.4, 2.9, 5.0]	...
0001	[0000,0001, 0011,1000]	[7.5, 1.6, 10.8, 4.3]	...
0011	[0000,0001, 0011,1000]	[12.4, 2.7, 0.2, 6.1]	...

Figure 9: Grouping the points with the same SFC head into blocks

4.3 Storage model

The schema of the SplitSFC approach is shown in Figure 10. There are two tables in the database. One is the metadata table that stores the description of the dataset, such as name, Spatial Reference ID, the number of points, spatial extent and the splitting information. The other data stores the dimensions of each point. It includes SFC head, SFC tile and other associate dimensions. Points with the same SFC head are stored in one block, and their SFC tiles and other associate dimensions are stored in arrays. More over, a B-tree index is built on sfc_head column.

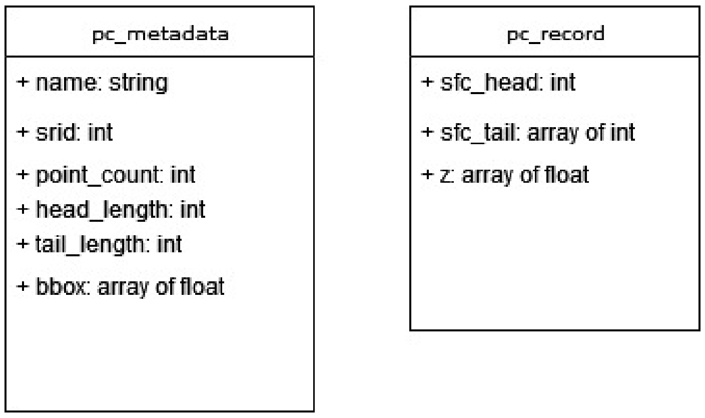


Figure 10: The schema of the SplitSFC approach

4.4 Benchmark

The goal of benchmark is to measure performance of the storage solution. We adpot the benchmark framework proposed by Van Oosterom et al. (2015). The detailed information of the used dataset, Actueel Hoogtebestand Nederland 2 (AHN2), is introduced in Section 5.2.

Four datasets have been tailored to perform the benchmark, varying from 20 million points in TU Delft campus to 20 billion points in South Holland Province. Each dataset stores ten times more points than the previous dataset. Table 2 and Figure 11 show the details of the dataset and an approximate spatial extent projection.

The original benchmark framework includes three stages:

- **Mini-benchmark**: It starts with a small dataset of about 20 million points in Delft Campus. In this stage, we can explore the effects of different parameters, and the results serve as a basis to decide on parameter settings in the next stage benchmark.
- **Medium-benchmark**: In this stage, the benchmark uses several dataset up to 23090M with about 20 billion points. It explores the scaling behaviour of the database solution, and extends the querying functionalities of the mini-benchmark.
- **Full-benchmark**: In this stage, the complete AHN2 dataset is loaded into the database and queried. However, we only perform mini-benchmark and medium-benchmark in our experiments.

Name	Points	Files	Size (GB)	Area (km2)	Description
20M	20,165,862	1	0.4	1.25	TU Delft Campus
210M	210,631,597	1	4	11.25	Major part of Delft city
2201M	2,201,135,689	153	42	125	City of Delft and surroundings
23090M	23,090,482,255	1492	440.4	2000	Major part of South Holland

Table 2: Dataset description
Source: Van Oosterom et al. (2015)

