DELFT UNIVERSITY OF TECHNOLOGY

Using a database for dynamic point cloud data management

GRADUATION PLAN

Stella Psomadaki
4412052

Main mentor:
Drs. T.P.M. Tijssen,
Second mentor:
Prof.dr.ir. P.J.M. Oosterom,
External supervisor:
Fedor Baart (Deltares)

January 2016

Table of Contents

1 Introduction

Point cloud usage has seen a rapid growth over the last years (Kohli, 2015). This is mainly the result of the developments in the point cloud acquisition technologies; namely terrestrial and airborne laser scanning, mobile mapping, image matching and multi-beam echo-sound techniques etc. In addition to that, recent advances have produced many low-cost and generally easy-to-use devices and techniques, such as the Microsoft Kinect[1], Google's Project Tango[2], depth cameras and Structure from Motion combined with Unmanned Aerial Systems (UAS) which are able to acquire billions point clouds in a short period of time (Rusu and Cousins, 2011; Westoby et al., 2012; Khoshelham, 2011). These advances have allowed repeated scans of the same area on a frequent basis, resulting in even more available spatial information and the addition of the fourth dimension; the temporal component. This plethora of available data has increased the usage of point clouds in solving real world problems. More specifically, point clouds have many applications, like 3D urban modelling (Haala and Kada, 2010), indoor modelling (Previtali et al., 2014), flood modelling (Abdullah et al., 2009), line of sight analysis (Peters et al., 2015), forest mapping (White et al., 2013) and many other.

1.1 Problem statement

Although technological developments have made it feasible to acquire a huge amount of information, the management and storage of those massive point clouds is even today a very challenging topic (van Oosterom et al., 2015a). In the majority of today's applications file-centric approaches are chosen, meaning that point cloud data are stored as a collection of files. A typical workflow includes using desktop applications or command line executables like Rapidlasso's LAStools (Isenburg, 2012), reading one or more files, performing a processing and writing files back to the user. In the case of LAStools all this functionality is integrated with a project management and quality control framework making it a stand-alone application (Hug et al., 2004).

Despite the fact that file-based systems are continuously improving (Otepka et al., 2012), with the increasing amount of new point clouds present deficiencies especially in file formats allowed and absence of multi-user functionalities. The situation becomes even more complicated when taking into account the multiple dimensions of point clouds, as these are far from fixed. This means that depending on the acquisition method point clouds contain apart from X, Y and Z, also, time information, intensity, return number, number of returns, classification, colour etc. (Ramsey, 2014). Those attributes can be present in any order and combination (ASPRS, 2011) making simple selections in those dimensions cumbersome even for the most robust applications.

Database Management Systems (DBMS) have for many decades already been dealing with the data storage, indexing techniques, scalability and availability requirements needed by the majority of spatial and non-spatial applications. DBMSs provide concurrency control, transactions characterised by atomicity and isolation properties, security and version control. The database community, commercial and open-source, identifying the need for point cloud data management already provides native point cloud support. Both Oracle (Spatial and Graph) and PostgreSQL (PostGIS) use a rather similar approach for point cloud storage. Their model is based on the physical re-organisation of the point data into blocks that groups spatially close points (Ravada et al., 2010; Ramsey, 2014). In addition to that, other approaches have emerged for point cloud management.

Although the current point cloud management solutions offered in by the modern database

[1]http://www.xbox.com/en-US/xbox-360/accessories/kinect

[2]https://www.google.com/atap/project-tango/

systems perform well for specific cases, they have certain limitations. First, it is not well documented which approach best fits specific applications. Second, they consider point clouds as rather static objects not taking into account the time dimension. This comes in contrast with the growing number of point clouds continuously generated from low-cost sensors. In certain queries, even, the time dimension becomes as equally selective as the planimetric coordinates. This implies that storing time as an attribute no longer provides efficient searching, meaning that the query plan has to first select based on space and then on time. In addition to that, the current data organisation techniques do not scale good with continuous insertions. This hinders further development of applications that keep track of changes in an area using point clouds. An application where this applies is in the cases of coastal and river monitoring where the querying of time and space is highly correlated. Therefore, spatio-temporal data management structures are needed to fully exploit point clouds.

To be able to formulate the research question properly, an overview of the methods available to handle point clouds either spatially or temporally and spatio-temporally is given in the next chapter.

1.2 Scientific relevance

In many of the current point cloud applications, time plays a secondary role or no role at all. This means that time is either stored as an attribute within the file system or database, or is only present within the file or table name. Therefore, time does not directly influence the organisation of the data. Although time and space have been studied as separate entities, their integration is the one that will eventually describe the full reality captured by point clouds and enable spatio-temporal analysis and operations. This research aims to provide a solution for the integration of space and time by realising a 3D (2D space and time) point cloud data model using a database management system. Researchers that will benefit from this integration of space and time for example are, van Oosterom et al. (2015a), Bruens et al. (2012) and de Boer and Baart (2015).

The social benefit of providing integrated spatio-temporal functionalities lies in use cases like coastal monitoring and flood safety. By developing a way of managing space and time together, very useful information can be extracted which can create new knowledge and understanding of the physical processes around us.

2 Research framework

2.1 Point clouds in DBMS approach

Point clouds are often huge in size. In the majority of the current applications point clouds are being handled using files. This has lead to the creation of many proprietary or open file formats, with the LAS format (ASPRS, 2011) being the most widely used. Files have several advantages over database systems; specifically, available software and libraries that can be used for processing, ease of use, compression schemes etc. Since these file-based solutions can offer efficient accessing and processing of point cloud data, the question of whether a database can provide better management arises.

One very common advantage of utilising a file system is the trivial process of loading of the data. Database systems, on the other hand, utilise less trivial processes for the loading of the data, but as the size of the data increases file systems tend to require similar amount of time during this step (van Oosterom et al., 2015b). DBMS, also, tend to have difficult installations and configurations, contrary to files that can be managed by general purpose programming languages and available software. Both solutions, however, when operating on licensed software suffer from the cost of purchasing licenses.

Although some of the practical problems before are handled better with file-based systems, several disadvantages can, also, be identified. With the increasing low-cost acquisition possibilities, available point cloud information for a specific area can increase dramatically. This means that updates (inserts or deletes) take place more often these days, which for a file-system organisation means continuous documentation of metadata.

Additionally, querying massive data (spatio-temporal or not) is problematic in the file-based approaches. Firstly, the existence of many files makes the simplest selection a very time-consuming and expensive process. In fact, the selection algorithm has to perform a scanning of all the header information for each file. In van Oosterom et al. (2015a) the authors had to use a hybrid solution; a combination of LAStools and PostgreSQL with the file metadata, to optimise query response times. Secondly, file solutions do not support ad hoc queries. The use of a declarative language like SQL is thus required. Rather than reinventing the wheel and developing a new language to support queries, reusing DBMS solutions is more straightforward.

Finally, databases offer benefits like multi-user access, scalability and easier integration with other spatial data (vector or raster). These types of functionalities require (re-)development for file-based solutions (van Oosterom et al., 2015a). Again, instead of reinventing the wheel, DBMS can offer native support. However, all this additional functionality in the DBMS causes some overhead creating slower response times and more difficulty in their usage. The overhead can, however, be decreased with a good organisation of the data.

2.2 Spatial access methods

Spatial data, including point clouds, are often large in volume, dynamic and multidimensional objects which makes their management very difficult. This complexity, however, should not come at the cost of inefficient queries. For this reason, for many decades now spatial access methods have been a very important topic in the research community. The term spatial access methods refers to both spatial indexing and clustering techniques (van Oosterom, 1999). Since the main object managed is point clouds, only spatial access methods relevant to points are described.

2.2.1 Spatial Indexing

The purpose of spatial indexing, or in general indexing, is to support efficient data retrieval. In other words, indexes are important otherwise all rows within the table must be scanned during querying. The most well-known one-dimensional index is the `B-Tree` invented by Bayer and McCreight (2002). B-Trees are, however, not suitable for spatial objects which have at least two dimensions. Many indexing techniques have been developed during the years (Gaede and Günther, 1998) the most important being the R-Tree where a grouping of spatially close objects takes place and spatial objects are represented with their minimum bounding rectangle.

2.2.2 Space filling curves

One of the characteristics of spatial data, and specifically point clouds, is that there is no straightforward way to preserve the spatial proximity of the points in the storage medium. In other words, it would be ideal to store data close to each other in reality, also, close to the storage medium (e.g. hard disk). This is very significant as computer memory is 1-dimensional in contrast to point clouds that are multidimensional. In order to follow such an approach a primary key value that preserves locality is needed. A mapping from a higher dimensional space to one dimensional can be handled using space filling curves. The advantages of space filling curves for the indexing of multi-dimensional data are discussed in Lawder and King (2000). An advantage of this method is that scalable one-dimensional access methods (like a B-Tree) can be applied to the multidimensional point data.

Many space filling curves have been developed each one of them preserving a different degree of proximity in the data. An overview of space filling curves can be found in (van Oosterom, 1999). In this case, only two of them are discussed: the Z-order and Hilbert curve.

