Organizing and visualizing point clouds with continuous levels of detail

van Oosterom, Peter; van Oosterom, Simon; Liu, Haicheng; Thompson, Rod; Meijers, Martijn; Verbree, Edward

DOI
10.1016/j.isprsjprs.2022.10.004

Publication date
2022

Document Version
Final published version

Published in
ISPRS Journal of Photogrammetry and Remote Sensing

ISPRS Journal of Photogrammetry and Remote Sensing 194 (2022) 119–131

Contents lists available at ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

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

Organizing and visualizing point clouds with continuous levels of detail

Peter van Oosterom [a,*], Simon van Oosterom [b], Haicheng Liu [c], Rod Thompson [a], Martijn Meijers [a], Edward Verbree [a]

[a] Faculty of Architecture and the Built Environment, Delft University of Technology, Delft, The Netherlands
[b] Faculty of Aerospace Engineering, Delft University of Technology, Delft, The Netherlands
[c] Alibaba Group, China

ARTICLE INFO

Keywords:
nD point clouds
Continuous level of detail (cLoD)
Space Filling Curve (SFC)
Perspective view selection

ABSTRACT

Point clouds contain high detail and high accuracy geometry representation of the scanned Earth surface parts. To manage the huge amount of data, the point clouds are traditionally organized on location and map-scale; e.g. in an octree structure, where top-levels of the tree contain few points suitable for small scale overviews and lower levels of the tree contain more points suitable for large scale detailed views. The drawback of this solution is that it is based on discrete levels, causing visual artifacts in the form of data density shocks when creating the commonly used perspective views. This paper presents a method based on an optimized distribution of points over continuous levels, avoiding the visualization shocks. The traditional distribution ratio's of data amounts over discrete levels of raster or vector data is considered the reference. How to convert this to point clouds with continuous levels (still benefiting from the proven advantages of the data distribution in discrete levels for efficient access at a wide range of scales)? In our solution, for each point a cLoD (continuous Level of Detail) value is computed and added as dimension to the point. A SFC (Space Filling Curve)-based nD data clustering technique can be used to organize the points, so that they can be efficiently queried. It should be noted that also other multi-dimensional indexing and clustering techniques could be applied to realize continuous levels based on the cLoD value. Besides the mathematical foundation of the approach also several implementations are described, varying from a 3D web-browser based solution to an augmented reality point cloud app in a mobile phone. The cLoD enables interactive real-time visualization using perspective views without data density shocks, while supporting continuous zoom-in/out and progressive data streaming between server and client. The described cLoD based approach is generic and supports different types of point clouds: from airborne, terrestrial, mobile and indoor laser scanning, but also from dense matching optical imagery or multi-beam echo soundings.

1. Introduction

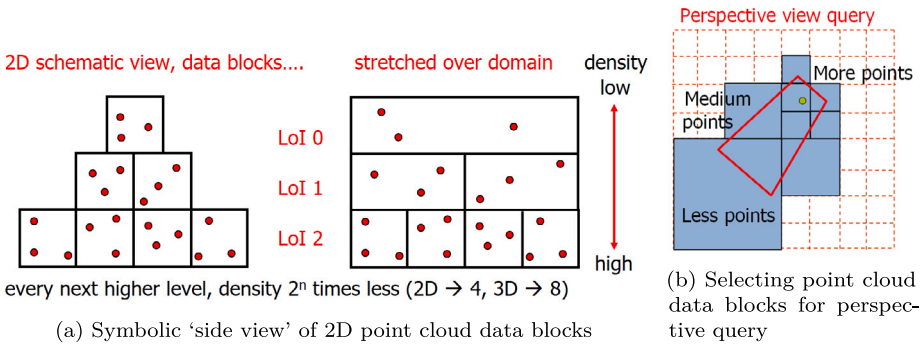
Big geo-data requires good spatio-temporal data organization, including levels of detail that allow to zoom in from high-level overviews (complete countries/continents) to the smallest detail (as the curb stones of a sidewalk) and everything in between. Currently, only a limited number of discrete detail levels can be provided. In spatial information sciences, a major challenge is to bridge the gap between the big data generated in the form of point clouds by various sensors and processing techniques (lidar, dense matching optical imagery, multi-beam echo sounding, PS-InSAR) and the applications based on traditional gridded or object representations and functionality. The world's largest data sets, despite their high potential value, are heavily under exploited due to the problematic data management, access, and limited software tools that are available to directly employ them.

Fig. 1(a) shows that the current state of the art is to organize large point clouds in discrete levels of data pyramids (Arikan et al., 2014; Oosterom et al., 2016). However, discrete levels have well known restrictions, with the abrupt transition between various levels being very disturbing (cf. Fig. 1(b)). The human eye is sensitive to the effect of multi-level organization as can be noticed in Fig. 2, but the same applies to 'suboptimal' analysis based on discrete multi-level organization.

In our nD-PointCloud model the cLoD becomes a true continuous dimension and it is used in the data organization for visualization as well as for spatial analyses. We propose to generate random cLoD values according to an optimized distribution function for continuous levels, add this as dimension, and use high-resolution nD space-filling curves that have shown to be beneficial in organizing multi-dimensional spaces and for task decomposition in parallel computing (Nivarti et al., 2015; Xia

* Corresponding author.
E-mail address: p.j.m.vanoosterom@tudelft.nl (P. van Oosterom).

https://doi.org/10.1016/j.isprsjprs.2022.10.004
Received 3 February 2022; Received in revised form 30 September 2022; Accepted 3 October 2022
Available online 23 October 2022



(a) Symbolic 'side view' of 2D point cloud data blocks

(b) Selecting point cloud data blocks for perspective query

Fig. 1. Discrete levels (scale/importance) data organization in blocks.

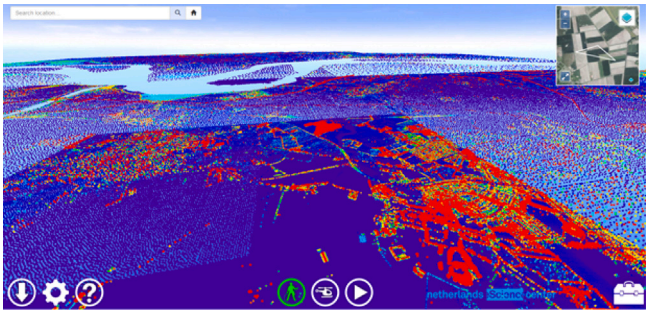


Fig. 2. Potree webviewer with AHN-2 data showing discrete levels (blocks as artifact).

and Liang, 2016; Guan et al., 2018), but which have not been applied yet to these volumes and higher dimensions (>3D). The hypothesis is that the nD-PointCloud based implementation will outperform the grid-approach in accuracy and computation/retrieval speed.

Our research methodology is 'design science research' (Hevner and Chatterjee, 2010): gaining knowledge through designing and building tangible artifacts. In earlier research it was proven that high-resolution nD-SFCs (Space Filling Curves) in combination with parallel processing by employing and adapting solutions driven by High Performance/High Throughput Computing (HPC/HTC) environments (Guan and Wu, 2010; Gentzsch et al., 2011; Memon et al., 2014) have the potential to cluster points close in reality (Fig. 2).

Oosterom et al. (2015) and Liu et al. (2020a) developed an Index-Organized Table (IOT) approach to address nD window queries on massive point clouds. Considering characteristics of spatial data and applications, the approach utilizes a SFC to encode each nD point into a one-dimensional key. Then, a B+-tree organized table is built to store and index these keys. When querying, the approach transforms the nD query window into a set of one-dimensional SFC ranges for selection (Fig. 3(a)). In this transformation, an nD histogram is used given non-uniform data distribution. The approach on the one hand achieves high efficiency to index all related dimensions in the query, while on the other hand avoids the time-consuming block unpacking and filtering process of block-based approaches. Benchmarking results (Liu et al., 2020a,b) confirm the superiority of this approach over conventional block-based approaches, especially when the distribution of nD points is skewed.

Any query geometry can be approximated and represented by a set of SFC ranges (Fig. 3(b)). So, we can extend the IOT approach for more query geometries beyond orthogonal windows. This can be expected to achieve better results than a Minimum Bounding Box (MBB) strategy which loosely approximates the search region by its MBB and locates all points inside. The MBB works well if the volume of the query region is a large proportion of the volume of the bounding box, but falls down if the query region runs at awkward angles to the axes. It also becomes more problematic with larger numbers of dimensions.

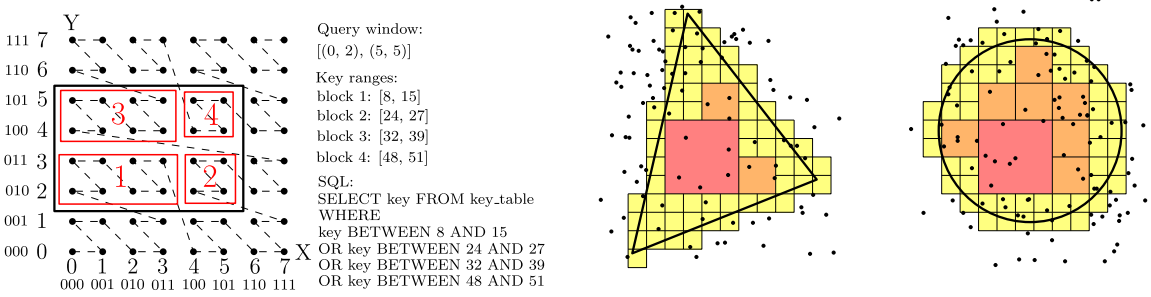
This paper describes our approach to realize vario-scale point clouds. These are point clouds without data density shocks in case of perspective views and when gradually zooming in or out. The rest of the paper is organized as follows: Section 2 reviews previous studies and current best practices concerning nD point clouds management and querying. Section 3 is answering the question: What is an optimized continuous distribution of point clouds over continuous levels. Then, based on the answer to the continuous distribution question, Section 4 derives the distribution function and the cumulative distribution function for 1D to nD point clouds. Section 5 explains how to use the cLoD value in order to approximate a certain point cloud density in the output, which may be non-uniform; e.g. in the case of a perspective view. Based on this, Section 6 presents several implementations using the cLoD enriched point clouds, including a perspective view query frequently used in practice. Section 7 concludes the paper and lists options for future work.