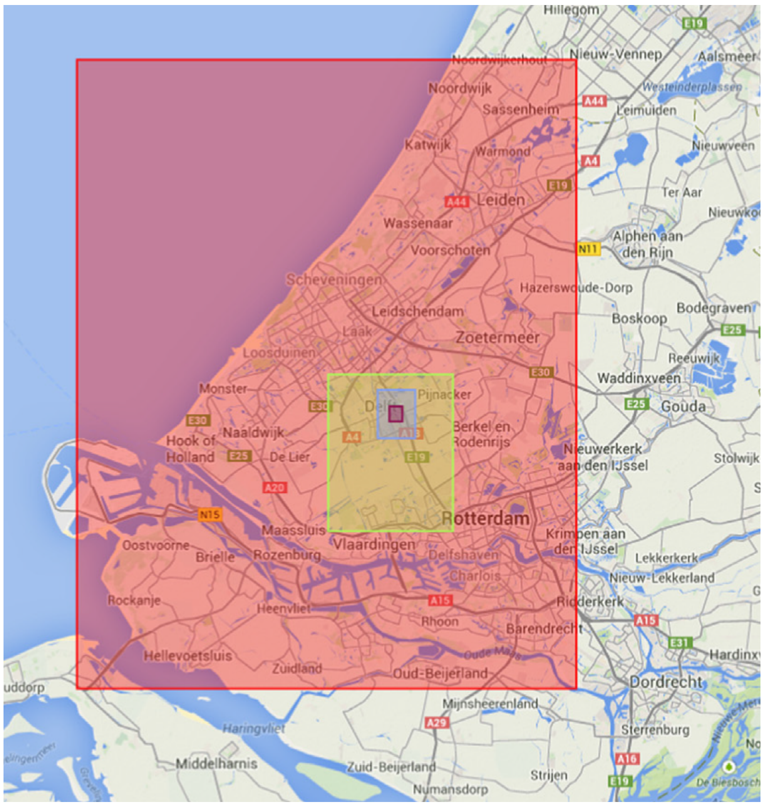
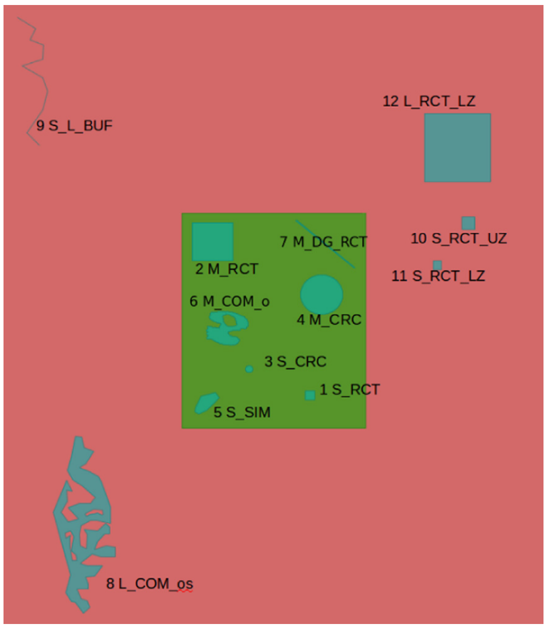
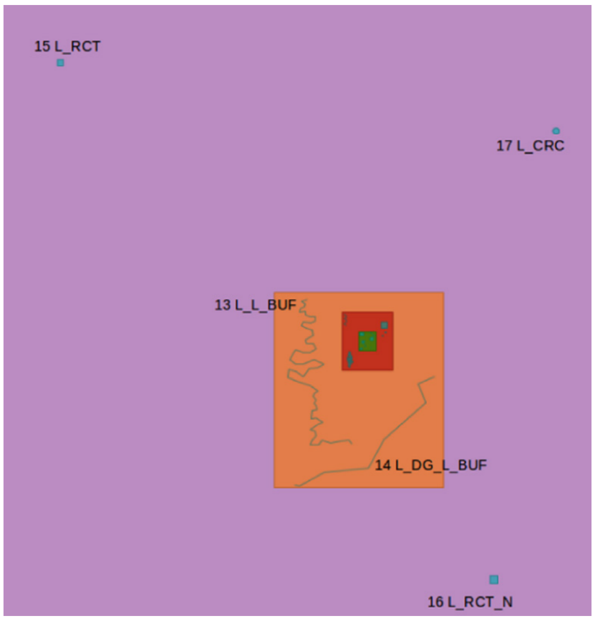


Figure 11: Approximated projection of the extents of the used datasets in Google Maps: Purple for 20M, blue for 210M, green for 2201M, red for 23090M
Source: Van Oosterom et al. (2015)



(a) Up to 210M dataset



(b) Up to 23090M dataset

Figure 12: Queries used in the medium benchmark
Source: Van Oosterom et al. (2015)

The geometry queries have been well designed too. We list all queries in Table 3. Most are selected within a 2D query region, and the types of regions include rectangles (axis-aligned or diagonal), circles, simply polygons, polygons with holes, buffer of polylines, etc. The geometries are shown in Figure 12. Some queries also include the selection on the z dimension. The query return method is Create Table As Select (CTAS), which means the returned points are stored in a new table.

Dataset	ID	Key	Area (km2)	Description
20M	1	S_RCT	0.0027	Small axis-aligned rectangle
	2	M_RCT	0.0495	Medium axis-aligned rectangle
	3	S_CRC	0.0013	Small circle, radius 20 m
	4	M_CRC	0.0415	Medium circle, radius 115 m
	5	S_SIM	0.0088	Small, simple polygon
	6	M_COM_o	0.0252	Medium, complex polygon, 1 hole
	7	M_DG_RCT	0.0027	Medium, narrow, diagonal rectangular area
210M	8	L_COM_os	0.1341	Large, complex polygon, 2 holes
	9	S_L_BUF	0.0213	Small polygon (10 m buffer in line of 11 pts)
	10	S_RCT_UZ	0.0021	Small axis-aligned rectangle, cut in max. z
	11	S_RCT_LZ	0.0051	Small axis-aligned rectangle, cut in min. z
	12	L_RCT_LZ	0.1419	Large axis-aligned rectangle, cut in min. z
2201M	13	L_L_BUF	0.0475	Large polygon (1 m buffer in line of 61 pts)
	14	L_DG_L_BUF	0.0499	Large polygon (2 m buffer in diag. line of 8 pts)
23090M	15	L_RCT	0.2342	Large axis-aligned rectangle
	16	L_RCT_N	0.1366	Large axis-aligned rectangle in empty area
	17	L_CRC	0.1256	Large circle