- Z-order: The `Z-order` or `N order` or `Morton` curve (Peano and Kennedy, 1973) maps the coordinates of points to one dimension by interleaving bits from each coordinate in a recurrent way. For example, assuming a point with coordinates (9,12) its binary representation is (1001, 1100). By interleaving these bits we get the following binary number 11100001 which represents number 225. This number is the morton key of point (9,12). A graphic representation of this bitwise interleaving is shown in Figure 1.

- Hilbert: The `Hilbert` curve (Hilbert, 1891) maps multidimensional data into one dimension by following a recursive procedure. For the generation in 2D, first, the square domain is subdivided into four. Then for each one of these sub-squares, the first step is repeated and the previous version of the curve is rotated or reflected such that the pieces are connected to each other, see Figure 2. Similar concept is followed in higher dimensions. In 3D, for example, a cube is recursively subdivided and the curves are connected if they share a face.

From the above description it is obvious that the Z-order curve is generally easier to be constructed in comparison to the Hilbert curve and more easily extended in higher dimensions. The ease of the construction, however, comes at the cost of preserving less proximity compared to the Hilbert curve. This characteristic is apparent by the presence of “jumps” in the Z-order curve. Faloutsos and Roseman (1989) showed that the Hilbert curve provides better results for range and nearest neighbour queries.

2.3 Spatio-temporal DBMS

It is a fact that the world around us is subjected to discrete or continuous changes. The same accounts for the measurements related to dynamic phenomena. The management of those

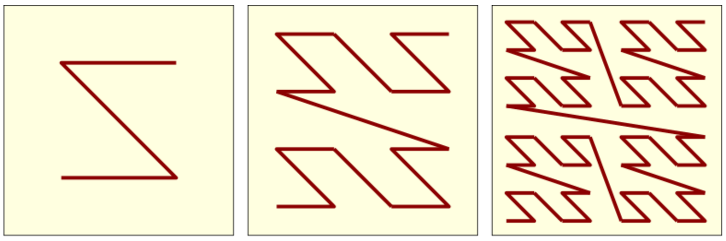


Figure 1: Z order curve organisation

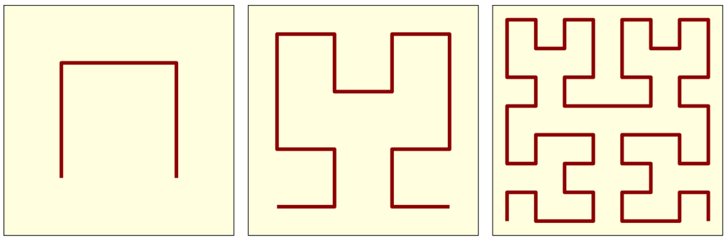


Figure 2: Hilbert curve organisation

measurements can take place in spatio-temporal databases which aim to combine the spatial and temporal features of the data. The first steps towards the integration of space and time started with the separate research on temporal (Clifford et al., 1993) and spatial databases (Güting, 1994). The main temporal data types found in DBMS are: instant, interval and period (Snodgrass, 1999). Respectively, they represent a moment in time, the length of time and time duration. In the `SQL92` standard the temporal aspect can take the form of `TIMESTAMP`, `DATE`, `TIME` and `INTERVAL`. For the modelling of space the Open Geospatial Consortium (OGC) and International Organization for Standardization (ISO) have established the standards for the management of spatial objects. These efforts resulted in the Simple Features Specifications for the Structured Query Language (Herring, 2006), where the geometry types, relationships and other operations are defined.

The integration of spatial and temporal datatypes is, however, not trivial. One of the challenges faced with spatio-temporal databases (STDBMS) is the different nature and semantics of the two concepts that need to be taken into account. The data models that have emerged through the years vary, among others, in two aspects: (a) the temporal resolution, meaning the way of partitioning the line of time, and (b) the granularity of time, which denotes in which level of the spatial data the time dimension is added, i.e. to the whole dataset or the attribute level. Another challenge in the spatio-temporal research is the efficient querying both in time and space. As a result, efficient indexing techniques and access methods (Theodoridis et al., 1998) are required. An overview of spatio-temporal indexing techniques can be found in Mokbel et al. (2003) and Nguyen-Dinh et al. (2010). Some implementations treat the time dimension as an additional dimension on top of the spatial component, others store time in the nodes of the index as an attribute and other implementations are using overlapping index structures to show the objects at different time instances (Theodoridis et al., 1998). Finally, STDBMS tend to have a dynamic nature, meaning many updates (deletions and insertions) take place constantly. A data structure and indexing scheme that handles this dynamic behaviour without degrading over time and as the database grows is therefore needed.

Spatio-temporal models present in the literature focus on various application domains and are thus different in how they handle time. One of the simplest model is the "snapshot model" by Langran and Chrisman (1988). The modelling of space takes place in time layers, in other words, it represents the state of the objects at different times (Figure 3). The different time layers are independent from one another. The drawbacks of this model are duplication of the data, difficulty in detecting the changes from one moment to the next one and difficulty in applying integrity rules (Langran, 1992). Other spatio-temporal models are (Pelekis et al., 2004): (a) based on timestamping at the creation and termination of the object, (b) based on managing the events that lead to changes, (c) representing space, time and semantics separately, (d) based on moving objects, etc.

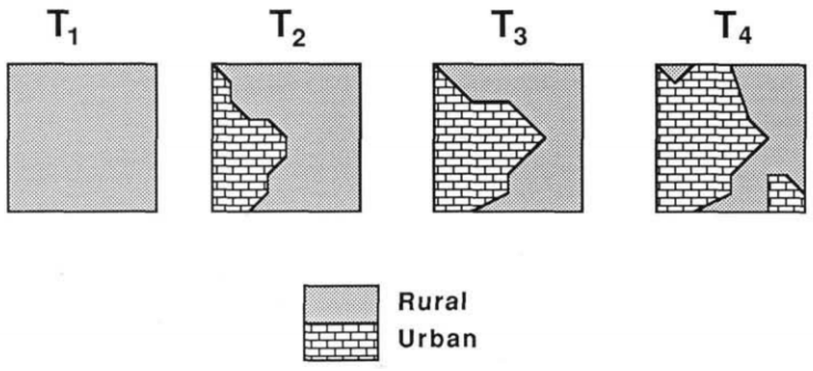


Figure 3: Time slices of an urbanisation phenomenon
Source: Langran (1992)

2.4 Relevant point cloud data management aspects

Although point cloud availability is increasing rapidly the last years, the full richness of point clouds is in general underutilised. What makes point clouds a challenging source of data is their generally unstructured nature and multidimensionality of their properties. Whatever management medium is used for organising massive point cloud data, van Oosterom et al. (2015a) present the relevant aspects that should be be taken into account during the design phase:

- The storage of the X, Y, Z coordinates and support for Coordinate Reference Systems.
- The attributes attached to the points like, intensity, return number, number of returns, classification, colour (Red, Green, Blue) in any order and combination.
- The spatial organisation of the points including efficient blocking, clustering (space filling curves) and indexing techniques in the multidimensional space. A parameter relevant to the clustering aspect is the choice of dimensions used for the key calculation, for example 2D (X, Y) or 3D (X, Y, Z).
- The time component and its importance in the organisation of the point cloud. Time can either have no role in the organisation (stored as an attribute), or play a crucial role in the organisation by taking place in the key calculation of the space filling curve in the form (X, Y, time) or (X, Y, Z, time).
- Compression techniques in order to reduce storage and enable dissemination via wireless networks.
- Level of Detail (LoD) strategies in order to significantly reduce the amount of data that are sent from the server to the client and to support efficient computations and visualisations. The latter involves decreasing the complexity of the scene as the viewer moves away. In other words, the further the viewer from an area, the lesser the points presented (Figure 4). Two types of approaches can be identified for LoD structuring: the

multi-scale and the vario-scale (or continuous) approach. In the former, the levels are predefined while in the latter the level of detail is another dimension, called importance. In contrast to the discrete levels, the importance value can be of floating point data type. This extra dimension can then be used as a parameter for the organisation itself by sorting and clustering using space filling curves e.g. Hilbert or Morton curve. The spatial clustering could then be in the form (X, Y, time, importance) or (X, Y, Z, time, importance)

- Operations like loading, selecting, and algorithms that operate on point clouds e.g. normal estimation, nearest neighbour finding etc.
- Parallel processing techniques in order to fully exploit the computing capabilities of the modern computers. Parallel processing should, in general, offer better performance than a single process.

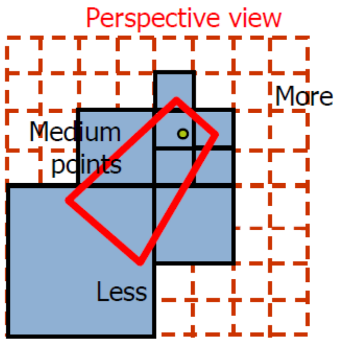


Figure 4: Perspective query returning blocks from multiple LoDs
Source: van Oosterom et al. (2015a)

2.5 Existing point cloud implementations in DBMS

The management of point clouds in DBMSs has been in the centre of research for many years already. According to Dobos et al. (2014) some basic requirements that a relational point cloud database should hold are: spatial and non-spatial filtering, primary key searches, nearest neighbour determination, interactive visualisations and efficient loading, insertions and deletions that are supported by efficient indexing schemes. Richter and Döllner (2014) also driven by the increasing availability of point cloud data describe the architecture of a system that deals with data coming from varying sensors and devices, handling also database updates. They propose an incremental database update process in order to avoid storing unchanged parts between different time stamps. To achieve this, change detection techniques need to take place before the data are loaded into the database. The data structure used in their system is an octree.

