Fundamentals, implementations and experimental benchmarks of nD-polytype queries on point cloud data sets

Liu, Haicheng; Thompson, Rodney; Oosterom, Peter van; Meijers, Martijn

Publication date
2021

Document Version
Final published version

Citation (APA)
Liu, H., Thompson, R., Oosterom, P. V., & Meijers, M. (2021). *Fundamentals, implementations and experimental benchmarks of nD-polytype queries on point cloud data sets*. (GISt Report; No. 78). http://www.gdmc.nl/publications/reports/GISt78.pdf

Fundamentals, implementations and experimental benchmarks of nD-polytype queries on point cloud data sets

Haicheng Liu, Rodney Thompson, Peter van Oosterom, Martijn Meijers

August, 2021
GISt Report No. 78

Fundamentals, implementations and experimental benchmarks of nD-polytype queries on point cloud data sets

Haicheng Liu, Rodney Thompson, Peter van Oosterom, Martijn Meijers

August, 2021
Chair GIS technology
Faculty of Architecture and the Built Environment
Julianalaan 134, 2628 BL Delft
Tel. (015) 22786950
E-mail: P.J.M.vanOosterom@tudelft.nl
http://gdmc.nl
ISSN: 1569-0245
ISBN: 978-90-77029-46-6

Fundamentals, implementations and experimental benchmarks of nD-polytype queries on point cloud data sets

Haicheng Liu, Rodney Thompson, Peter van Oosterom, Martijn Meijers
TU Delft
h.liu-6@tudelft.nl

August 24, 2021

Abstract

As an extension to 2D polygonal queries, the nD-polytope queries on point clouds also play a crucial role in nD GIS applications such as the perspective view selection. This report first defines the nD-polytope mathematically, and then develops an efficient nD-polytope querying solution by extending an index-organized table (IOT) approach. The solution integrates four novel intersection algorithms including CPLEX, SWEEP, SPHERE and VERTEX, each of which can be used to realize the primary filtering for polytope querying. The performance of these algorithms is then measured and compared using an representative nD-simplex and an nD-prism query region, respectively. It turns out that SWEEP performs the best over all, but it may degrade significantly as dimensionality goes up. On the other hand, the linear programming algorithm CPLEX although takes more time on intersection computation, performs more stable. Besides, the experiments also reveal that the properties of a same geometry can change significantly across different dimensionality, and thus optimal strategies developed in 2D/3D may not be applicable in high dimensional spaces.

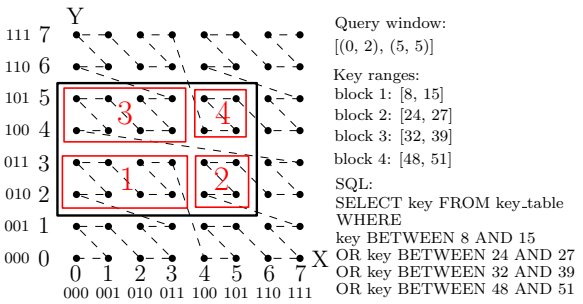
1 Introduction

With respect to range queries on point clouds, the polytope query — which is known as polygonal queries in 2D — also plays a significant role in nD applications, e.g., perspective view selection which forms the basis for visualizing point clouds. Besides, the scalar product queries can also be resolved by using the polytope query to approximate [4, 6].

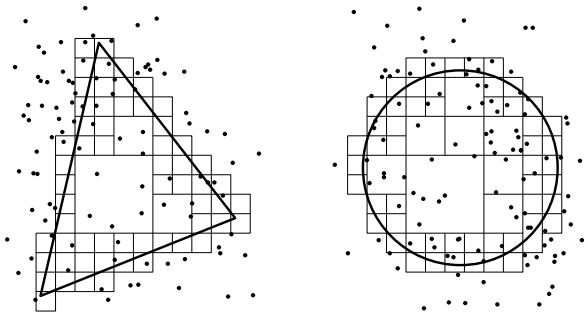
However, we lack efficient solutions. Researchers studying geometric algorithms mainly propose and analyze different polytope querying algorithms theoretically [2, 8, 1] using in-memory data structures. These algorithms are difficult to implement and may not be applicable to address big point data [4]. Based on the R-tree, a "simple" method and a clipping method are developed in [3]. However, they discussed little about the performance in nD space, beyond testing a uniformly distributed 5D data set from the business domain. None of the algorithms proposed distinguishes the type of intersection (inside or touching), leading to redundant intersection computation for a branch block totally inside the polytope.