2. Related work

This section first presents commonly applied solutions for point cloud data organization in Section 2.1. Then the Space Filling Curve (SFC) based approach for efficiently storing and retrieving nD point clouds is shortly summarized (Section 2.2). Finally, in Section 2.3 some initial post-processing approaches to get rid of discrete levels are reviewed.

2.1. Point cloud data organization

Countries and cities have programs for scanning their surfaces to create 3D models for a range of applications. For example, the Netherlands has the 'Actueel Hoogtebestand Nederland' (AHN) where every five years its surface is scanned (with 640 billion points in AHN-2, and similar numbers in the successors AHN-3, also completed, and AHN-4 to be completed before the end of 2022). Already today 35 trillion points (35.000.000.000.000 particles) data sets are used in astronomy (Crankshaw et al., 2013) and a similar sized data set has been announced in geo-information (Netherlands' road infrastructure, street furniture and buildings generated by CycloMedia; https://www.cyclomedia.com). Far larger data sets are expected, due to increased sampling resolution of the sensors, the larger areas covered and the repeated acquisition of same area for updating and monitoring changes. Given the huge volume of spatio-temporal data, traditionally a spatial index structure, such as kd-tree, R-tree, octree/quad-tree, low-resolution SFC cell's (Oosterom, 1999; Samet, 2006) is added in the database to enable efficient access to the data. However, also these indices require a lot of storage space, especially when indexing individual points. Therefore in earlier work, blocks/groups of points (of certain bucket size) are used to reduce the size of the index (Wijga-Hoefsloot, 2012; Ravada et al., 2014; Cura et al., 2016; Butler et al., 2020). Still, these indices may be rather large and the fineness of access is reduced.



(a) Executing a window query on a uniformly distributed 2D point set based on Morton encoding

(b) Searching point sets with a triangle and a circle

Fig. 3. Recursively partitioning the extent of data according to SFC regions to match different query geometries, for selecting data in IOT.

The current state of the art is organizing point clouds in discrete LoD, or data pyramids as explained in the introduction and as illustrated in Fig. 1. This organization allows fast spatial searching including LoD selection: The further away from the viewer the fewer points are selected, i.e. the higher level blocks/points (Fig. 1 right). These representations offer a discrete set of levels, i.e. multiple map-scales (Jones et al., 1996; Kilpelainen, 1997; Friis-Christensen and Jensen, 2003). Discrete levels have well known restrictions; e.g. in perspective views (more detail nearby, less detail further away), the abrupt transition between various levels is often disturbing to the human eye, in spatio-temporal analysis/computation just the predefined discrete levels are available and 'optimally detailed' representation may not be available. There is a continuous point cloud approach proposed by Microsoft (2014), but this has severe drawbacks as it is not based on using cLoD as an organizing dimension and the solution is less general (it focuses on visualization of 3D point clouds). The rising popularity of WebGL and the availability of cloud computing/ storage resources (Kodde, 2010) are changing the way in which point cloud data are consumed. Plasio and Potree (Schütz and Wimmer, 2015) are two examples of open source web point cloud viewers. In our earlier Netherlands eScience Center project 'Massive Point Clouds for eSciences' (Oosterom et al., 2015; Martinez-Rubi et al., 2015), the Potree-converter was extended to be able to visualize massive point clouds by speeding up the creation of massive multi-resolution octree based on a divide and conquer approach that splits the task into smaller pieces that can be parallel processed. However, none of the current solutions supports true vario-scale, smooth perspective (or other mixed scale) views.

2.2. SFC based point cloud organization

Using mathematical theories, materialized and validated through prototypes, we realized deep integration of space, time, and scale (cLoD) dimension, supporting huge volumes of data. This deep integration was earlier applied with fewer dimensions: Spatial and temporal dimensions in Hägerstrand's space–time cube (1970), or spatial and cLoD dimensions in the space-scale cube by Meijers and van Oosterom (2011). This section presents the overall architecture of the approach based on Liu et al. (2020a). In mathematics, a Space Filling Curve (SFC) is a continuous bijection from the hypercube in nD space to a 1D line segment, cf. Jaffer (2014). Therefore, SFCs have the ability to cluster points which are captured close in reality, close on the curve. The nD hypercube is of the order m if it has a uniform side length 2^m. Analogously, the curve C also has an order m and its length equals to the total number of $n2^m$ cells. Space-filling curves are used in task decomposition in parallel computing (Nivarti et al., 2015; Xia and Liang, 2016). The nD SFC key can be applied in combination with spherical coordinate reference systems, which is getting close to spatial representations based on discrete global grid structures (Sahr et al., 2003; Sirdeshmukh et al., 2019).

The improved SFC-based organization for point clouds was introduced by Oosterom et al. (2015), used for two spatial dimensions (Martinez-Rubi et al., 2015), and initially explored for space–time organization (Psomadaki et al., 2016), and space-scale organization (Guan et al., 2018). In realization, instead of using a (heap) table with a B-tree index, an Index Organized Table (IOT) is used, which avoids storing a large, separate index, thus not requiring to perform a join during query execution (between index and data). During the SFC calculation, it is possible to define the curve for the full resolution of the point cloud domain. This allows us to avoid storing the x, y and other organizing dimension values, since those can be decoded from the SFC key.

In terms of data management, we identify two types of dimensions: *organizing dimensions* are used to cluster and index the data such as spatio-temporal dimensions; the other *property dimensions* are not frequently used in the SQL WHERE clause are affiliated, such as color and intensity, and are usually irrelevant for data indexing. These two types of dimensions are not fixed, and may be varied depending on applications. In general, a cube refers to a 3D box with equal edge length. This geometric concept extended into nD space becomes the hypercube.

Fig. 4 presents the workflow of the IOT approach. It first encodes each nD-point to a full resolution Morton key, interleaving the bits of all organizing dimensions. Property dimensions are attached to the key. Then, the approach adopts the B+-tree to manage the point data. The querying module applies two filters, where the first filter computes the ranges for selecting keys while the second filter decodes the keys and conducts point-wise filtering to derive the final exact answer.

2.3. Resolving discrete levels by post-processing

The earlier attempts at continuous point clouds were based on post-processing the (discrete) level point clouds. van der Maaden (2019) uses a discrete level organized point cloud as starting point. Based on the viewing position, viewing direction, and opening angle the required point cloud data blocks are selected at the various levels as explained in the Introduction; see Fig. 1. Next, a post-processing step attempts to give the impression of continuity by removing from the denser data block, some points that are close the edge of the sparser data block. This is done because in a perspective view these parts of the visualization contain too many points, and the boundary with the more sparse block can be very clear to observe and distracting. The change in data distribution between adjacent blocks is sketched in Fig. 5.

The decision to keep or remove a point is based on an empty sphere condition attached to every point. This starts by assigning a radius to the points based on distance to camera/ viewing position: gradually

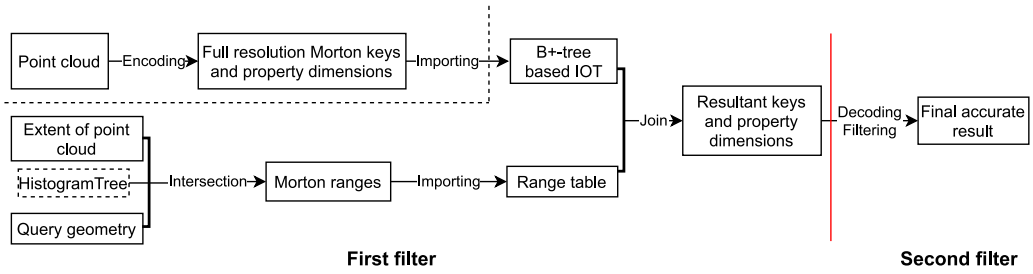


Fig. 4. The loading and querying procedure of the IOT approach.
Source: Image from Liu et al. (2021).

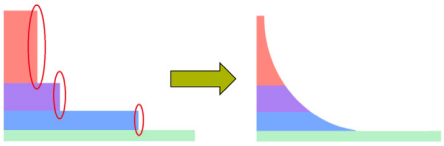


Fig. 5. Smoothing point cloud data distribution by post-processing.
Source: Image from van der Maaden (2019)

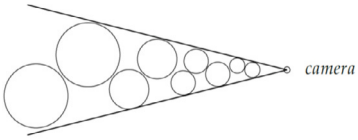


Fig. 6. Points with empty sphere are selected for the view.
Source: Image from van der Maaden (2019)

ranging from small radius nearby to a large radius far away. The points from the selected data blocks are assigned to the current view when their sphere is empty, i.e. contain no other points in the current view; see Fig. 6. The resulting images only looked smooth if the number of removed points was quite high, typically about 70%–90% of points had to be removed to give a smooth impression. Another drawback of the method is that it is not stable when panning or zooming in a certain direction - a few times points may be switched on and off repeatedly. This is unwanted behavior: when zooming or panning, points should appear or disappear cleanly. The problem of unstable behavior of continuous point clouds was solved by Schütz et al. (2019) in the context Virtual Reality. However, they do not consider big data organized at the server side (in data blocks at different levels). All data is local in the main memory of the GPU, in their case an indoor point cloud data set of 86.000.000 points. In their Virtual Reality client they apply a different post-processing technique by first adding a uniform random value (between 0 and 1) to every point. Then based on viewing position and viewing direction, every 5 to 6 frames they select 5.000.000 points at 90 frames per second (fps, for both left eye and right eye, which means a total of 180 fps).

In the selection they use the added random value to select a sufficient ratio of the points for the needed target density, which depends on distance and how much the location is in the center of the view; see Fig. 7. Because for every point the random number stays the same during an interactive VR session, their solution is stable, and has no issues with flickering points (switching points on/off in an annoying manner). Their solution is not intended/ used for big data organization at server side, where both spatial and LoD dimension are used together to optimize the multi-dimension data clustering/ indexing for fast retrieval.