Zlatanova (2006) argues that in theory point clouds can be either stored with already existing data types, `POINT` or `MULTIPOINT`, or with user defined types. Storing point clouds using `POINT` data types was, also, proposed in Höfle et al. (2006). To speedup time queries, the authors introduced a secondary index on the time attribute. Ott (2012) and Ramsey (2014), however, explain that the previous implementation has serious drawbacks, the most important being the separation of the point geometry and the attributes in the `MULTIPOINT` approach, the amount of rows in the `POINT` approach and the memory intensive operations needed for both of them.

Currently, the database community provides several approaches for point cloud management. In their majority two storage models can be distinguished. The first model is based upon the organisation of points in `blocks`, meaning in groups of spatially close points. The second organisation is the `flat` table model, where each point is stored separately in one row. In this section the existing implementations in several systems are presented shortly. In all of the following systems, the time dimension can be considered only as an attribute and is not part of the main organisation.

2.5.1 Oracle

The options for storing point clouds in the Oracle database are three. The first option involves the blocked model, the second using a flat table (normal relational table) to store the points and finally, the hybrid model.

For blocked organisation, Oracle Spatial and Graph provides the `SDO_PC` and `SDO_PC_BLK` data types which represent the logical object and physical storage respectively. `SDO_PC` stores the metadata information of a point cloud like its name, the spatial extent, the number of dimensions (spatial and non spatial), the resolution, the name of the `SDO_PC_BLK` table that contains blocks, the parameters for partitioning etc. The `SDO_PC_BLK` is the physical storage table with the block information, that includes the spatial extent of the block, the resolution, the number of points, the `Binary Large Objects` (BLOB) that include the points, etc. (Oracle, 2015a). When creating the blocks two blocking methods can be used (Oracle, 2015a): the R-Tree which groups spatially close points, the Hilbert R-Tree which groups points that are closer in the Hilbert curve or using an external blocking scheme like the one offered in Point Data Abstraction Library (PDAL)[3].

The second option for storing point clouds in an Oracle database is by using a flat table approach with database specific hardware like the Oracle Exadata Database machine (Oracle, 2015b). This hardware is designed to offer maximum performance for running the Oracle Database. Its performance for point cloud management was tested in van Oosterom et al. (2015a).

Finally, point clouds can be stored by using a `Hybrid` solution. This solution involves using an `Index Organised Table` (IOT) with a key based on a space filling curve (Godfrind and Horhammer, 2015). The former concerns a table where the data are stored in a B-Tree structure. Normally, a query would have to traverse the index table in order to find the requested value and after that return the actual data by visiting the table where they are stored. In an IOT, the second step is not needed since the rest of the columns are stored together with the index. The key is based on the Oracle implementation of the Hilbert curve. So far the key can be calculated only in 2D using the function `sdo_pc_pkg.hilbert_xy2d` or `sdo_pc_pkg.generate_hilbert_vals`. This organisation model can be considered as a combination of the advantages of the previous two.

2.5.2 PostgreSQL

PostgreSQL provides point cloud support through the `pgpointcloud` extension developed by Ramsey (2014). The idea behind the development was that points should not be stored as PostGIS `POINT`s per row, but rather organised into patches (the equivalent of blocks in Oracle). For this reason, the `PC_Point` and `PC_Patch` data types were developed. Since the data are packed into byte arrays, a description of how this packing is performed is described by an `XML` schema document (Ramsey, 2014). This document is used to take into account the multidimensionality of point clouds. The actual loading of the point cloud into the database can take place either by making use of well-known binary (WKB) objects or with the PDAL driver for pgpointcloud. The points and the patches are tied together using a common identifier. This

[3]http://www.pdal.io/

organisation is extended in Cura et al. (2015), where the authors describe a complete point cloud management system supporting metadata, compression, filtering, fast loading and processing, as well as integration with vectors and rasters .

Space filling curve approaches are not available in the current implementation of the point cloud support of PostgreSQL. The only available spatial clustering functionality in PostGIS[4] is the GeoHash[5] function that is based on the Z-order curve.

2.5.3 MonetDB

The flat table approach is followed within MonetDB by Martinez-Rubi et al. (2014, 2015); Alvanaki et al. (2015). MonetDB is a column store database meaning that the X, Y, Z data and their attributes are stored in different columns. The characteristic that makes it comparable with the previous approaches is the "a novel and powerful, yet lightweight, cache conscious secondary index" (Sidirourgos and Kersten, 2013) that is used as rough filtering step during querying.

2.5.4 Other point cloud data management implementations

Apart from the previous straightforward ways to store and organise the point cloud information in a DBMS, also other ways of management can be found in the literature. In van Oosterom et al. (2015a) the authors combined LAStools together with PostgreSQL. To implement their solution, first they sorted and indexed the files using the corresponding command line executables (lassort and lasindex). Then, the extents of each file were stored into PostgreSQL. In this way, when a query is performed the database is scanned first to locate the files that overlap with the query area. The files that satisfy the previous query are then used as input to LAStools to further refine the selection.

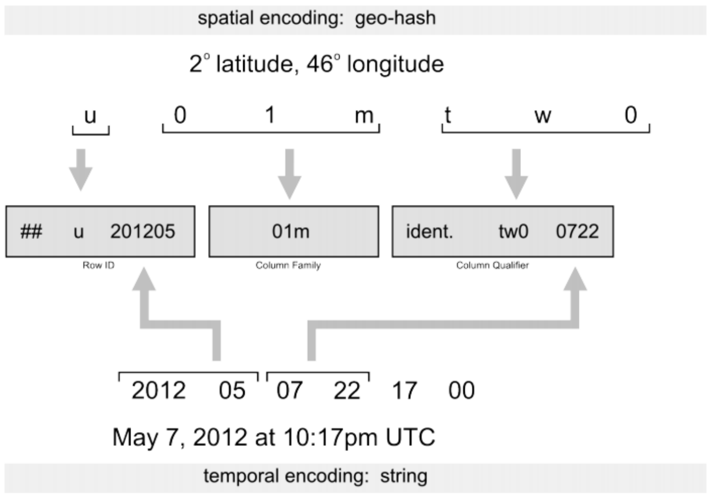

spatial encoding: geo-hash
2° latitude, 46° longitude
u 0 1 m t w 0
## u 201205
Row ID
01m
Column Family
ident. tw0 0722
Column Qualifier
2012 05 07 22 17 00
May 7, 2012 at 10:17pm UTC
temporal encoding: string

Figure 5: Encoding of the spatio-temporal index in Fox et al. (2013)

Another way that stays away from Relational DBMS are the NoSQL (Not Only SQL) databases that emerged with the development of the Internet. Those databases stay away from the tabular way of organising data and the use of SQL. There are four types of NoSQL databases: key-value stores, document, column and graph databases. Boehm and Liu (2015)

[4]http://postgis.net/

[5]https://en.wikipedia.org/wiki/Geohash

used a document based `NoSQL` database, MongoDB, to store LIDAR point clouds. The documents were used to store the metadata for each file, while the actual files were organised in a distributed file system (GridFS). A similar way is proposed in Wang and Hu (2014) for the same platform.

Spatio-temporal point cloud implementations are present in the literature, although limited in number. A NoSQL spatio-temporal implementation using the Apache Accumulo software is presented in Fox et al. (2013). The spatio-temporal index structure developed interleaves the parts geohash of the `(x,y)` point with parts of the string representation of the date (see figure 5) to accommodate the key structure of the Accumulo system. The data type of the key is a string and it is indexed lexicographically. The results show that the spatial and time complexity of the queries increases the query response times.

Another spatio-temporal implementation closely related to point clouds can be found in Tian et al. (2015) where the authors apply a space filling curve (Morton code) to efficiently organise 3D points `(x,y,t)` in a 3D R-tree which is stored in a relational database. Their prototype was compared with a spatial database and their organisation gave faster queries and scalability with concurrent queries.

3 Research objectives

The research framework presented before tried to describe all the relevant aspects that have an influence on the management of point cloud data (section 2.4). In addition to that, the current implementations were presented shortly in section 2.5 and their limitations in handling temporal (or dynamic) point clouds became obvious due to the limited research on the area.

Although all the relevant aspects in the previous list can play an important role in the management of dynamic point cloud data, the focus of this research will only be on the integration of space and time and their organisation using a space filling curve approach. This is mandated by the use cases (section 4.1) where the space and time components are highly correlated. Therefore, aspects like Level of Detail, and parallel processing are of less priority for this project. Their influence on the management of dynamic point cloud data could be part of a future work.

3.1 Research question

The main research question of this thesis is:

How can a space filling curve (Morton curve) be used for integrating the space and time components of point clouds in order to support efficient additions and queries in a Database Management System (DBMS)?