Our recent study [7] developed an index-organized table (IOT) approach to execute nD orthogonal window queries on points efficiently. IOT first converts each point into a one-dimensional key using a Space Filling Curve (SFC), and then organizes and indexes these keys with a B+-tree. When querying, IOT converts the nD query window into a set of ranges of the one-dimensional key, and then selects the data (Figure 1a). Apparently, this strategy can be extended to more query geometries which are then approximated and represented by a set of SFC ranges (Figure 1b).

Following this idea, this report develops novel algorithms and integrates them into IOT to solve the nD-polytope querying problem. The performance of the integrated framework is tested and analyzed. The rest of the paper is as follows: Section 2 revisits the IOT approach. Section 3 first defines the nD-polytope mathematically, and then devises four different intersection algorithms that can be used to realize polytope querying. Section 4 establishes 2 query geometries including an nD-simplex and an nD-prism, and use them to test the performance of the algorithms. Section 5 concludes the report with further discussion and reflection.



(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

Figure 1: Recursively partitioning the extent of data according to SFC regions to match different query geometries, for selecting data in IOT

2 IOT approach

This section presents the overall architecture of the approach based on [7]. This forms the basis to realize polytope querying described in Section 3.

2.1 Nomenclature

The key terminology used to illustrate the approach is introduced below.

Dimension

Unlike general vectors in machine learning, the dimensions discussed here possess either physical or semantic meanings that humans can perceive and interact with. 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* which are not frequently used in the SQL WHERE clause are affiliated, such as color and intensity, and they are irrelevant for data indexing. These two types of dimensions are not fixed, and may be varied depending on applications.

Hypercube, node and range

In general, a cube refers to a 3D box with equal edge length. We extend such a geometric concept to the nD space which then becomes the hypercube.

Figure 2 illustrates the node and the range. All 2D points to be managed have integer coordinates, and cost 3 bits for each dimension. Thus, by interleaving X bits and Y bits, we can get a 6-bit Morton key for each point. The right sub-figure presents Morton keys of four points inside the region [0, 2; 1, 3]. By truncating the last n bits (, $n = 2$), we get the same value 0010 which corresponds to the Morton key at an upper level. By truncating once more, we derive the Morton key 00 at a higher level. In this way, the Morton keys are formed into a hierarchy which is actually a Quadtree structure. We can easily extend this structure to higher dimensional spaces so that a Morton node refers to the corresponding 2^n-tree node embedded in the Morton hierarchy. A branch node indexes the nodes on the level below, and represents the extent of a hypercubic region (, e.g., a block in the Quadtree). Thus, the branch node is also a range of Morton codes starting from the lower-left corner to the upper-right. A leaf node is a point in the data. So, the upper and lower bounds of its range both equal the Morton key.

2.2 Basic settings

Figure 3 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. Such a full resolution key can be decoded directly to give the original coordinates. Then, the approach utilizes Index-Organized Tables [9] which adopt the B+-tree to manage points. Hence, the abstract model is a simple flat table, with each record presenting an nD-point. However, the approach integrates the indexing structure and the data table together, which differs from previous solutions that

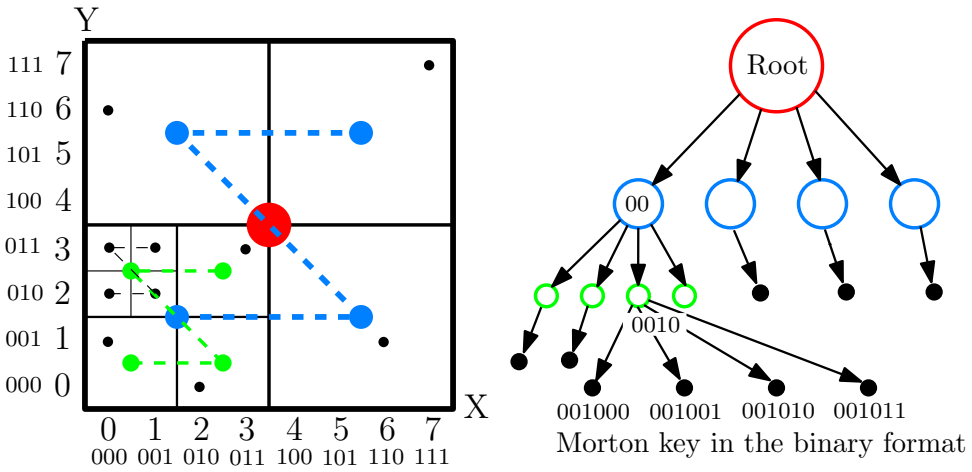


Figure 2: 2D Morton hierarchy forms into a Quadtree structure: black dots are real points to be managed, while colored dots are Morton branch nodes at different levels

separate them. In this way, the storage becomes more compact, and data retrieval using indexes becomes more direct and efficient.