3. Continuous levels

This section first presents our initial ideas to dispense with discrete levels by pre-processing and explains why this is to be preferred over post-processing approaches (Section 3.1). Next, in Section 3.2 we recapture from the well-know discrete level what are good distributions over the various levels, in order to learn from this for continuous levels. By refining the original discrete integer levels, we get a more fine grained distribution, while keeping the general good characteristic of the original distribution (Section 3.3). By infinitely refined discrete levels we arrive at a continuous level distribution which still has the good general characteristic of the original distribution (Section 3.4). Finally, Section 3.5 discusses how many levels are needed for a specific data set.

3.1. Resolving discrete levels by pre-processing

Inspired by the promising post-processing results in the above described attempts to arrive at continuous point clouds, we designed a solution based on pre-processing of the large point clouds in order to resolve the issues of: not being stable and sending too much data; or needing to have all points at the client side. Our nD-PointCloud solution is also inspired by research on vario-scale vector maps for 2D area partitions (Oosterom and Meijers, 2014; Meijers et al., 2020). The lesson obtained from the vario-scale vector maps research was: add one continuous dimension to the geometry to represent scale (LoD). So, in case of the 2D continuous LoD vector map, the result is represented by 3D geometries. Assuming, we scan assign proper cLoD values for the points then we can apply the following overall strategy for point clouds (fitting in the scheme of Fig. 4):

1. Compute the cLoD value;
2. Add this as organizing dimension, either x, y, cLoD (z and others attributes) or x, y, z, cLoD (and others as attributes) using a 3D or 4D SFC key;
3. Cluster/index the 3D or 4D point/SFC key;
4. Define perspective view selection by a view frustum with one more dimension by using cLoD to the spatial extent (depending on viewer distance and/or displacement from the center of the view); see Fig. 8.

3.2. Learning from discrete levels

How should cLoD values be computed? Should we just use (uniform) random values, like Schütz et al. (2019), or should the level values have more meaning? From raster maps or tiled vector maps being served over the web, it is possible to learn that between two discrete levels there is always a fixed ratio in scale (LoD/data density): that is, a factor 2 for every dimension. For example, the next lower (more detailed) level in the pixel data pyramid contains 2 times more data per dimension, which is 4 times, as pixel data has two dimensions (and for 3D voxels the data ratio between subsequent levels would

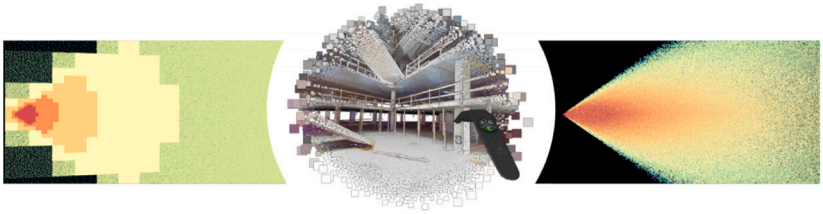


Fig. 7. Left: discrete LoD, Middle: continuous point cloud view in VR, Right: continuous LoD.
Source: Image from Schütz et al. (2019).

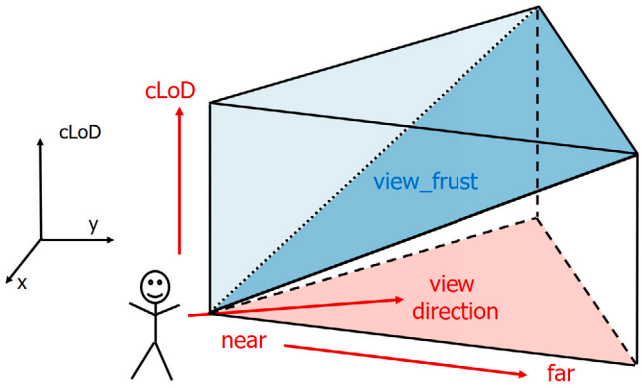


Fig. 8. Integrated space-cLoD (or scale) selection via the upper blue tetrahedron (view_frust) from the vario-scale x, y, cLoD point cloud data cube. The darker blue bottom plane of this tetrahedron is not normal geometry, but a combination of spatial and cLoD dimensions. Please note that on the vertical axis represents the cLoD dimension (and the z dimension is not shown). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Zoom levels in the Dutch well-known scale set, based on Geonovum (2012).

Zoom level	Resolution (m/pixel)	Scale denominator
0	3440,640	12.288.000
1	1720,320	6.144.000
2	860,160	3.072.000
3	430,080	1.536.000
4	215,040	768.000
5	107,520	384.000
6	53,760	192.000
7	26,880	96.000
8	13,440	48.000
9	6,720	24.000
10	3,360	12.000
11	1,680	6.000
12	0,840	3.000
13	0,420	1.500
14	0,210	750

be 8). Table 1 shows the definition of the 15 Dutch zoom levels in the data pyramid and their relationships to appropriate map scales. This Dutch 'profile' for tiling can be used by Web Map Tile Services (WMTS), with the Spatial Reference Id EPSG:28992 (Amersfoort/RD New), and the tiles are $256 \cdot 256$ pixels; see (Geonovum, 2012). For global tile sets more levels are used, both for raster and vector tiling schemes; e.g. the well-known scale set GlobalCRS84Pixel has 18 levels; see (Masó et al., 2010). For point clouds there are similar approaches using quadtree/octree with discrete levels; e.g. the 3D webviewer for the Dutch AHN-2 dataset needs 12 levels (Oosterom et al., 2016). It is important to realize that the discrete levels have a relationship to the scales for which they are suitable.

3.3. Refined discrete levels

The cLoD value should not be just a random number, but should support the level thinking that has been developed in raster and vector mapping over the past decades. So, how can we obtain a similar distribution for cLoD values compared to the existing discrete schemes? This is a hard question bothering the authors for several years. What if we do not try to solve this question directly, but first solve the question: How can we refine the existing discrete (integer) level distributions? This question was raised and answered during the keynote presentation at the ISPRS Geospatial Week 2019 (Oosterom, 2019). Fig. 9 first re-captures show the traditional distribution of data over discrete integer levels.

The ratio of the target number of points N_l per level l is given by $N_l = 2^l$ for integer values of l from 0 (most important, few points) to maximum level L (least important, most points). Note that this is for the 1D case ($n = 1$) the optimized distribution over the levels. The multi-dimensional case is quite similar $N_l = 2^{n \cdot l}$ with n the number of dimensions. In the following, we first continue with 1D data and later on the formulas for the nD case will be given. As there are no refinements to the discrete integer levels, this is called refinement $r = 0$. The probability that a point belongs to level l is defined by dividing the number of blocks at this level by the total number of blocks in all levels: $\text{Tot}_0 = \sum_{l=0}^{L} 2^l$ (for $r = 0$); see the mathematics in the next section (Eq. (2)).

Next step is to refine the discrete integer levels into discrete half levels, so doubling the number of levels in this first refinement $r = 1$. The formulas for computing the number of blocks per (half) level, and the distribution over the levels stay the same, see Fig. 10. In the formulas only the total number of blocks at refinement $r = 1$ summed over all levels is different and given in the next section (Eq. (8)). It is important to note that when summing the probabilities of two half levels that originated from the same integral level, this results in exactly the same probability as we had for this integer level. Of course, this procedure can be repeated and after 2 refinements we end up with 16 quarter levels; see Fig. 11. Again, the formulas stay the same (only now with $r = 2$) and summing 4 quarter levels that used to form one level, results in exactly the same probability as we had for this integer level. When in Figs. 10 and 11 the probabilities are summed for all refined levels belonging to one integer level, the we get back the original probability. For example, Fig. 9 it states that the integer level 0 has 6.7% probability, which is equal to the sum of probabilities of the first 2 levels after one refinement; see Fig. 10, which is in turn equal to the sum of the probabilities of the 4 first levels after two refinements; see Fig. 11.

3.4. Infinitely refined discrete levels

We can continue to refine the levels beyond two times as explained above. When $r \to \infty$ then the actual result is a continuous distribution function $f(l)$ over the levels l from 0 to $L + 1$ (next section, Eq. (12)). In addition we define also the cumulative version of the distribution function, which is called $F(l)$ over the same level l domain.

The input point cloud data gets one more dimension by adding the cLoD value to every input point. For this, the inverse of the cumulative

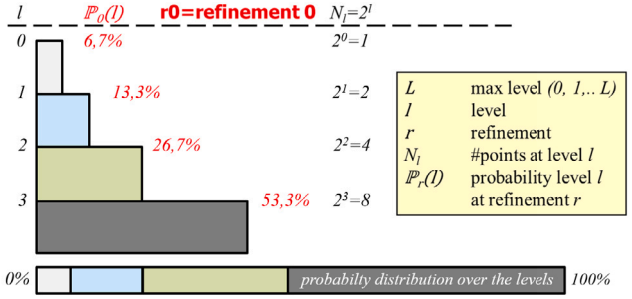


Fig. 9. Distribution over 4 discrete integer levels ($l = 0, 1, 2, 3$) in case of 1D data.

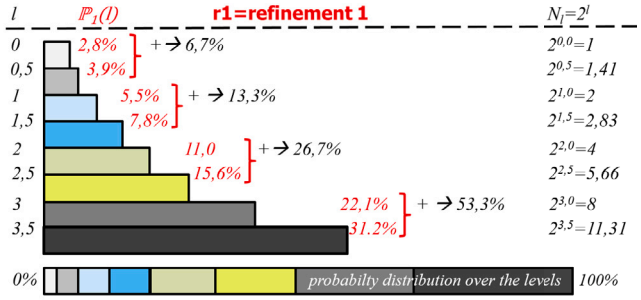


Fig. 10. Distribution over 8 discrete half levels ($l = 0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5$) in case of 1D data after one refinement $r = 1$.