The aim of this thesis is to research the potential and apply a methodology that offers efficient data handling and storage of dynamic point clouds. In order to answer the main question, the following sub-questions are relevant:

- What are dynamic point clouds and what are relevant use cases and requirements for their querying?
- What types of data structures can be used to support the integration of space and time, taking into account the continuous insertions of new points and efficient querying?
- Is the complete integration of space and time with a space filling curve approach efficient, or is the less deep integration of space and time performing better?
- What is a suitable scaling of the various dimensions depending on the use cases?
- What is a suitable resolution of the dimensions to be used for the space filling code calculation?
- How does the suggested method compare to the current way of working?

3.2 Scope

The focus of this research will be on the efficient handling and storage of dynamic point clouds as defined by the use cases which support coastal monitoring applications. The methodology suggested will focus on using space filling curves for the structuring of space and time. Only if necessary different approaches will be investigated. Because of time constraints the focus will be on on the integration of the x, y, t dimensions. This decision is based upon the fact that x,y coordinates are, in general, more selective than the z dimension in the use cases. Same applies for time (more selective than z). Finally, comparing the efficiency of different space filling curves is not of high priority for this research.

4 Methodology

This part of the research proposal aims to provide the general strategy that will be followed in order to answer the research question defined in the previous section. For this reason, the research will follow three general steps (figure 6). The first step includes the conceptual modelling of the problem, which deals with the structuring of space and time. The second step is implementing the conceptual model in order to, as a final step, validate whether it fits a specific use case. Those three steps can be put into one methodology and create a schema like the one shown in figure 7. As can be seen, the most relevant aspects towards the solution are: identifying the use cases, identifying the structure of space and time that best suits the use case, initialising the database and loading the data, inserting more data (dynamic nature) and performing queries. Therefore, the main aim for each different use case is to observe how the database system behaves when applying a different structure of space and time. The output of the methodology proposed will yield comparisons that are relevant to answer the research question.

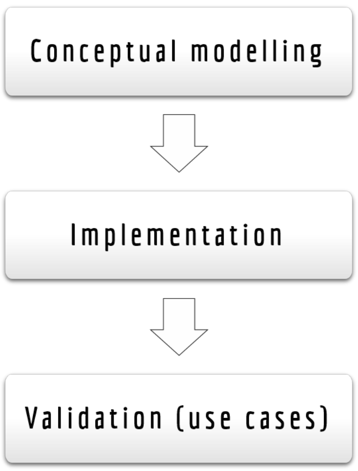


Figure 6: The general workflow of the research

4.1 Use cases

The Netherlands is a country situated at the North Sea and its southwestern part is the delta of the three large rivers: Rhine, Meuse and the Scheldt. Over the years, protection of the coast has played a very important role for the development of the country, since almost one quarter of the land lies below sea level. Typical flood protection structures include dams, dikes and dunes.

The Dutch coastal policy was introduced in 1990. There it was stated that the country should *"Hold the line"* (Sistermans and Nieuwenhuis, 2004), meaning that the coastline should be prevented from moving towards the mainland. For this reason, the Ministry of Transport, Public Works, and Water Management (Rijkswaterstaat) as part of the coastal monitoring guidelines performs a yearly survey of the Dutch coast in order to determine changes in coastal elevations. These surveys are a combination of depth surveys from echo sounding equipments mounted on vessels and height surveys coming from Laser altimetry technology (Pot, 2011). A more detailed description of the data is provided in section 5. Deltares, an independent institute for applied research in the field of water and subsurface, is a user of the data.

Another relevant use case is the "Zandmotor" (Sand Engine) (Mulder and Stive, 2011), 21

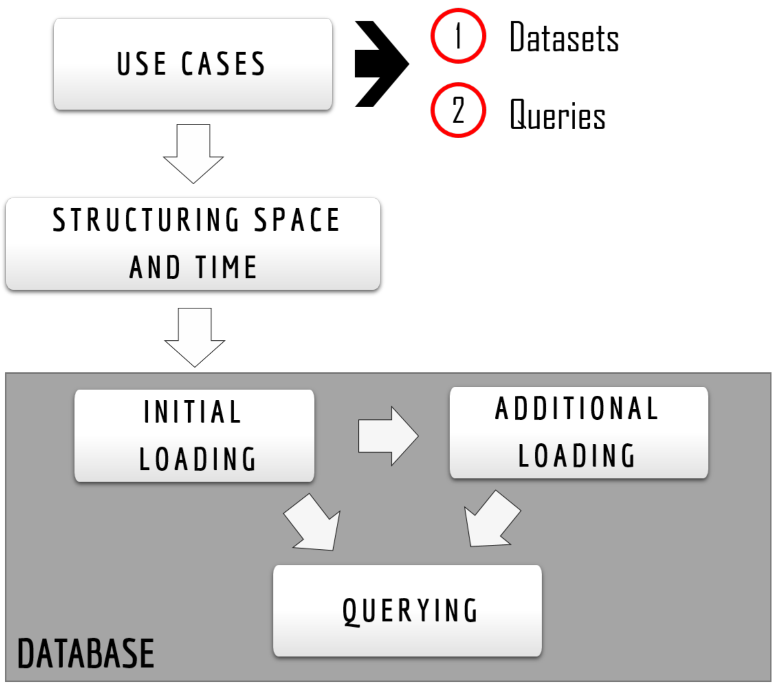


Figure 7: The general methodology schema to be used

million cubic metres of sand deposited between Ter Heijde and Kijkduin at the province of South Holland. This amount of sand has created a man-made peninsula. The purpose of this pilot program is to investigate how nature spreads this amount of sand along the coast as the years go by. In order to see if the experiment is going as thought, daily monitoring of the area takes place. A more detailed description of the data is provided in section 5.

In general, both use cases identified have three components in common: they belong to the general use case of monitoring changes, they include specific datasets of growing point clouds whose spatio-temporal management is very important and specific queries to study the physical process they represent. More specifically, the queries posed are used for the calculation of several coastal indicators (Giardino et al., 2014), the detection of new features, changes in the volume and erosions or depositions etc. Those processes are directly related to time, although the space and time responses significantly differ from process to process, as shown in figure 8.

The current way of working includes using multidimensional storage, like `netcdf` and `hdf5`, by transforming the original point clouds into grids. Although this transformation enables spatial operations, the original richness of the point cloud is lost. Therefore, the user requires the preservation of the original dataset supported by spatiotemporal management options.

The user requirements for the querying of dynamic point cloud DBMS are:

- Space (`x,y[,z]`) and time ranges.
- Selection of points based on a line (with/without buffer zone) in certain time ranges.
- Minimum, maximum, average heights from range queries in space and time.
- Selection based on attributes, for example on classification (less important).
- Selection based only on time or only in space (less important).

Those queries are then used to create animations or time series analysis for monitoring volume and growth (Figure 9).

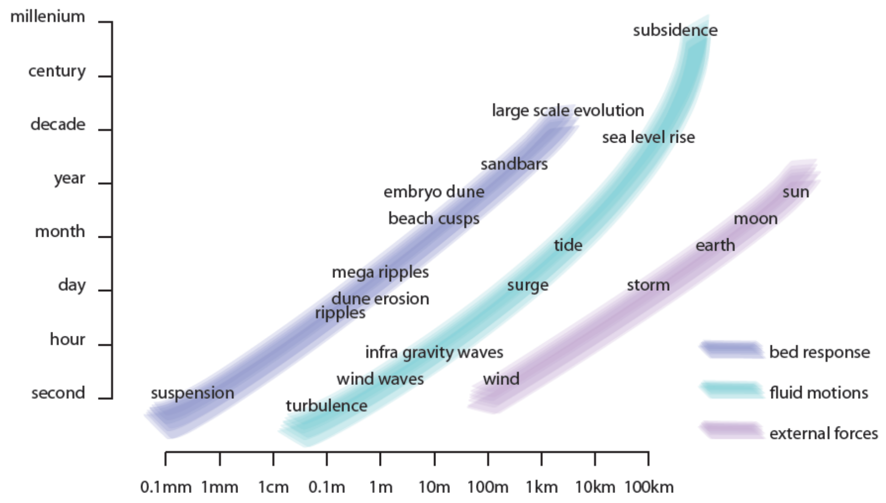


Figure 8: Time and spacial scales of bed responses in the coastal zone
Source: Baart (2013)