Table 3: Query description
Source: Van Oosterom et al. (2015)

5 Tools and datasets

5.1 Hardware and software

5.1.1 Hardware

Because we adpot the benchmark framework developed by Van Oosterom et al. (2015), we run the programs on exactly the same hardware, which is the supercomputer of the GIS Technology Group. The supercomputer is called pakhuis, and it includes:

- A HP DL380p Gen8 server with 128 GB RAM and 2 × 8 Intel Xeon processors E5-2690 at 2.9 GHz;
- RHEL 6 as operative system and different disks directly attached including 400 GB SSD;
- 5 TB SAS 15 K rpm in RAID 5 configuration (internal);
- 2 × 41 TB SATA 7200 rpm in RAID 5 configuration (in Yotta disk cabinet).

5.1.2 Software

The main softwares for the implementation are PostgreSQL and Python.

- **PostgreSQL** is the DBMS used for the storage and manipulation of point cloud data. It is one of the most advanced open-source relational database management systems, and has good spatial support with PostGIS.
- **Python** handles complex calculation outside the database such as SFC encoding and decoding. It was chosen for its simplicity, but it is much slower than lower-level languages like C++ due to its interpreted nature. We can use either psycopg2 package or SQLAlchemy package to connect Python and PostgreSQL.

5.2 Datasets

We employ the Actueel Hoogtebestand Nederland 2 (AHN2) dataset to evaluate performance. This dataset comprises a total of 640 billion points, with a sample density ranging from 6 to 10 points per square meter across the country. The Spatial Reference System for AHN2 is the Amersfoort/RD New, identified by the Spatial Reference System Identifier (SRID) 28992. All datasets involved in these experiments are formatted in LAS and contain solely XYZ dimensions. We only perform the mini-benchmark and the medium benchmark from the original benchmarking framework developed by Van Oosterom et al. (2015), as mentioned in Section 4.4.

6 Time planning

The thesis has five major milestones. The timetable is displayed in Table 4, and Table 5 lists the milestones and their accompanying deliverables. Meetings with the first supervisor will typically take place twice a week. Work will cease during national holidays.

Period	**Task**	**Date**	
P1: Initial idea	Idea generation & literature review	01-11-2023	18-11-2023
P2: Research proposal	Literature review	18-11-2022	20-01-2023
	Theory study	18-11-2022	20-01-2023
	User requirement analysis	10-01-2023	13-01-2023
	Initial model design	16-01-2023	20-01-2023
	Writing a research proposal	16-01-2023	**27-01-2023**
P3: Mid-term	Implementation	29-01-2023	17-02-2023
	Model modification	17-02-2023	31-03-2023
	Benchmarking	17-02-2023	**05-04-2023**
P4: Green Light	Refinements	06-04-2023	01-05-2023
	Writing thesis	01-05-2023	**02-06-2023**
P5: Defense	Public presentation	**05-07-2023**	

Table 4: Time planning of the thesis process

Period	**Due date**	**Deliverables**
P2: Project Plan	27-01-2023	A project plan A presentation with the graduation committee
P3: Mid-term	**05-04-2023**	Draft code A benchmark report A meeting with the supervisors
P4: Green light	02-06-2023	Draft thesis paper A Github repository with code, documentation, etc. A meeting with the graduation committee
P5: Defense	05-07-2023	Final thesis paper A public presentation

Table 5: Deliverables in each period

The eventual deliverables will include:

- A thesis paper
- Code in a Github repository
- Documentation with examples and tutorials about how to use the code

The graduation committee includes:

- Martijn Meijers: First supervisor from GIS Technology Group, TU Delft
- Peter van Oosterom: Second supervisor from GIS Technology Group, TU Delft
- Ken Arroyo Ohori: Co-reader from 3D Geo-information Group, TU Delft
- Ellen Geurts: The delegate of the faculty

References

A. Alfarrarjeh, S. H. Kim, V. Hegde, C. Shahabi, Q. Xie, S. Ravada, et al. A class of r*-tree indexes for spatial-visual search of geo-tagged street images. In *2020 IEEE 36th international conference on data engineering (ICDE)*, pages 1990–1993. IEEE, 2020.