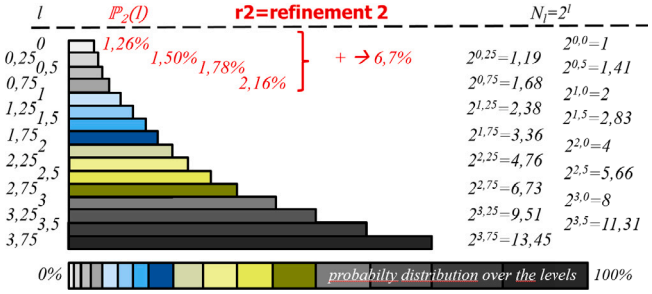


Fig. 11. Distribution over 16 discrete quarter levels ($l = 0, 0.25, .., 3.75$) in case of 1D data after two refinements $r = 2$.

distribution function is used on the output of a random generator U producing uniform distributed values between 0 and 1 (or $l = F^{-1}(U)$ more details in the next section Eq. (17)). The resulting point cloud data set has one cLoD dimension more, and this distribution of this cLoD dimension is according to the optimized continuous distribution function $f(l)$. The value of the level dimension can be used to organize the data next to xy(z), but can also be used to select the points for a specific LoD (scale) or for a perspective view (with varying LoD/scale in a single image).

3.5. How many levels are needed?

Given a certain data set with N points, how many levels would be appropriate, i.e. what is a good value for L? This question is valid for both discrete and continuous levels. We first analyze the discrete integer levels, and the outcome is also valid for refined discrete levels and continuous levels. In the 2D case about 80% of the data is at lowest discrete level with highest level number as we start numbering from the top with level 0 (and in the 3D case about 90% at lowest level). So, let us say as a first approximation that all data are at lowest level. In case of a data set like AHN-2 with about 640.000.000.000 point and if at top level (block in data pyramid) with for example 10.000 points per block, then we need to store 64.000.000 blocks in total, each containing about 10.000 points. In 2D with $L = 13$ we can host $4^{13} = 67.108.864$ blocks at level 13 (the lowest of the 14 levels named level 0, level 1, …, level 13). This is enough for AHN-2 and also the actual depth of the AHN-2 octree (given the nature of the quadtree, as the AHN-2 data is basically a 2.5D surface in 3D space), we created with the potree-converter in our earlier research (Oosterom et al., 2015). In general, the expression to define the L for an nD point cloud with N points and blocks with capacity of C points per blocks:

$$L = \left\lceil \frac{1}{n} \log_2(N/C) \right\rceil \tag{1}$$

The USGS 3D Elevation Program (3DEP) at the moment has 43.252.493.596.939 points; see https://usgs.entwine.io/. The coverage is not yet complete, so the final data set will contain more points, say about 100.000.000.000.000 (= 10^{14}) points. It is also a 2.5D surface in 3D space, so $n = 2$ and assuming the same capacity for the blocks $C = 10.000$, then we would need $L = 17$ as maximum level to store the future complete USGS lidar point cloud data set. As the refined discrete levels and the continuous levels have the same general distribution characteristics as the discrete integer levels, the same number of Levels L is sufficient. If for some reason we want to use 32 levels (maximum level $L = 31$), $n = 2$ and with block capacity $C = 10.000$, then more than 10^{22} points can be accommodated at the lowest level.

4. The mathematics

After presenting in the previous section the idea how to arrive at continuous levels with an optimized distribution, we will now present the accompanying mathematics. Section 4.1 looks at the formulas for the discrete refinement of levels, followed by formulas for the infinite refinement (or continuous levels) in Section 4.2, both for the 1D case. Next, Section 4.3 generalizes the formulas from the 1D case to the nD case. Finally, Section 4.4 presents an initial assessment by creating various distribution graphs with our formulas implemented in Matlab, showing discrete distributions with less and more refinement and also the corresponding continuous distribution.

4.1. Refined discrete levels

First the formula for the distribution over discrete levels are introduced for the 1D case. Suppose we have a point x, and want to decide with what probability x is put in some layer. For a one dimensional system with L layers, the probability of a point to be located in layer l, with $l \in \{0, 1, \ldots, L\}$, is proportional to 2^l ($l = 0$ is the top layer, and $l = L$ is the most detailed one). We can calculate the probability:

$$\mathbb{P}_0[x \to l] = \frac{2^l}{\sum_{l=0}^{L} 2^l} = \frac{2^l}{\mathrm{Tot}_0} \tag{2}$$

Now, we are going to refine the number of layers by splitting each one in half. The refinement is indicated with level r, the number of times we split the layers ($r = 0$ is unrefined). The refined levels become:

$$k \in \{0, 2^{-r}, 2 \cdot 2^{-r}, \ldots, ((L+1)2^r - 1) \cdot 2^{-r}\}$$

where $k = 0$ is the top layer, and $k = L + 1 - 2^{-r}$ is the most detailed one. If we refine the levels r times, the probability is given by:

$$\mathbb{P}_r[x \to k] = \frac{2^k}{\mathrm{Tot}_r} \tag{3}$$

We have found a recursive formula for Tot_r, from which we can express it in terms of Tot_0:

$$\mathrm{Tot}_r = \sum_{k=0}^{(L+1)2^r-1} 2^{k \cdot 2^{-r}} \tag{4}$$

$$= \sum_{k=even} \left[2^{k2^{-r}} + 2^{(k+1)2^{-r}} \right] \tag{5}$$

$$= \sum_{k=even} 2^{k2^{-r}} \left[1 + 2^{2^{-r}}\right] \quad (6)$$

$$= (1 + 2^{1/2^r})\mathrm{Tot}_{r-1} \quad (7)$$

$$= \mathrm{Tot}_0 \prod_{i=1}^{r} (1 + 2^{1/2^i}). \quad (8)$$

If we substitute this, we get a probability density for refinement r:

$$f_r(l) = 2^r \mathbb{P}_r[x \to l] = 2^l \frac{\prod_{i=1}^{r} \frac{2}{1+2^{1/2^i}}}{2^{L+1} - 1} \quad (9)$$

4.2. Infinite refinement

If we want the continuous distribution, we let $r \to \infty$, where:

$$f_\infty(l) = K2^l \quad (10)$$

where K is a normalization constant:

$$\int_0^{L+1} K2^l dl = K[2^l / \ln 2]_0^{L+1} = K\frac{2^{L+1} - 1}{\ln 2} = 1 \quad (11)$$

This results in the following exponential function:

$$f(l) = \frac{2^l \ln 2}{2^{L+1} - 1} \quad (12)$$

for $l \in [0, L+1]$. This function has CDF $F(l)$ obtained by integration $f(x)$ over x from $x = 0$ to l:

$$F(l) = \int_0^l f(x)dx = \frac{2^l - 1}{2^{L+1} - 1} \quad (13)$$

And a 1D cLoD level l is randomly generated by:

$$l = F^{-1}(U) = \log_2(U(2^{L+1} - 1) + 1) \quad (14)$$

where U is uniformly distributed between 0 and 1.

4.3. Higher dimensions

For higher dimensions, we can perform similar computations and get:

$$f_n(l) = \frac{2^{nl} n \ln 2}{2^{n(L+1)} - 1} \quad (15)$$

Where n is the number of dimensions. This function has CDF:

$$F_n(l) = \frac{2^{nl} - 1}{2^{n(L+1)} - 1} \quad (16)$$

And a nD cLoD level value is generated by:

$$l = \frac{1}{n} \log_2(U(2^{n(L+1)} - 1) + 1) \quad (17)$$

4.4. Distribution graphs

The developed formulas are used in the Matlab script (see Appendix) to produce the graphs in this section. It should be noted that all distribution graphs are created for the 1D case, that is, have $n = 1$. The nD counterparts look very similar, but have more steep appearance.

5. Using the cLoD values

After having extended the point attributes with cLoD dimension and values for the individual points, then next challenge is how to use the cLoD value. Section 5.1 first explores the relation between level values and expected data density for the discrete levels. Next Section 5.2 analyses the use continuous levels for producing visualizations with uniform scale. Finally, Section 5.3 shows how to use cLoD values for mixed scale situations, specifically the perspective view.

5.1. Expected density at discrete levels

Let us assume that we have in total N points in the data set and that we have an nD domain in which every dimension has the extent E. So, this is a data (hyper) cube, with equal size of all dimensions. Then total expected data density is given by $D = N/E^n$, that is without any levels. With the discrete probability function (nD case) at refinement r defined as:

$$\mathbb{P}_{r,n}[x \to l] = \frac{2^{l \cdot n}}{\mathrm{Tot}_{r,n}} \quad (18)$$

then the expected density at discrete level l at refinement r is given by:

$$D_{r,n}(l) = \frac{N}{E^n} \mathbb{P}_{r,n}[l] \quad (19)$$

Note that there is a direct linear relation between probability of a level $\mathbb{P}_{r,n}[l]$ and expected density $D_{r,n}(l)$!

5.2. Uniform scale, continuous levels

The continuous dimension level also corresponds to data density. If we take a slice of the cLoD values that belongs to one discrete integer level, that is the semi-open range $[l, l+1)$, then we get exactly the same expected density as that of the discrete level l. It should be noted that the 'thickness' of single value of cLoD is 0, which gives near 0 probability of having any point with exactly that value (and thus 0 expected density at that level). Therefore, we did take a slice corresponding to an integer level, but we can take any range of cLoD values. It is very convenient to take the range from the top (cLod= 0) to a specific level (cLoD= l) as in the top there are rather limited number of points anyhow. This corresponds well to the Cumulative Distribution Function (CDF) for nD case $F_n(l)$ with l between 0 and $L+1$, see Eq. (16). The expected Cumulative Density CD_n at continuous level l for nD case is:

$$CD_n(l) = \frac{N}{E^n} F_n(l) \quad (20)$$

with N total points in data set and E^n size spatial domain in the nD case. Note that there is again a direct linear relation between continuous level (CDF $F_n(l)$) and the expected cumulative density $CD_n(l)$!