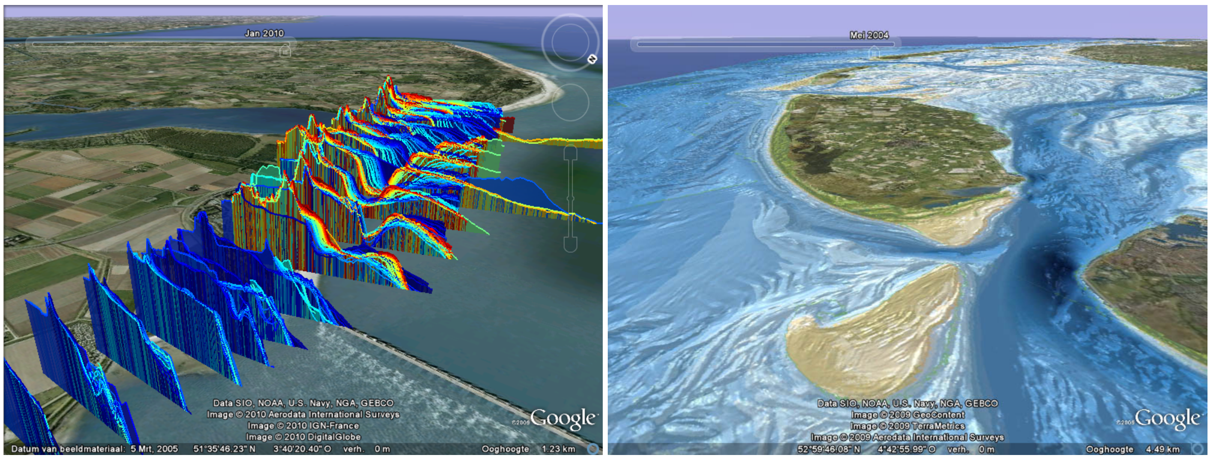


Figure 9: Animations and time series for coastal applications
Source: Bruens et al. (2012)

4.2 Structuring space and time

Structuring space and time to support dynamic point clouds is not a trivial problem. The reason for this is that two contradictory requirements need to be taken into account; points in space and time should be stored very close (clustered) in the memory but in a way that the already organised data are not affected from the newly coming data. The contradiction takes place when adding new data. The new points, in order to preserve space-time locality, will have to be inserted between the already stored points. As a result, the data already organised in the memory will have to be shifted to new places which is a very costly operation. Therefore, a possible solution for the problem at hand can be an intermediate going from the very fine integration of space and time to a coarse integration, where time dominates over space.

These two approaches differ in the way they structure dynamic point clouds. In the following subsections the possibilities are discussed.

4.2.1 Deep integration of space and time

The complete integration of space (`x,y[,z]`) and time (`t`) implies that the same treatment is given to all three or four dimensions, both in the structuring and the querying. The complete integration of space and time can be achieved with the following data structure.

Structuring with space-filling curves. The motivation for using space filling curves is threefold. First, they lead to dimensionality reduction. Spatial objects of higher dimension are transformed to a single one, making it feasible to apply classical B-Tree indexing techniques. Second, space-filling transformations preserve the spatial locality of the points. There is, however, no ordering that preserves the total proximity and therefore, the application of different curves is relevant. Even so, points close in the n-dimensional space will also be stored close in the physical storage of the database. This will minimize the I/O cost for fetching the data of a query. Finally, research shows that multidimensional indexing is possible using space filling curves (Lawder, 2000).

Within this solution two alternatives can be identified. The first includes giving a space filling key to points of a specific granularity. This organisation can be applied in case it is required that the size of the key should hold only a certain number of bits and can be considered as a way of tiling (blocking). The second includes giving each point the full key. In this case, the storing of the space (`x,y[,z]`) and time (`t`) dimensions is redundant because they can be decoded from the key. This significantly reduces storage in the database (Martinez-Rubi et al., 2015) but adds extra processing time since the encoding and decoding of the keys have to take place.

Drawbacks of this solution become visible with the new insertions of point clouds. Since space filling curves are used, with every insertion of a new point cloud for the specific area of interest, the new points will have to be inserted between the already stored points in the B-Tree. This happens exactly because of the locality preserving characteristics of space filling curves. When the table is growing constantly, this reorganisation will have a negative influence on the efficiency of the system. The effect of this process is part of the research of this thesis.

4.2.2 Time domination

The organisation of space and time can also be achieved in a less integrated way. This solution could be exploited in case the continuous re-organisation of the data from the previous approach needs to be avoided to a certain extent. Time in this alternative is the one that first determines how the newly coming data will be stored. Space has then a secondary role. The less deep integration of space and time can be achieved with the following data structure.

Time with spatial clustering. In contrast to the previously mentioned space-filling method, time no longer takes place in the determination of the morton key. The morton key is only determined from the spatial components and time is concatenated with the spatial key in the following way: `time + key` (Figure 10). A similar time-partitioning approach of moving objects is used in Jensen et al. (2004). Assuming that the data are added in an ascending date order, the already organised data will not be affected. From a spatial perspective though this results that spatially close points but in consecutive years will be stored far away in the physical storage making queries for specific location over all times slow.

This organisation although supports the dynamic nature of the use cases poses a limitation in performing certain queries. For example, a query requesting for a certain location the heights of fifteen consecutive years will be performed relatively inefficiently, since a lot of rows within each year need to be traversed. Another limitation can be found when the dynamic nature of the use case is so fine that data are growing every minute. Then this solution will create a very large number of time layers which will be very difficult to query. Nonethe-

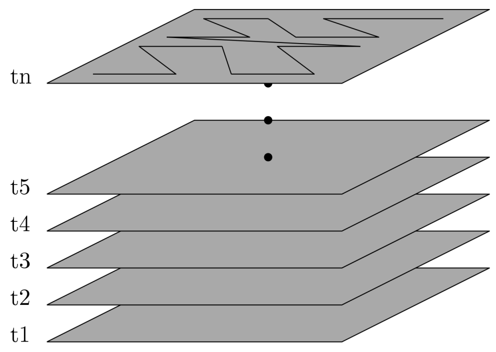


Figure 10: Organising space in time layers

less, the data will be clustered both in time and space and B-Tree indexing techniques can be applied to support efficient queries.

For both of the two possibilities introduced above the Hilbert or Morton curves are the most prominent space filling options to be used, since they are generally more suitable for multidimensional access methods (Abel and Mark, 1990; Gaede and Günther, 1998). Another possibility to investigate for both approaches is table partitioning; that is having an archiving table with the historic data (archive table) and a separate table for the newly coming one (current table). After the latter has gained a certain amount of new spatio-temporal information, at the end of the month as an example, only then these information is transfered to the archiving table. Therefore, a large amount of re-organisation happens only at few moments in time.

4.2.3 Dimensional scaling

Another important aspect that requires investigation is the relative scaling of the dimensions used. This is of great significance because of the inherently different nature of the spatial and temporal dimensions. The spatial units usually used are kilometers, meters, centimeters or millimeters. Time, on the other hand, is measured in years, months, days, hours, minutes or seconds. The correspondence of those two can be considered as the factor of how much time is integrated with how much space. This factor should be chosen according to the most prominent querying of the data in order to maximise the chance of performing minimum disk access.

4.3 Initial loading

For the loading of the datasets the platform independent process described in van Oosterom et al. (2015a) will be used. More specifically, the process is split into three parts starting with the initialisation of the database and table along with the enabling of all the needed functional extensions. As a second step, the data are prepared and then populated (or loaded) into the database. The loading can be in bulk or dynamically i.e. as the points are measured. In this step the calculation of the space-filling keys can occur. Finally, the indexes necessary are build. Depending on the system other intermediate steps might need to take place. These steps are, also, shown in a diagrammatic form in figure 11.

4.4 Additional loading

The purpose of this process is to populate the already loaded table with newly-coming data. This additional loading can resemble the loading step with the difference that the initialisation

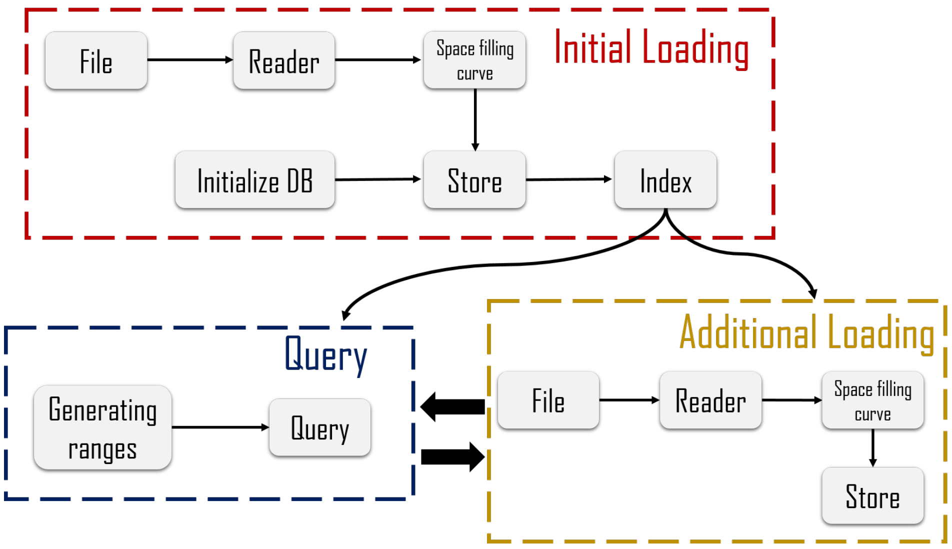


Figure 11: An overview of more refined steps for the processes of Initial loading, Querying and Additional loading

part in not repeated (see figure 11). In case of the table partitioning approach this step is differentiated in that two steps have to take place: first, the data are loaded into the current table and then, moved to the archive table at an appropriate time. In all cases, the system should take care of the updating of the index. Which structure of the previously chosen performs better during this step is part of the research of this thesis.

4.5 Querying

The goal of the querying process is to present the number of points and response times of queries corresponding to the user requirements. Querying in space and time will be the major component of this process. The actual queries are derived from the use cases and are mentioned briefly in section 4.1.