J. Baert. Morton encoding/decoding through bit interleaving: Implementations. 2013. URL `https://www.forceflow.be/2013/10/07/morton-encodingdecoding-through-bit-interleaving-implementations/`.

R. Bayer and E. McCreight. Organization and maintenance of large ordered indices. In *Proceedings of the 1970 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control*, pages 107–141, 1970.

H. Butler, B. Chambers, P. Hartzell, and C. Glennie. Pdal: An open source library for the processing and analysis of point clouds. *Computers & Geosciences*, 148:104680, 2021.

J. Chen, L. Yu, and W. Wang. Hilbert space filling curve based scan-order for point cloud attribute compression. *IEEE Transactions on Image Processing*, 31:4609–4621, 2022.

R. Cura, J. Perret, and N. Paparoditis. Point cloud server (pcs): Point clouds in-base management and processing. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2:531–539, 2015.

R. Cura, J. Perret, and N. Paparoditis. A scalable and multi-purpose point cloud server (pcs) for easier and faster point cloud data management and processing. *ISPRS Journal of Photogrammetry and Remote Sensing*, 127:39–56, 2017.

V. Gaede and O. Günther. Multidimensional access methods. *ACM Computing Surveys (CSUR)*, 30(2):170–231, 1998.

X. Guan, P. Van Oosterom, and B. Cheng. A parallel n-dimensional space-filling curve library and its application in massive point cloud management. *ISPRS International Journal of Geo-Information*, 7(8):327, 2018.

M. M. X. G. E. V. M. H. Haicheng Liu, Peter van Oosterom. Histsfc: Optimization for nd massive spatial points querying. *International Journal of Database Management Systems*, 12(3): 7–28, 2020.

D. Hilbert. Über die stetige abbildung einer linie auf ein flächenstück. *Dritter Band: Analysis· Grundlagen der Mathematik· Physik Verschiedenes: Nebst Einer Lebensgeschichte*, pages 1–2, 1935.

M. Isenburg. Laszip: lossless compression of lidar data. *Photogrammetric engineering and remote sensing*, 79(2):209–217, 2013.

ISPRS. Las 1.4 format specification (tech. rep.). 2019. URL `http://www.asprs.org/wp-content/uploads/2019/07/LAS_1_4_r15.pdf`.

G. Jin and J. Mellor-Crummey. Sfcgen: A framework for efficient generation of multi-dimensional space-filling curves by recursion. *ACM Transactions on Mathematical Software (TOMS)*, 31(1):120–148, 2005.

J. K. Lawder. *The application of space-filling curves to the storage and retrieval of multi-dimensional data*. PhD thesis, Citeseer, 2000.

J. Li. Manage 4d historical ais data by space filling curve. 2020.

H. Liu. nd-pointcloud data management: continuous levels, adaptive histograms, and diverse query geometries. *A+ BE— Architecture and the Built Environment*, (12):1–206, 2022.

H. Liu, P. Van Oosterom, M. Meijers, and E. Verbree. An optimized sfc approach for nd window querying on point clouds. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 6:119–128, 2020.

M. Meijers. Pcserve–nd-pointclouds retrieval over the web. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 10:193–200, 2022.

G. M. Morton. A computer oriented geodetic data base and a new technique in file sequencing. 1966.

W. Nulty and J. Barholdi III. Robust multidimensional searching with spacefilling curves. In *Proceedings of the 6th International Symposium on Spatial Data Handling, Edinburgh, Scotland*, pages 805–818, 1994.

S. Psomadaki. Using a space filling curve for the management of dynamic point cloud data in a relational dbms. 2016.

P. Ramsey. Lidar in postgresql with pointcloud. *FOSS4G, Nottingham*, 2013.

R. Richter and J. Döllner. Concepts and techniques for integration, analysis and visualization of massive 3d point clouds. *Computers, Environment and Urban Systems*, 45:114–124, 2014.

P. van Oosterom. Spatial access methods. *Geographical information systems*, 1:385–400, 1999.

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

P. van Oosterom, O. Martinez-Rubi, T. Tijssen, and R. Gonçalves. Realistic benchmarks for point cloud data management systems. *Advances in 3D Geoinformation*, pages 1–30, 2017.

P. van Oosterom, S. van Oosterom, H. Liu, R. Thompson, M. Meijers, and E. Verbree. Organizing and visualizing point clouds with continuous levels of detail. *ISPRS Journal of Photogrammetry and Remote Sensing*, 194:119–131, 2022.

J. Wang and J. Shan. Space filling curve based point clouds index. In *Proceedings of the 8th International Conference on GeoComputation*, pages 551–562, 2005.