Let us have a look at a simple concrete example of a 2D world ($n = 2$) with extent 5×5 m^2 ($E = 5^2$), 10.000.000 points ($N = 10.000.000$), $L = 6$, and continuous levels. This means that $0 \le l < 7$, the maximum total density = 10.000.000/25 = 400.000 points/m^2 and putting in these numbers in the generic Cumulative Density function (Eq. (20)) we get:

$$CD_2(l) = 3149,60(2^l - 1) \quad (21)$$

When having a target cumulative density TCD_2 value, the cLoD value l can be obtained by rewriting Eq. (21) into the following form:

$$TCD_2 = 3149,60(2^l - 1) \Rightarrow \quad (22)$$

$$2^l = 1 + (TCD_2/3149,60) \Rightarrow \quad (23)$$

$$l = \log_2(1 + (TCD_2/3149,60)) \quad (24)$$

Assume we have a rendering budget $B = 100.000$ points no matter what is visualized. If we know the spatial extent of a query, how can we compute the cLoD level that will deliver a number of points within our budget B? The following three queries mimic a zoom out action with a square (see Fig. 13 Left):

- 1×1 m^2 box $\Rightarrow TCD_2 = 100.000$ points/m^2 $\Rightarrow l = 5,03$
- 2×2 m^2 box $\Rightarrow TCD_2 = 25.000$ points/m^2 $\Rightarrow l = 3,16$
- 4×4 m^2 box $\Rightarrow TCD_2 = 6.250$ points/m^2 $\Rightarrow l = 1,58$

Note that in case of $1/2 \times 1/2$ m^2 box (not shown in Fig. 13 Left) we would arrive at the maximum possible density $TCD_2 = 400.000$ points/m^2 $\Rightarrow$ $l = 7,00$. Zooming in more will not result in higher density as we are already at the maximum cLoD l of 7.

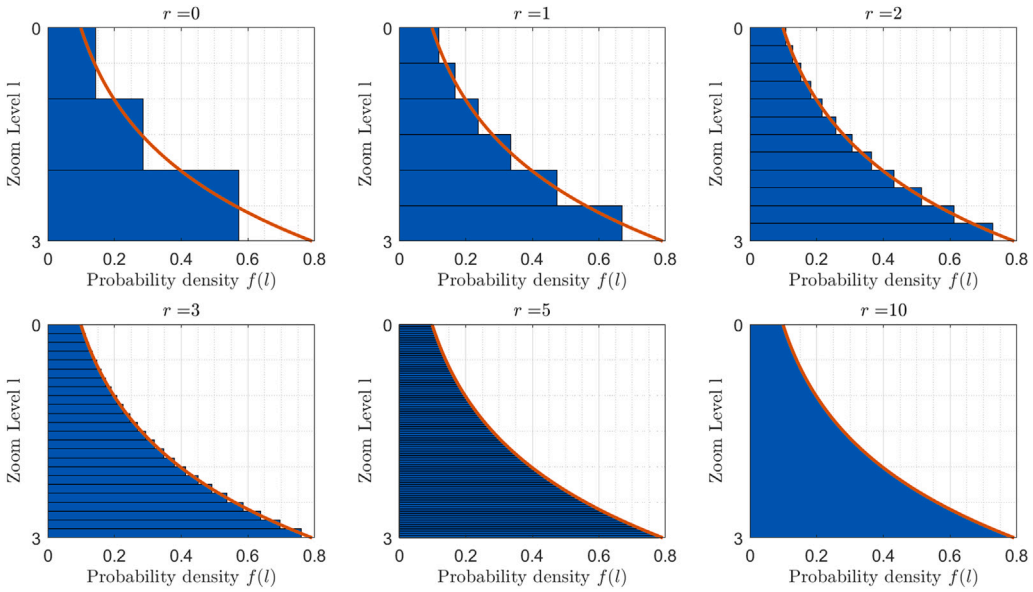


Fig. 12. Distribution over the refined levels with dimension n = 1, max levels L = 2, refinements r = 0, 1, 2, 3, 5, and 10 (note that the blue bars depict the probability of discrete levels, while the red curve depicts the exact, continuous probability function).

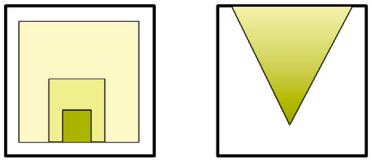


Fig. 13. Left: Uniform scale, three queries, Right: Perspective view query looking up, both in the 5×5 m^2 domain (note darker means higher density).

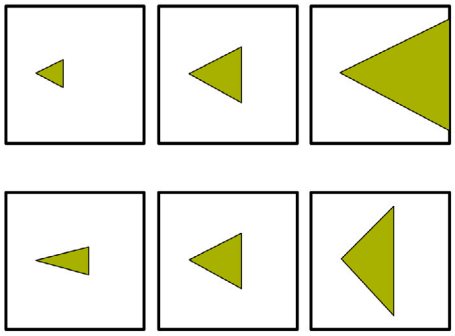


Fig. 14. Viewer is left side vertex, looking to the right, Top: changing the selected area size (flying out), Bottom: changing the field of view.

5.3. Mixed scale, continuous levels

The density should not be constant in a perspective view, but depend on the distance d to the viewer. For example, when at a distance of 1 meter the density could compare to 1×1 m^2 box with cLoD $l = 5,03$ and when at 4 meter the density could compare to 4×4 m^2 box with cLoD $l = 1,58$ in our example from the previous subsection. We use Eq. (24) with target cumulative density $TCD_2 = B/d^2$ (with d for distance and B for our budget) to obtain the cLoD l value anywhere in the domain. Integrating the wanted density over all distances in the query region, the triangular shaped view frustum, will give the expected number of points; see Fig. 13 Right.

Now let us see what happens when we change the selected area size or change the field of view (width of viewing angle). When enlarging the selected area size, the shape of view frustum is staying the same, just getting bigger (Fig. 14 Top). When the budget B remains the same the cLoD values should get smaller to get lower density. Note that the actual cLoD value should still be distance dependent. The same reasoning applies when we change the field of view (width of viewing angle (Fig. 14 Bottom)).

6. Implementations

In this section three different implementations of the continuous levels approach will be shown. The first application is a 3D continuous webviewer for points using the cLoD to avoid the visual distraction of the state-of-the art webviewers having block boundaries with high density on one side and low density on the other side. The second application of cLoD is in the context of an Augmented Reality (AR) application for a mobile phone. The third application is a database query using cLoD in the SQL where-clause.

6.1. 3D continuous webviewer

The cLoD has been used to realize a 3D data organization by using xy and cLoD to create point cloud data blocks at the server side (Guan, 2020). The test data is (part of) AHN-2. Based on the earlier defined approach (see Section 3.5), the value for maximum levels L is obtained and the cLoD values are added to the points. The 3D blocks are created by grouping the points in integer levels using rounding of cLoD. Next per level l, the points are put in 2^l by 2^l grid cells based on their xy-values. The resulting blocks are organized in a full 3D octree with xy and cLoD organization. The points also have their z value (but not as organizing dimension), which is used in the rendering. The webclient is requesting the relevant data blocks by a combined xy-cLoD request (based on overlap with the view frustum of a perspective view query). The content of the received data blocks is used to create the interactive 3D visualization now deciding at individual point level if a point should be displayed. This depends on the distance to the viewer and the actual cLoD value of the point. When the user is zooming, panning and rotating, additional data blocks may be needed and retrieved. In Fig. 15-middle the selected points are displayed and the color assigned to a point is based on cLoD (or data density) from the point of view of a third person. There is also a kind of analysis/debug mode to show the retrieved data blocks by depicting their edges with black lines. The 3D point cloud webviewer is available at: http://47.112.97.110/

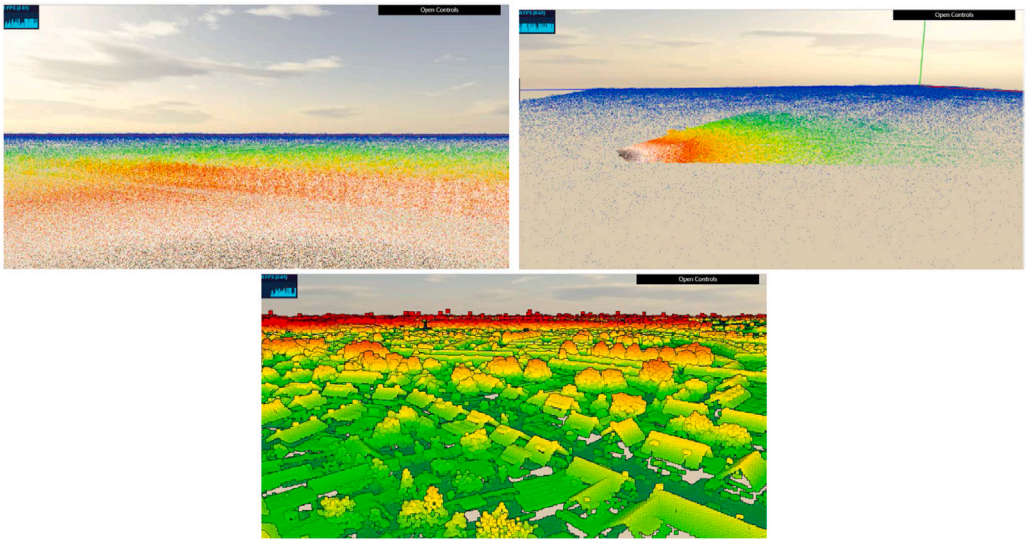


Fig. 15. 3D point cloud webviewer, Top left: first person colored by cLoD = density, Top right: same for third person view, Bottom: first person view rendering color by height. *Source:* Images from Guan (2020).

PointCloud/try.html with AHN data near Oirschot (tip: press z-key and use scroll wheel for zooming, left mouse for rotating, right mouse for panning).

6.2. Augmented Reality

Though getting more powerful, the hardware of smartphones is still limited. When working with larger point cloud data sets it can easily result in too many data points for the smartphones. Augmented Reality (AR) is based on a combination of real and virtual worlds, and supports interactive and real-time rendering. The challenge to deal with large point cloud data sets with limited availability of memory, CPU and GPU resources of mobile devices and reaching relatively high visual quality and performance requirements has been solved. A prototype AR system was developed using ARCore and the Unity game engine. The prototype uses cLoD to select density every X frames with X a parameter which can be specified, in order to reduce the number of points (not needed for viewer when at certain distance). The source code is available at https://github.com/LiyaoZhang0702/AR_PointCloud. Fig. 16 shows two screenshots from the smartphone AR application with point cloud models. The solution presented indeed reduces the number of points by selecting based on the cLoD value and distance, which resulted in a sufficient high frame rate for use in the AR application on a smartphone, while still presenting good visual quality (Zhang et al., 2020).