In order to perform spatial and time selections, the queries need to be modified so that the space filling organisation is taken into account. The reason behind this requirement is that the transformation into a linear space results in a loss of spatial relationships between objects. As defined by the user requirements, the majority of the queries performed are some kind of range query (Figure 11). A technique for mapping the space-filling values to the range queries needs to be identified. A general methodology includes, mapping the query to sets of continuous space filling curve traversals, visiting those sets in the B-Tree and identifying the relevant points and finally validating that the points actually overlap with the specific area. For the Morton space filling curve its relationship with the Quad-Tree (van Oosterom and Vijlbrief, 1996) or the Octree can be taken into advantage.

5 Datasets Used

The datasets that will be used for the validation of the methodology developed originate from the domain of coastal monitoring. Their nature is, however, different since they represent use cases with different space and time scales. This is relevant in order to evaluate how the proposed structures behave with data of different characteristics.

5.1 Coastline dataset

The first dataset is that of the coastline of the Netherlands (`Kustlidar` in dutch), provided by Rijkswaterstaat. The data is open data in point cloud format at least for the years 2014 - 2015. An approximation of the coverage of is shown in figure 12 and a view of the point cloud itself in figure 13.

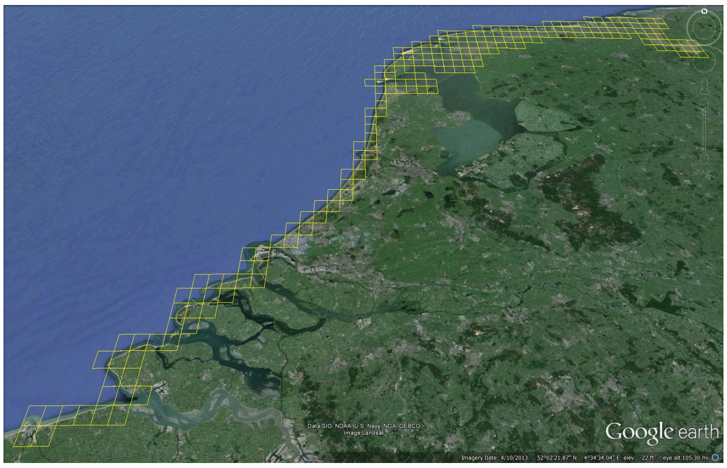


Figure 12: Overview of coastline dataset
Source: Deltares (2015)

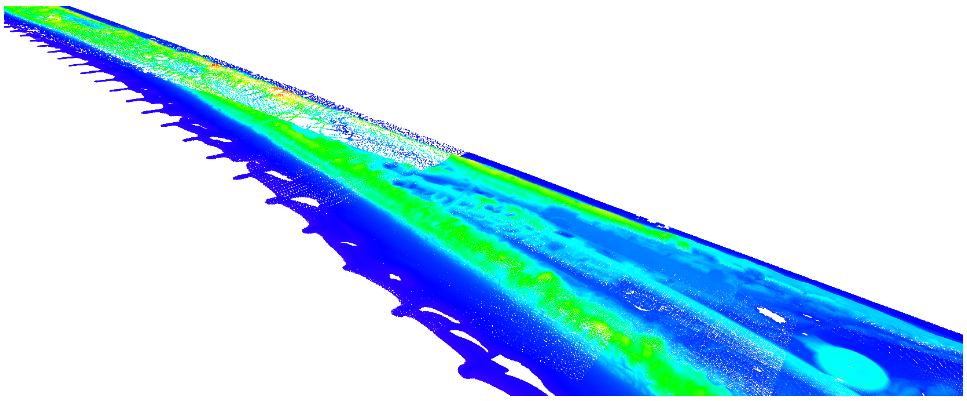


Figure 13: Overview of the coastline point cloud

The data has the following characteristics:

- 1 update per year
- The total point cloud is sectioned into tiles
- The files contain only the X, Y, Z information
- Time is only mentioned in the file names
- The point clouds of 2014 and 2015 are given as filtered and unfiltered (buildings, people, trees etc.). For the previous years only the filtered point clouds will be provided
- The format is LAZ

5.2 Zandmotor

The second set of point clouds comes from the Zandmotor use case (see figure 14 and 15). The acquisition takes place using jet skis and all-terrain vehicles and point clouds of the same area are available from 2011 till 2015. The data are property of Deltares but will become open data in February 2016. The Zandmotor area is measured at irregular periods after storms take place in the area. Therefore, although it is not measured every single day, the time resolution will be in days in order to offer the best organisation. This means that "gaps" will be present within the time dimension. Same as with the coastline dataset, the time information is only present at the file name.

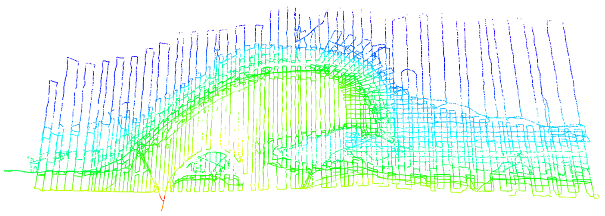


Figure 14: A zandmotor dataset of 2011

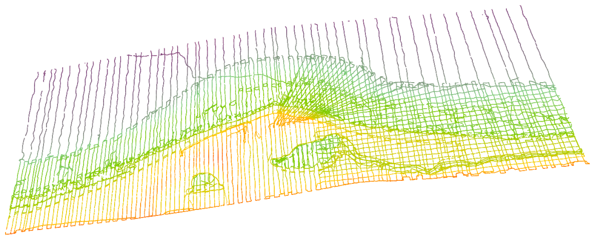


Figure 15: A zandmotor dataset of 2015

The two datasets presented are different in nature, size, number of files and spatial and time resolution. Their different nature poses different requirements in their organisation. For example, the coastline dataset will be updated only once per year but the amount of data is significantly more than the Zandmotor. On the other hand, the Zandmotor datasets are updated more frequently (every day) and therefore, the methodology developed should not make the database unavailable for a long period of time with every update. Hence, it is important to see how the data structures react to the differences. A comparison of the differerent characteristics of the datasets is provided in table 1.

Table 1: An overview of the different characteristics of the datasets used

Dataset	No. of files	No. of points per file	Time resolution	Spatial resolution
Zandmotor	33	100,000 – 200,000	Day	mm
Coastline	222	50,000 – 20,000,000	Year	cm

6 Experimental design

Based upon the user requirements and use cases, an experimental design is proposed. This design will be part of the validation step discussed in the methodology section and will assess how the proposed method is performing according to the use cases. Aspects relevant to this assessment are the storage space, the loading and querying response times. The benchmark will be implemented in three (or four) stages, corresponding to a growing number of point clouds.

Table 2: The description of the datasets used for the coastline use case

Benchmark	No. of files	No. of points	Disk size (MB)	No. of years
Small	4	25,000,000	31.7	2
Medium	15	120,000,000	130	2
Large	50	200,000,000	233	2

Table 3: The description of the datasets used for the Zandmotor use case

Benchmark	No. of files	No. of points	Disk size (MB)	No. of years	No. of days
Small	5	391299	7.5	1 (2011)	5
Medium	16	1332734	25.4	2 (2011 - 2012)	16
Large	22	1923746	36.6	3 (2011 - 2013)	22
Full	33	3147134	47.1	5 (2011 - 2015)	33

For the coastline use case, the details of the datasets used in every stage of the benchmark is given in table 2. This benchmark investigates how the structures proposed behave by increasing the spatial coverage, since only two time moments are available. The extents of the proposed benchmark are depicted in figure 16. Similarly for the point clouds coming from the Zandmotor use case the benchmark details are present in table 3. This benchmark will investigate how the structures behave with a wider time layer and the same spatial extenstion. All the benchmarks should be repeated for both structures of space and time, including their alternatives.

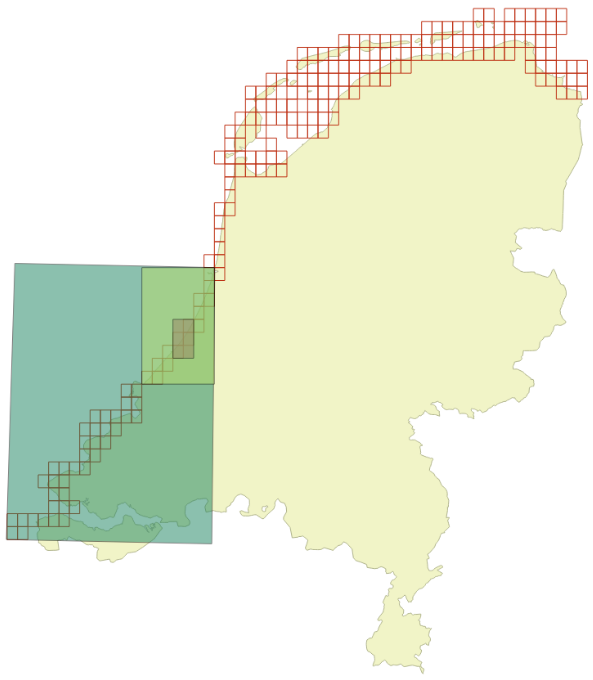


Figure 16: Overview of the coastline point cloud benchmark

7 Tools used

During the implementation of this thesis several tools will be utilised. For viewing of the results of the queries either the FME software will be used or the Potree 3D viewer[6]. The decision on the use of the DBMS to be used is based upon the information in table 4. As can be seen, the use of Oracle Spatial and Graph is obvious, although the system is commercially available through licences. The determining functionality that makes Oracle the most obvious choice instead of open source alternatives is the existence of Index Organised Tables. This available functionality has the potential of providing a very efficient organisation together with the use of space filling curves. Only if the system does not give the functionality needed, other databases like PostgreSQL will be investigated as a second option.