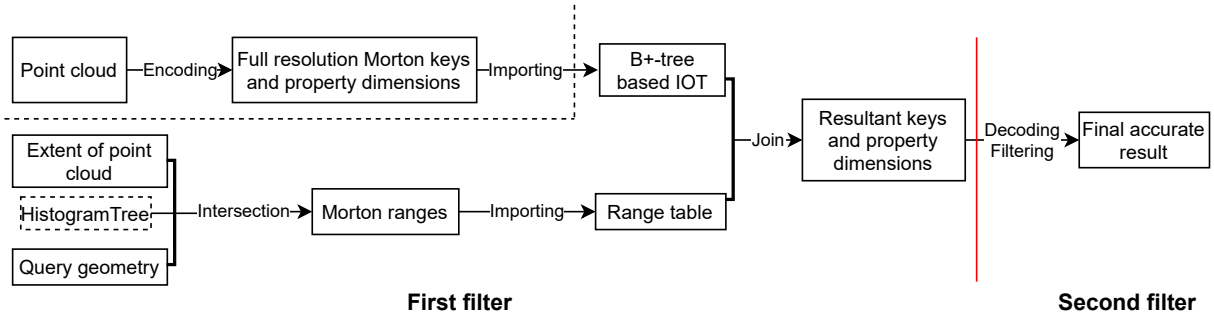


Figure 3: The loading and querying procedure of the IOT approach

As to querying, the approach adopts two filters. The first filter uses the Morton hierarchy to approximate the query window and derive the ranges. Take Figure 1a to illustrate: the first filter starts by examining whether the root node (, i.e., the overall extent of the data) intersects the query window. If they intersect, the root node will be decomposed into 4 sub-nodes and the spatial relationship between each node and the query window will be assessed again. During the range computing process, if a node is inside the query window, the range will be exported directly without further decomposition. Near the query boundary, the decomposition goes on recursively until the maximum depth defined. A large depth would cause huge number of ranges generated, slowing down the first filter. A small depth however results in rough query result, which is also unacceptable because it moves the burden to a second filter for an accurate answer. Consequently, we set a threshold on the maximum number of ranges to confine the search depth. After the searching reaches this threshold, the first filter exports all ranges into a range table, and then joins it with the IOT for selection, where the index automatically functions. A second filtering will be conducted in a following step to complete the query.

However, a drawback of this querying process is that part of the ranges generated may actually contain few or even no points when the data distribution is skewed. An example is the Airborne Laser Scanning (ALS) point data, where most points lie on the ground, with few above the average height of buildings. For one thing, this drawback implies extra time is spent on keeping refining nodes containing few points; for another, large quantities of non-dense and empty ranges add the memory cost. We thus propose using HistogramTree to improve the quality of ranges and thereafter the whole querying performance. As Figure 4 shows, HistogramTree counts the points of the nodes at each level of the Morton hierarchy. If the number exceeds the threshold of the tree, the node is partitioned. It should be noted that HistogramTree contains neither points nor pointers to points. Thus, it is a compact structure which can be stored in a flat table.

That is to say, the approach previously only uses the principle of the Morton hierarchy to generate

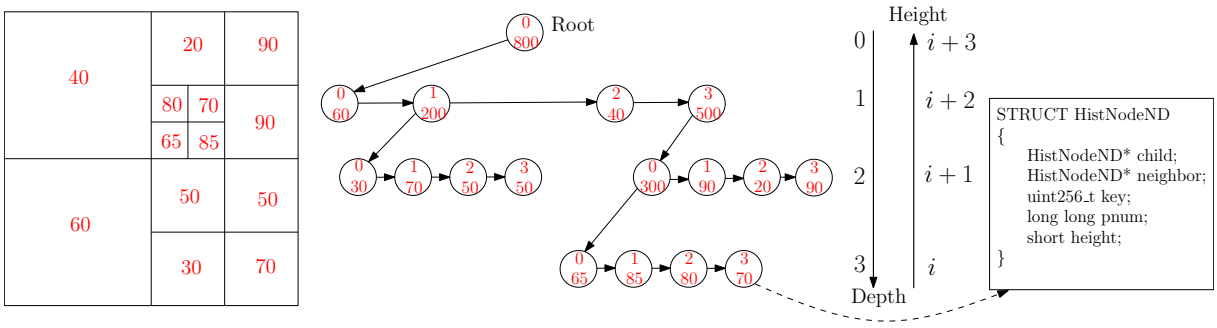


Figure 4: A 2D HistogramTree example, where the threshold is 100; left: point counting, middle: pointer structure of HistogramTree, with each node storing a Morton key and number of points, right: structure of a HistogramTree node