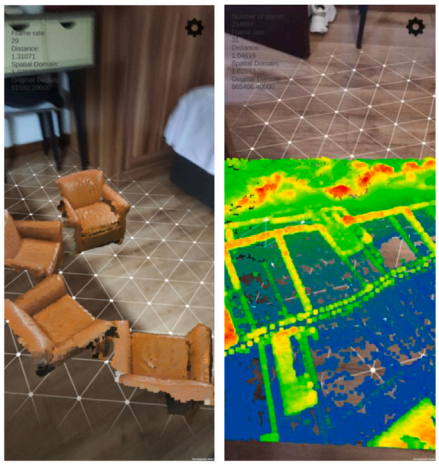


Fig. 16. AR smartphone app using cLoD value, Left: Adding several point cloud chairs to the room, Right: putting part of AHN on the ground floor in the room. *Source:* Images from Zhang (2020).

6.3. Database nD convex polytope query

When combining multiple dimensions, such as xyz spatial, t temporal, and cLoD into a 5D point cloud stored in the database, how can we select the relevant points (as our geometries and their operations are typically limited in the database to 3D; e.g. a polyhedron)? Take for example the selection query needed for a perspective view on a 3D building during a specific moment in the construction phase. There could even be more attributes used as organizing dimensions, and in general we speak of nD point clouds. The nD convex polytope is a relative simple way to express the query region by intersecting half-spaces, each defined by a hyperplane inequation (Liu et al., 2021). When specifying the view frustum the xyz and cLoD dimensions are used (in the from of $a_i x + b_i y + c_i z + d_i cLoD < f_i$ as in Fig. 8), and when specifying the time interval this can be done by two simple inequations: $tmin < t$ and $t < tmax$. If we use the relation between distance to decide

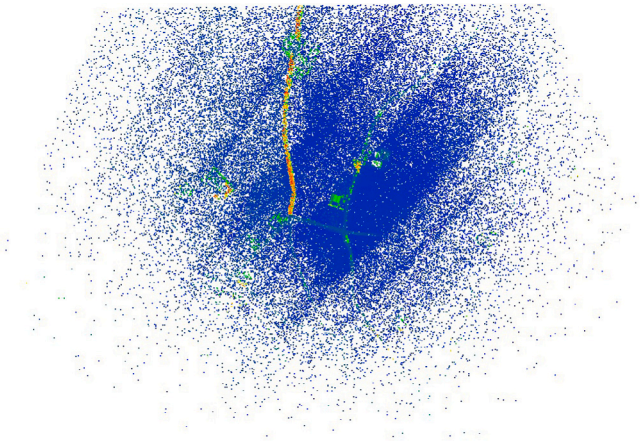


Fig. 17. Result of database query using cLoD while looking straight down. *Source:* Image from Liu et al. (2021).

```
L = 31;                                   % max level
R = [0,1,2,3,5,10];                       % refined levels
    to be drawn

for i = 1:length(R)                       % draw refined
   levels
    r   = R(i);
    [l,P]   = Prob(r,L);
    subplot(2,3,i)
    barh(l+2^(-r-1),fliplr(P*2^r),1)
    yticks([0,L+1]);
    yticklabels({num2str(L+1),'0'})
    ylim([0,L+1])
    hold on
    l       = linspace(0,L+1,1000);
    plot(2.^l *log(2)/(2^(L+1)-1),fliplr(l),'linewidth
       ',2)
    title(strcat('$r =$ ',num2str(r),'                '
       ),'Interpreter','Latex')
    ylabel('Zoom Level l','Interpreter','Latex')
    xlabel('Probability density $f(l)$','Interpreter',
       'Latex')
    grid on; grid minor
    YGrid = 'off';
end

set(gcf,'Position',[200 200 1000 500]) % positioning
   of the figure window

function [l,P]= Prob(r,L)                 % probability
   levels l at refinement r
l = 0:0.5^r:L+1-0.5^r;                    % compute
   refined levels l
P = (2.^l)/Tot(r,L);                      % prob= #points
    at level l / total
end

function tot = Tot(r,L)                   % recursive
   computation of total #points
if r ==0
    tot = 2^(L+1)-1;
else
    tot = (1+2^(1/2^r))*Tot(r-1,L);
end
end
```

Fig. 18. Matlab code to produce the various distribution graphs over the refined discrete levels.

the needed density/cLoD values, this results in non-linear equation. This can be used directly, or approximated by linear equations. The approach is very generic, easy to use and has been tested up to 10D point clouds using various types of nD convex polytopes for selection (2D to 10D prisms and 2D to 10D splices) in an efficient manner (Liu et al., 2021). Fig. 17 presents the result of a query on AHN data looking straight down and with varying density depending on distance was realized by using inequalities using cLoD in the SQL-where clause. Note that we can observe the overlapping flight paths of the data acquisition phase. These "shocks" are in the data, and are not artifacts of the viewing software. We now see more the real data and not 'uniformized' data as the result of the potree-converter gives as in Oosterom et al. (2015). Please also note that if the input data is uniformly distributed, then true random-selection will also result in a uniform distribution at higher cLoD levels as all regions in the domain are treated in same manner. However, If the input data has another distribution in space, the random sample based levels keep the nature of this distribution (as much as possible). Allowing the data user/viewer to see the true nature of the point cloud data set. It should finally be noted that there could also be applications in which a more uniform distributions in higher levels is preferred (as produced by the potree-converter).

7. Conclusions

In this paper we described our approach to realize vario-scale point clouds without data density shocks in the case of perspective views and when gradually zooming in or out, as current state-of-the-art solutions are struggling with. We answered the question: what is an optimized continuous distribution of point clouds into continuous levels, by (infinite) iterative refinement of discrete levels. We provided both the continuous distribution function and the cumulative distribution function for 1D and nD point clouds. The continuous distribution function is used together with a uniform random number generator to add the cLoD dimension to the point cloud. Together with the spatial–temporal dimensions xy(zt) the cLoD dimension is used to cluster and index the point cloud data. We next explained how to use the cLoD value in order to arrive at a certain expected point cloud density by using the cumulative distribution function. Depending on the type of query, the expected output density may be uniform (fixed-scale) or non-uniform (mixed-scale, e.g. in the case of a perspective view). Finally we presented several implementations using the cLoD enriched point clouds.

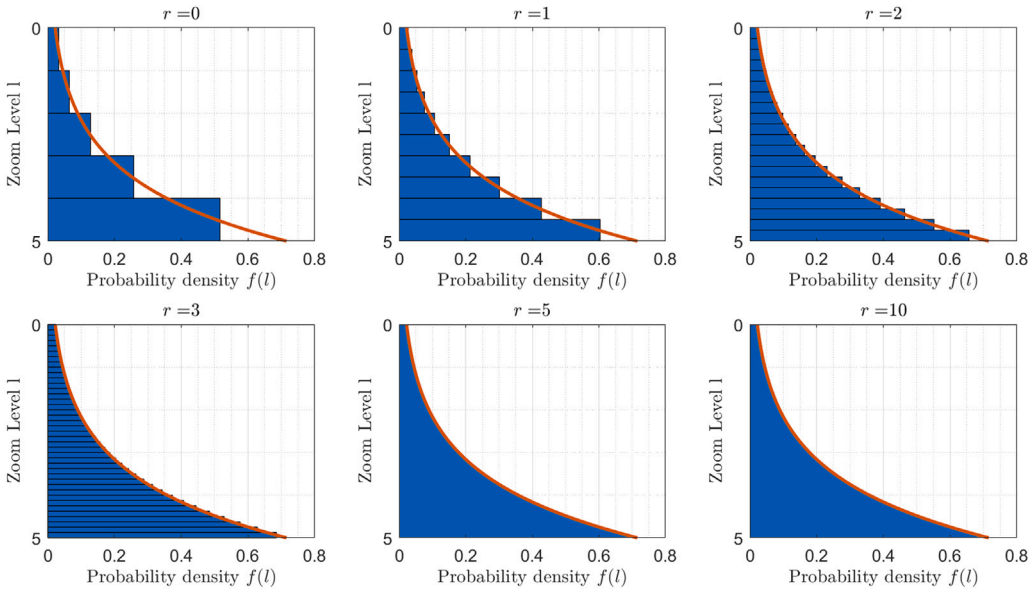


Fig. 19. Distribution over the refined levels with dimension n = 1, max levels L = 4, refinements r = 0, 1, 2, 3, 5, 10.

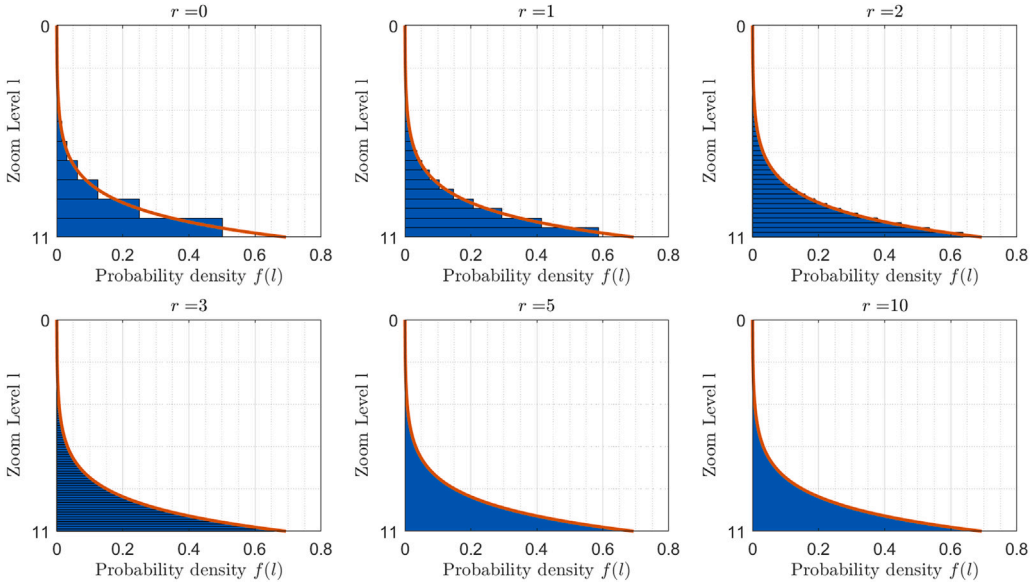


Fig. 20. Distribution over the refined levels with dimension n = 1, max levels L = 10, refinements r = 0, 1, 2, 3, 5, 10.