Table 4: Table of the requirements for the choice of the DBMS

Requirement	Oracle	PostgreSQL	SQL Server
Licence	Commercial	Open source	Commercial
Space Filling Curves	Encoding 2D Hilbert keys but not decoding	-	-
Index Organised Tables	Index organised table	static CLUSTER, index-only scans	clustered index
Blocking methods	R-Tree, Hilbert R-Tree	Using PDAL's chipper (outside the db)	-
Parallel querying	Yes	-	-

Python will be used when user-defined functionality is needed. This is the case for the calculation of the Morton keys, since the current systems do not fulfill this requirement. An implementation of the Morton key in 2D is provided in `https://github.com/NLeSC/pointcloud-benchmark/blob/master/python/pointcloud/morton.py`. In the full integration of space and time, however, the 3D keys are needed. The previous solution might have to be extended or a different solution will be identified. Python will, also, be utilised in order to connect to the database and perform the querying.

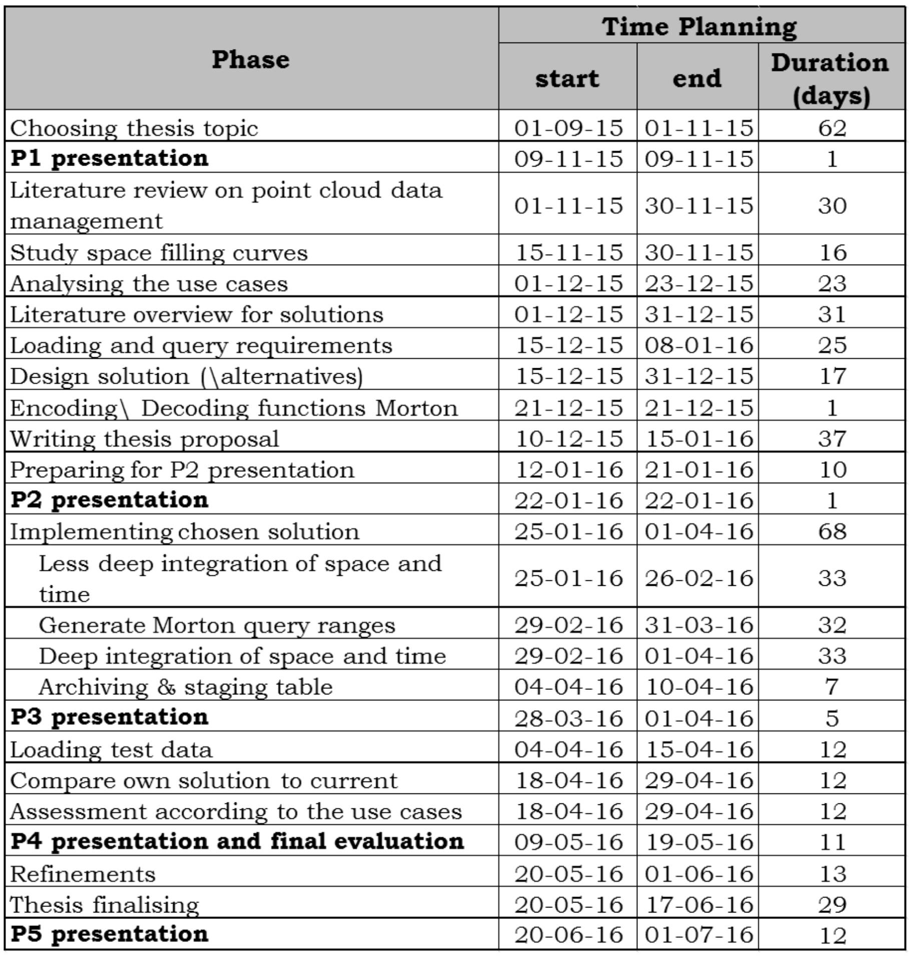
Finally, for the composition of the actual thesis document LaTeX and the unofficial template of Geomatics thesis available at `https://github.com/tudelftgeomatics/thesis_template` will be utilised. For making figures Rhino will be used in 3-dimensional sketching and Ipe for 2-dimensional sketching.

8 Activities & Schedule

The planning/ phasing according to the academic calendar is presented in figure 17. The most important dates and events are shown below. Those dates are taken from the academic calendar and the exact dates will be determined during the year.

Date	Event
2016-01-22	**P2 Presentation**
2016-03-01 – 2016-03-31	**P3 Presentation**
2016-05-09 – 2016-05-20	**P4 Presentation**
2016-06-20 – 2016-07-01	**P5 Presentation**

[6]http://potree.org/



Phase	Time Planning		
	start	end	Duration (days)
Choosing thesis topic	01-09-15	01-11-15	62
P1 presentation	09-11-15	09-11-15	1
Literature review on point cloud data management	01-11-15	30-11-15	30
Study space filling curves	15-11-15	30-11-15	16
Analysing the use cases	01-12-15	23-12-15	23
Literature overview for solutions	01-12-15	31-12-15	31
Loading and query requirements	15-12-15	08-01-16	25
Design solution (\alternatives)	15-12-15	31-12-15	17
Encoding\ Decoding functions Morton	21-12-15	21-12-15	1
Writing thesis proposal	10-12-15	15-01-16	37
Preparing for P2 presentation	12-01-16	21-01-16	10
P2 presentation	22-01-16	22-01-16	1
Implementing chosen solution	25-01-16	01-04-16	68
Less deep integration of space and time	25-01-16	26-02-16	33
Generate Morton query ranges	29-02-16	31-03-16	32
Deep integration of space and time	29-02-16	01-04-16	33
Archiving & staging table	04-04-16	10-04-16	7
P3 presentation	28-03-16	01-04-16	5
Loading test data	04-04-16	15-04-16	12
Compare own solution to current	18-04-16	29-04-16	12
Assessment according to the use cases	18-04-16	29-04-16	12
P4 presentation and final evaluation	09-05-16	19-05-16	11
Refinements	20-05-16	01-06-16	13
Thesis finalising	20-05-16	17-06-16	29
P5 presentation	20-06-16	01-07-16	12

Figure 17: The Gantt chart of the activities

9 Supervisors

The supervisors for this thesis project are as follows:

- Delft University of Technology
  - **Main mentor:** Drs. T.P.M. Tijssen
  - **Second mentor:** Prof.dr.ir. P.J.M. Oosterom
- Deltares
  - **External supervisor:** Fedor Baart

This thesis project is done in cooperation with Deltares. For this reason, Stella's work location will be at the Deltares offices in Delft. In addition to that, as shown above, an external supervisor is present at Deltares who will provide support for the use cases and technical matters. At TU Delft, Stella will receive support on the research issues from both supervisors, while more practical aspects will be discussed with T.P.M. Tijssen. Both TU Delft and Deltares will provide feedback during the progression of the thesis.

10 Meetings

Meetings every two weeks approximately are scheduled with the TU Delft supervisors. The meetings can, also, be in the form of emails if the schedule of the two supervisors is tight. In addition to that, meetings are scheduled with the Deltares supervisor for discussing certain aspects of the thesis. The meetings at Deltares will be more frequent after the Project Plan presentation on the 22nd of January 2016. Stella will, also, report frequently to all supervisors about the progress of the thesis.

11 Contact

For more details please contact:

Stella Psomadaki
Student number: 4412052
Telephone: +31 617397729
S.Psomadaki@student.tudelft.nl

References

Abdullah, A., Rahman, A., and Vojinovic, Z. (2009). LiDAR filtering algorithms for urban flood application: Review on current algorithms and filters test. *Laserscanning09*, 38:30–36.

Abel, D. J. and Mark, D. M. (1990). A comparative analysis of some two-dimensional orderings. *International Journal of Geographical Information System*, 4(1):21–31.

Alvanaki, F., Goncalves, R., Ivanova, M., Kersten, M., and Kyzirakos, K. (2015). GIS navigation boosted by column stores. *Proceedings of the VLDB Endowment*, 8(12):1956–1959.

ASPRS (2011). LAS specification, version 1.4 R13. Technical report, The American Society for Photogrammetry & Remote Sensing. The LAS specification.

Baart, F. (2013). *Confidence in coastal forecasts*. TU Delft, Delft University of Technology, Phd report.

Bayer, R. and McCreight, E. (2002). *Organization and maintenance of large ordered indexes*. Springer.

Boehm, J. and Liu, K. (2015). NoSQL for storage and retrieval of large lidar data collections. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1:577–582.

Bruens, A., de Boer, G., Ramaekers, G., and Mulder, J. (2012). The need for databases and viewers to support decision making and learning.