Point clouds such as AHN or 3DEP, are getting so large that just organizing them by spatial–temporal dimensions may not be sufficient to cluster and index them adequately them for users. For supporting huge point clouds the cLoD dimension is needed in data organization, not only for visualization purpose, but also for all kinds of other computations (analysis, simulations). Our method to add the cLoD dimension to the point cloud results in a stable solution; i.e. points not flickering on and off in a visualization. The presented cLoD computation is optimized in the sense that this has the same factor 2 per dimension properties as the well known raster and vector data pyramids on which many interactive applications are build, supporting a range of zoom options. Of course, there is always more research to be done:

- Instead of computing the cLoD values only via uniform random number and the continuous distribution function, also semantics may be used. If we have added a classification to the point clouds we could use this to improve the cLoD values. For example, in an 3D indoor point cloud model supporting navigation, the portals of the doors may be made more important/emphasized.
- Similar to the previous bullet to improve cLoD values by semantics, we could also try to improve the cLoD values by geometry. For example, in case of sparse points in a 3D space (but organized in some 1D manner) we might want to increase the cLoD value as there are few points on thin objects (such as power cable), while we would still like to see them from a distance. Same reasoning could be applied to emphasize edges and make the cLoD value of points near edges more important.
- Our new cLoD approach was just tested with a relatively small data set, e.g. parts of AHN, the next step is to apply the cLoD vario-scale point cloud approach to the complete AHN data set. Next to this we want to combine different (density) point cloud data sets (e.g. both AHN and the Icesat-2 or GEDI lidar data, two new space-based laser altimetry missions by NASA; see (Markus et al., 2017; Dubayah et al., 2020)) and/or multi-temporal data sets (e.g. AHN-1, AHN-2, AHN-3, AHN-4, . . .) in one environment.

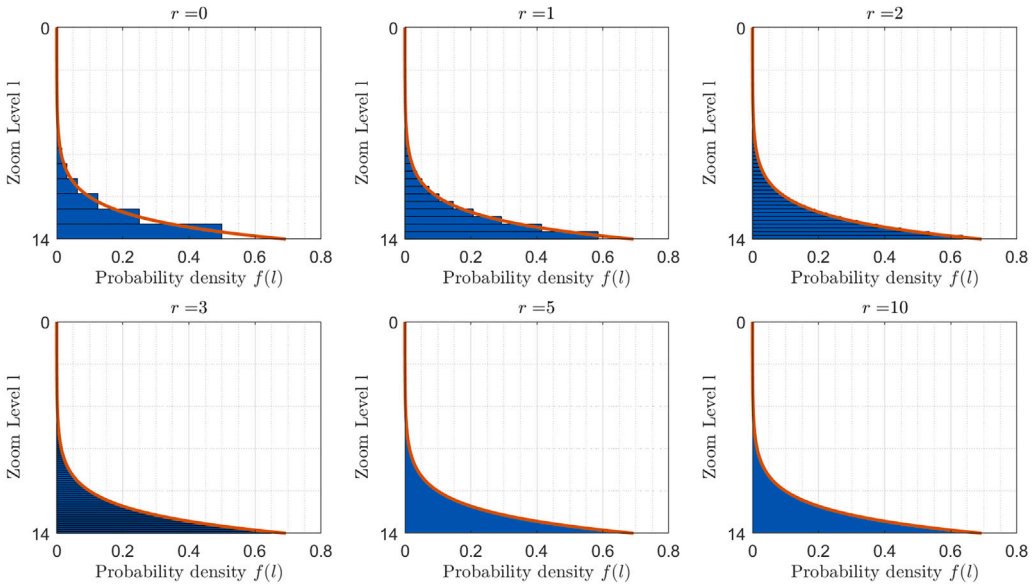


Fig. 21. Distribution over the refined levels with dimension n = 1, max levels L = 13, refinements r = 0, 1, 2, 3, 5, 10.

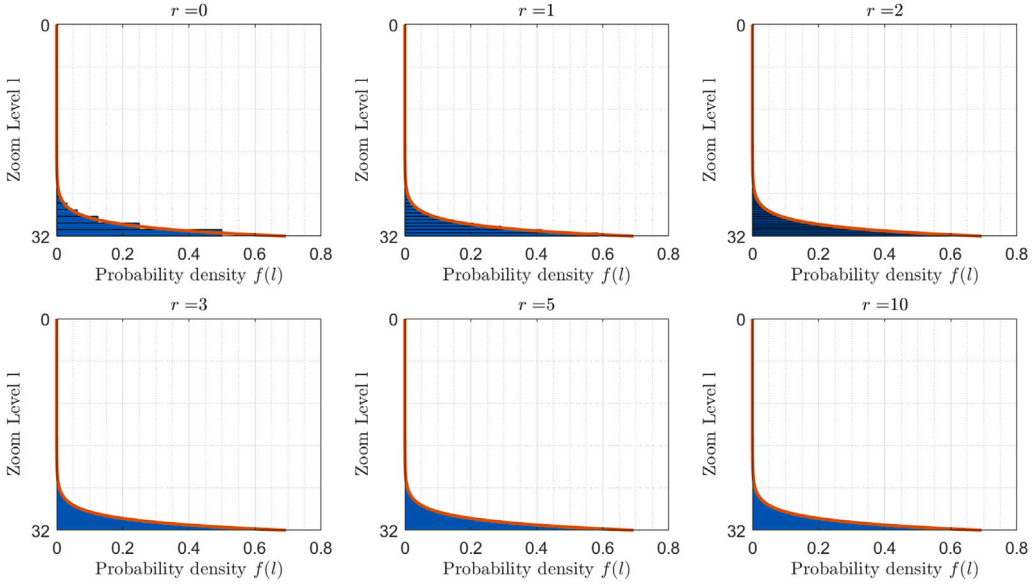


Fig. 22. Distribution over the refined levels with dimension n = 1, max levels L = 31, refinements r = 0, 1, 2, 3, 5, 10.

- We emphasized the used of cLoD for visualization, but as claimed above cLoD should also be useful to improve computations (analysis or simulations) for various tasks: solar energy potential, viewshed/ line-of-sight (incl. vegetation), 3D routing (e.g. for a drone), change detection/deformations, volume analysis computations, hydrology/ flow over surface, vegetation analysis, etc. We should prove the claims by at least showing some directions of how this can be done and it would be even better to develop prototypes.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is funded by the Netherlands eScience center (by SURF & NWO) as project number 27020G14 'nD-PointCloud continuous level representation for spatio-temporal phenomena in Open Point Cloud Maps'.

Appendix. Matlab script

Fig. 18 shows the Matlab code for the refined level computations as used to draw the distribution graphs in the paper; see Section 4.4. There is a key role for the discrete distribution function $Prob(r, L)$, which returns the probability for the $(L + 1) \cdot 2^r$ different levels l when we have at maximum $L + 1$ corresponding integer levels and then have r refinements. In the main text, Fig. 12 already shows the distribution for $L = 2$. In this Appendix we also show the graphs for $L = 4, 10, 13, 31$ (for

the same refinements $r = 0, 1, 2, 3, 5, 10$) in respectively the Figs. 19, 20, 21, and 22. Having $L = 13$ should be sufficient for most applications within the Netherlands, having $L = 31$ is more than sufficient for global applications (18 or 20 are most likely enough). Note that both the Matlab script and the graphs just show the 1D case.

References

Arikan, M., Preiner, R., Scheiblauer, C., Jeschke, S., Wimmer, M., 2014. Large-scale point-cloud visualization through localized textured surface reconstruction. IEEE Trans. Vis. Comput. Graphics 20 (9), 1280–1292.

Butler, Howard, Chambers, Bradley, Hartzell, Preston, Glennie, Craig, 2020. DAL: An open source library for the processing and analysis of point clouds. Comput. Geosci. 148, 104680. http://dx.doi.org/10.1016/j.cageo.2020.104680.

Crankshaw, D., Burn, R., Falck, B., Budavári, T., Szalay, A.S., Wang, J., 2013. Inverted indices for particle tracking in petascale cosmological simulations. In: SSDBM July'13. Baltimore, MD, USA.

Cura, Rémi, Perret, Julien, Paparoditis, Nicolas, 2016. A scalable and multi-purpose point cloud server (PCS) for easier and faster point cloud data management and processing. ISPRS J. Photogramm. Remote Sens. 127, http://dx.doi.org/10.1016/j.isprsjprs.2016.06.012.

Dubayah, Ralph, Blair, James Bryan, Goetz, Scott, Fatoyinbo, Lola, Hansen, Matthew, Healey, Sean, Hofton, Michelle, Hurtt, George, Kellner, James, Luthcke, Scott, Armston, John, Tang, Hao, Duncanson, Laura, Hancock, Steven, Jantz, Patrick, Marselis, Suzanne, Patterson, Paul L., Qi, Wenlu, Silva, Carlos, 2020. The global ecosystem dynamics investigation: High-resolution laser ranging of the earth's forests and topography. Sci. Remote Sensing (ISSN: 2666-0172) 1, 100002. http://dx.doi.org/10.1016/j.srs.2020.100002.

Friis-Christensen, C.S., Jensen, A., 2003. Object-relational management of multiply represented geographic entities. In: Proceedings of the Fifteenth International Conference on Scientific and Statistical Database Management. July 9–11, 2003, Cambridge, MA, USA.

Gentzsch, W., Girou, D., Kennedy, A., Lederer, H., Reetz, J., Riedel, M., Schott, A., Vanni, A., Vazquez, M., Wolfrat, J., 2011. DEISA - distributed European infrastructure for supercomputing applications. J. Grid Comput. 9 (2), 259–277.

Geonovum, 2012. Nederlandse richtlijn tiling, versie 1.1 (in Dutch),Vol. 2012. Dutch Government, p. 15.

Guan, Xuefeng, 2020. 3D Point Cloud Visualization System (in Chinese),Vol. 2020. Wuhan University, LIES5MARS, p. 22.

Guan, Xuefeng, van Oosterom, Peter, Cheng, Bo, 2018. A parallel N-dimensional space-filling curve library and its application in massive point cloud management. ISPRS Int. J. Geo-Inf. 7 (8), 19. http://dx.doi.org/10.3390/ijgi7080327.

Guan, Xuefeng, Wu, H., 2010. Leveraging the power of multi-core platforms for large-scale geospatial data processing: Exemplified by generating DEM from massive LiDAR point clouds. Comput. Geosci. 36, 1276–1282.

Hevner, A., Chatterjee, S., 2010. Design Research in Information Systems. Springer US.

Jaffer, A., 2014. Recurrence for Pandimensional Space-Filling Functions, Vol. 2014. MIT, Boston, US, p. 19, arXiv preprint arXiv:1402.1807.

Jones, C.B., Kidner, D.B., Luo, L.Q., Bundy, G.L., Ware, J.M., 1996. Database design for a multi-scale spatial information system. Int. J. Geogra. Inf. Sci. 10, 901–920.

Kilpelainen, T., 1997. Multiple representation and generalisation of geo-databases for topographic maps (Ph.D. thesis). Finnish Geodetic Institute.

Kodde, M., 2010. The art of collecting and disseminating point clouds. In: van Oosterom, P., Vosselman, G., van Dijk, T., Uitentuis, M. (Eds.), Management of Massive Point Cloud Data: Wet and Dry. Nederlandse Commissie voor Geodesie.

Liu, Haicheng, van Oosterom, Peter, Meijers, Martijn, Guan, Xuefeng, Verbree, Edward, Horhammer, Mike, 2020a. HistSFC: Optimization for nd massive spatial points querying. Int. J. Database Manag. Syst. (IJDMS) 12 (3), 7–28. http://dx.doi.org/10.5121/ijdms.2020.12302.

Liu, Haicheng, Thompson, Rodney, van Oosterom, Peter, Meijers, Martijn, 2021. Executing convex polytope queries on nD point clouds. Int. J. Appl. Earth Observ. Geoinf. 105 (102625), 1–11. http://dx.doi.org/10.1016/j.jag.2021.102625.

Liu, H., Van Oosterom, P., Meijers, M., Verbree, E., 2020b. An optimized SFC approach for nD window querying on point clouds. ISPRS Ann. Photogramm. Remote Sensing Spatial Inf. Sci. 6 (4/W1).

van der Maaden, Jippe, 2019. Vario-scale visualization of the AHN2 point cloud. Geomatics, Delft University of Technology, p. 109.

Markus, Thorsten, Neumann, Tom, Martino, Anthony, Abdalati, Waleed, Brunt, Kelly, Csatho, Beata, Farrell, Sinead, Fricker, Helen, Gardner, Alex, Harding, David, Jasinski, Michael, Kwok, Ron, Magruder, Lori, Lubin, Dan, Luthcke, Scott, Morison, James, Nelson, Ross, Neuenschwander, Amy, Palm, Stephen, Popescu, Sorin, Shum, CK, Schutz, Bob E., Smith, Benjamin, Yang, Yuekui, Zwally, Jay, 2017. The ice, cloud, and land elevation satellite-2 (icesat-2): Science requirements, concept, and implementation. Remote Sens. Environ. (ISSN: 0034-4257) 190, 260–273. http://dx.doi.org/10.1016/j.rse.2016.12.029.

Martinez-Rubi, Oscar, Verhoeven, Stefan, van Meersbergen, Maarten, Schütz, Markus, van Oosterom, Peter, Gonçalves, Romulo, Tijssen, Theo, 2015. Taming the beast: Free and open-source massive point cloud web visualization. In: Capturing Reality Forum 2015, Salzburg. p. 12.

Masó, Joan, Pomakis, Keith, Julià, Núria, 2010. Web Map Tile Service Implementation Standard, OGC 07-057r7, Vol. 2010. Open Geospatial Consortium Inc., p. 129.

Meijers, M., van Oosterom, P.J.M., 2011. The space-scale cube: an integrated model for 2D polygonal areas and scale. In: Fendel, E.M., Ledoux, H., Rumor, M., Zlatanova, S. (Eds.), ISPRS Archives Volume XXXVIII-4/C21, 28th Urban Data Management Symposium. Delft, pp. 95–101. http://dx.doi.org/10.5194/isprsarchives-xxxviii-4-c21-95-2011.

Meijers, Martijn, van Oosterom, Peter, Driel, Mattijs, Šuba, Radan, 2020. Web-based dissemination of continuously generalized space-scale cube data for smooth user interaction. Int. J. Cartogra. 6 (1), 152–176. http://dx.doi.org/10.1080/23729333.2019.1705144.

Memon, M.S., Riedel, M., Janetzko, F., Demeler, B., Gorbet, G., Marru, S., Grimshaw, A., Gunathilake, L., Singh, R., Attig, N., Lippert, Th., 2014. Advancements of the UltraScan scientific gateway for open standards-based cyberinfrastructures. Concurr. Comput. Pract. Experi. J. 26 (13), 2280–2291.

Microsoft, 2014. Patent US 20140198097, Continuous and dynamic level of detail for efficient point cloud object rendering, Vol. 2014. Patent US Office, p. 129.

Nivarti, G.V., Salehi, M.M., Bushe, W.K., 2015. A mesh partitioning algorithm for preserving spatial locality in arbitrary geometries. J. Comput. Phys. 281, 352–364.

Oosterom, Peter van, 1999. Spatial access methods, chapter T2.3. In: Longley, Goodchild, Maguire, Rhind (Eds.), Geographical Information Systems Principles, Technical Issues, Management Issues, and Applications, Vol.1. Wiley, pp. 385–400.

Oosterom, Peter van, 2019. From discrete to continuous levels of detail for managing nD-PointClouds. Keynote presentation at the Geospatial Week 2019, June 2019, Enschede, The Netherlands URL http://nd-pc.org/documents/vario-nD-PC-v7.pdf.

Oosterom, Peter van, Martinez-Rubi, Oscar, Ivanova, Milena, Horhammer, Mike, Geringer, Daniel, Ravada, Siva, Tijssen, Theo, Kodde, Martin, Gonçalves, Romulo, 2015. Massive point cloud data management: Design, implementation and execution of a point cloud benchmark. Comput. Graph. 49, 92–125.

Oosterom, Peter van, Martinez-Rubi, Oscar, Tijssen, Theo, Gonçalves, Romulo, 2016. Realistic benchmarks for point cloud data management systems. In: Abdul-Rahman, Alias (Ed.), Advances in 3D Geoinformation. pp. 1–30. http://dx.doi.org/10.1007/978-3-319-25691-7_1.

Oosterom, Peter van, Meijers, Martijn, 2014. Vario-scale data structures supporting smooth zoom and progressive transfer of 2D and 3D data. Int. J. Geogr. Inf. Sci. 28 (3), 455–478. http://dx.doi.org/10.1080/13658816.2013.809724.

Psomadaki, Stella, Oosterom, Peter van, Tijssen, Theo, Baart, Fedor, 2016. Using a space filling curve approach for the management of dynamic point clouds. In: Dimopoulou, E., van Oosterom, P. (Eds.), ISPRS Annals Volume IV-2/W1, 11th 3D Geoinfo Conference. Athens, pp. 107–118. http://dx.doi.org/10.5194/isprs-annals-iv-2-w1-107-2016.

Ravada, Siva, Horhammer, Mike, Kazar, Baris, 2014. Point Cloud: Storage, Loading, and Visualization. Oracle, US.

Sahr, K., White, D., Kimerling, A.J., 2003. Geodesic discrete global grid systems. Cartogr. Geogr. Inf. Sci. 30 (2), 121–134.

Samet, H., 2006. The Morgan Kaufmann Series in Computer Graphics.

Schütz, Markus, Krösl, Katharina, Wimmer, Michael, 2019. Real-time continuous level of detail rendering of point clouds. In: 2019 IEEE Conference on Virtual Reality and 3D User Interfaces. VR, IEEE, pp. 103–110.

Schütz, Markus, Wimmer, Michael, 2015. Rendering large point clouds in web browsers. In: Proceedings of CESCG 2015: The 19th Central European Seminar on Computer Graphics. Vienna University of Technology. Institute of Computer Graphics and Algorithms, pp. 83–90.

Sirdeshmukh, Neeraj, Verbree, Edward, Oosterom, Peter van, Psomadaki, Stella, Kodde, Martin, 2019. Utilizing a discrete global grid system for handling point clouds with varying locations, times, and levels of detail. Cartographica 51 (1), 4–15. http://dx.doi.org/10.3138/cart.54.1.2018-0009.

Wijga-Hoefsloot, Martine, 2012. Point Clouds in a Database. Delft University of Technology, p. 70.

Xia, X., Liang, Q., 2016. A GPU-accelerated smoothed particle hydrodynamics (SPH) model for the shallow water equations. Environ. Model. Softw. 75, 28–43.

Zhang, Liyao, 2020. Visualization of Point Cloud Models in Mobile Augmented Reality Using Continuous Level of Detail Method. Geomatics, Delft University of Technology, p. 75.

Zhang, Liyao, Oosterom, Peter van, Liu, Haicheng, 2020. Visualization of point cloud models in mobile augmented reality using continuous level of detail method. In: ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XLIV-4/W1. Copernicus GmbH, 15th 3D GeoInfo Conference, London, pp. 167–170. http://dx.doi.org/10.5194/isprs-archives-XLIV-4-W1-2020-167-2020.