Clifford, J., Gadia, S., Jajodia, S., Segev, A., and Snodgrass, R. (1993). Temporal databases: theory, design and implementation. *Redwood City: Benjamin/Cummings*.

Cura, R., Perret, J., and Paparoditis, N. (2015). Point cloud server (PCS): Point clouds in-base management and processing. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1:531–539.

de Boer, W. and Baart, F. (2015). Point clouds in the delta. *Management of massive point cloud data: wet and dry (2)*.

Deltares (2015). Kusthoogte (kustlidar) bathymetry data in OpenEarth [url: `https://publicwiki.deltares.nl/display/OET/OpenEarth`].

Dobos, L., Csabai, I., Szalai-Gindl, J. M., Budavári, T., and Szalay, A. S. (2014). Point cloud databases. In *Proceedings of the 26th International Conference on Scientific and Statistical Database Management*, page 33. ACM.

Faloutsos, C. and Roseman, S. (1989). Fractals for secondary key retrieval. In *Proceedings of the eighth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 247–252. ACM.

Fox, A., Eichelberger, C., Hughes, J., and Lyon, S. (2013). Spatio-temporal indexing in non-relational distributed databases. In *Big Data, 2013 IEEE International Conference on*, pages 291–299. IEEE.

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

Giardino, A., Santinelli, G., and Vuik, V. (2014). Coastal state indicators to assess the morphological development of the holland coast due to natural and anthropogenic pressure factors. *Ocean & Coastal Management*, 87:93–101.

Godfrind, A. and Horhammer, M. (2015). Oracle support options for point clouds. In *Management of massive point cloud data: wet and dry (2)*.

Güting, R. H. (1994). An introduction to spatial database systems. *The VLDB Journal*, 3(4):357–399.

Haala, N. and Kada, M. (2010). An update on automatic 3D building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(6):570–580.

Herring, J. R. (2006). OpenGIS implementation specification for geographic information-simple feature access-part 2: SQL option. *Open Geospatial Consortium Inc*.

Hilbert, D. (1891). Ueber die stetige abbildung einer line auf ein flächenstück. *Mathematische Annalen*, 38(3):459–460.

Höfle, B., Rutzinger, M., Geist, T., and Stötter, J. (2006). Using airborne laser scanning data in urban data management-set up of a flexible information system with open source components. In *Proceedings of UDMS*, volume 2006, page 25th.

Hug, C., Krzystek, P., and Fuchs, W. (2004). Advanced LIDAR data processing with LAStools. In *XXth ISPRS Congress*, pages 12–23.

Isenburg, M. (2012). LAStools: Efficient tools for LiDAR processing. *Available at: http: http://www. cs. unc. edu/˜ isenburg/lastools/[Accessed October 9, 2012]*.

Jensen, C. S., Lin, D., and Ooi, B. C. (2004). Query and update efficient B+-tree based indexing of moving objects. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 768–779. VLDB Endowment.

Khoshelham, K. (2011). Accuracy analysis of kinect depth data. In *ISPRS workshop laser scanning*, volume 38, page W12.

Kohli, B. (2015). World LIDAR market - opportunities and forecasts, 2013 - 2020. Technical report, Allied Market Research.

Langran, G. (1992). Time in geographic information systems, technical issues in geographic information systems.

Langran, G. and Chrisman, N. R. (1988). A framework for temporal geographic information. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 25(3):1–14.

Lawder, J. K. (2000). Calculation of mappings between one and n-dimensional values using the hilbert space-filling curve. *School of Computer Science and Information Systems, Birkbeck College, University of London, London Research Report BBKCS-00-01 August*.

Lawder, J. K. and King, P. J. (2000). Using space-filling curves for multi-dimensional indexing. In *Advances in Databases*, pages 20–35. Springer.

Martinez-Rubi, O., Kersten, M., Goncalves, R., and Ivanova, M. (2014). A column-store meets the point clouds. *FOSS4G-Europe Academic Track*.

Martinez-Rubi, O., van Oosterom, P., Gonçalves, R., Tijssen, T., Ivanova, M., Kersten, M. L., and Alvanaki, F. (2015). Benchmarking and improving point cloud data management in MonetDB. *SIGSPATIAL Special*, 6(2):11–18.

Mokbel, M. F., Ghanem, T. M., and Aref, W. G. (2003). Spatio-temporal access methods. *IEEE Data Eng. Bull.*, 26(2):40–49.

Mulder, J. and Stive, M. J. (2011). Zandmotor (sand motor): building with nature. In *25th ICID European Regional Conference, Integrated Water Management for Multiple Land Use in Flat Coastal Areas, Groningen, The Netherlands, 16-20 May 2011. Paper III-21*. ICID.

Nguyen-Dinh, L.-V., Aref, W. G., and Mokbel, M. (2010). Spatio-temporal access methods: Part 2 (2003-2010).

Oracle (2015a). *Oracle Database, Online Documentation, 12c Release 1 (12.1): Spatial Data Types and Metadata / Point Cloud-Related Object Types*.

Oracle (2015b). *Oracle Exadata Database Machine X5-2*.

Otepka, J., Mandlburger, G., and Karel, W. (2012). The OPALS data manager—efficient data management for processing large airborne laser scanning projects. *Proceedings of the ISPRS Annals of the Photogrammetry, Melbourne, Australia*, 25:153–159.

Ott, M. (2012). Towards storing point clouds in PostgreSQL. *HSR Hochschule für Technik Rapperswil, Rapperswil, Switzerland*.

Peano, G. and Kennedy, H. C. (1973). *Selected Works of Giuseppe Peano*. University of Toronto Press.

Pelekis, N., Theodoulidis, B., Kopanakis, I., and Theodoridis, Y. (2004). Literature review of spatio-temporal database models. *The Knowledge Engineering Review*, 19(03):235–274.

Peters, R., Ledoux, H., and Biljecki, F. (2015). Visibility analysis in a point cloud based on the medial axis transform. In *Europhysics Workshop on Urban Data Modelling and Visualisation, Delft (The Netherlands), Nov. 23th, authors version*.

Pot, R. (2011). *System Description Noord-Holland Coast*. PhD thesis, TU Delft, Delft University of Technology.

Previtali, M., Barazzetti, L., Brumana, R., and Scaioni, M. (2014). Towards automatic indoor reconstruction of cluttered building rooms from point clouds. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1:281–288.

Ramsey, P. (2014). A PostgreSQL extension for storing point cloud (LIDAR) data.

Ravada, S., Horhammer, M., and Baris, M. K. (2010). Point cloud: Storage, loading, and visualization `http://www.cigi.illinois.edu/cybergis/docs/Kazar_Position_Paper.pdf`.

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

Rusu, R. B. and Cousins, S. (2011). 3D is here: Point cloud library (PCL). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE.

Sidirourgos, L. and Kersten, M. (2013). Column imprints: a secondary index structure. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 893–904. ACM.

Sistermans, P. and Nieuwenhuis, O. (2004). Holland Coast (the Netherlands). *EUROSION Case Study. DHV group*.

Snodgrass, R. T. (1999). *Developing time-oriented database applications in SQL*. Morgan Kaufmann Publishers Inc.

Theodoridis, Y., Sellis, T., Papadopoulos, A. N., and Manolopoulos, Y. (1998). Specifications for efficient indexing in spatiotemporal databases. In *Scientific and Statistical Database Management, 1998. Proceedings. Tenth International Conference on*, pages 123–132. IEEE.

Tian, Y., Ji, Y., and Scholer, J. (2015). A prototype spatio-temporal database built on top of relational database. In *Information Technology-New Generations (ITNG), 2015 12th International Conference on*, pages 14–19. IEEE.

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

van Oosterom, P., Martinez-Rubi, O., Ivanova, M., Horhammer, M., Geringer, D., Ravada, S., Tijssen, T., Kodde, M., and Gonçalves, R. (2015a). Massive point cloud data management: design, implementation and execution of a point cloud benchmark. *Computers & Graphics*.

van Oosterom, P., Martinez-Rubi, O., Tijssen, T., and Gonçalves, R. (2015b). Realistic benchmarks for point cloud data management systems.

van Oosterom, P. and Vijlbrief, T. (1996). The spatial location code. In *Proceedings of the 7th International Symposium on Spatial Data Handling, Delft, The Netherlands*.

Wang, W. and Hu, Q. (2014). The method of cloudizing storing unstructured LiDAR point cloud data by MongoDB. In *Geoinformatics (GeoInformatics), 2014 22nd International Conference on*, pages 1–5. IEEE.

Westoby, M., Brasington, J., Glasser, N., Hambrey, M., and Reynolds, J. (2012). 'structure-from-motion'photogrammetry: A low-cost, effective tool for geoscience applications. *Geomorphology*, 179:300–314.

White, J. C., Wulder, M. A., Vastaranta, M., Coops, N. C., Pitt, D., and Woods, M. (2013). The utility of image-based point clouds for forest inventory: A comparison with airborne laser scanning. *Forests*, 4(3):518–536.

Zlatanova, S. (2006). 3d geometries in spatial DBMS. In *Innovations in 3D geo information systems*, pages 1–14. Springer.