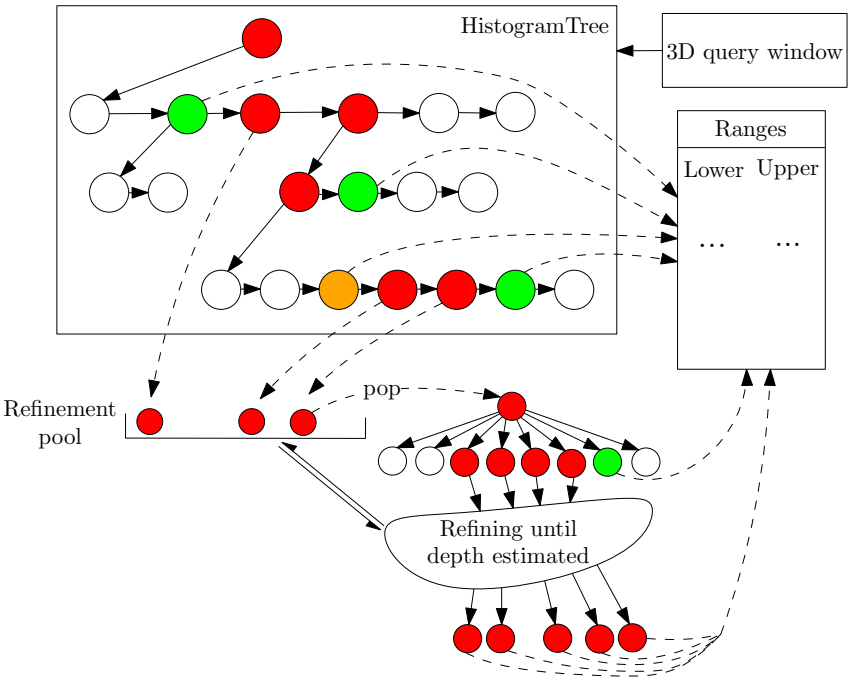


Figure 5: Range generation using a 3D HistogramTree: wither respect to the query window, green nodes are inside; red nodes are on the boundary; organge nodes are on the boundary but with few points; white nodes are outside

ranges, while now it employs HistogramTree, an explicit and simplified representation of the Morton hierarchy, together with the knowledge of data distribution to compute more effective ranges. Figure 5 presents the querying procedure. Starting from the root node, by performing intersection between the HistogramTree and the query window iteratively, the function retrieves all relevant nodes to build the range table. Non-overlapping nodes are abandoned. Nodes which are inside the query window are immediately added to the range table with no further processing needed. The nodes on the boundary with few points inside are also exported immediately. The remaining nodes intersect the boundary of the query window and are temporarily held in a refinement pool. These can be further refined based on fixed recursive decomposition. The process stops when the refinement pool is empty or the number of ranges reaches the threshold. The rest of querying remains the same as before.

Apparently, this approach is smarter and more flexible than the conventional 2^n-tree index which stores pointers to point blocks: a branch node totally inside the query geometry can be processed immediately; point distributions are considered so that partial refinement is possible; there is no block unpacking process.

The theoretical querying time of the IOT approach is as follows:

$$T = T_{pre} + T_{io} + T_{post} \tag{1}$$

where T_{pre} is the time cost of the first filter, and mainly comprises range computation and B+-tree traversal; T_{io} indicates the main I/O cost to retrieve points inside the ranges; T_{post} refers to the final decoding and filtering. T_{pre} is bounded by $\mathcal{O}(r \log_B N)$, where r is the number of ranges generated; B is page capacity in the number of points; N represents the input size. T_{io} maximally covers $\mathcal{O}(\frac{k'}{B} + r)$ I/Os, where k' equals the number of points returned by the first filter. In fact, the more accurate expression should be $\mathcal{O}(\sum_{i=1}^{r} \lceil \frac{k_i}{B} \rceil)$, where k_i represents the number of points inside a specific range, and $\sum_{i=1}^{r} k_i = k'$. T_{post} is bounded by $\mathcal{O}(k')$. Once parallelism is applied, T_{post} becomes $\mathcal{O}(\frac{k'}{p})$, given p processors.

3 Polytope querying

A convex polytope is simply a convex hypervolume. It is defined as an nD region for which, given any 2 points within the region, every point along a straight line joining the points is also within the region. To use the polytope practically, this section first provides the mathematical definition. Then, novel algorithms for polytope querying based on the IOT approach are developed.

3.1 Mathematical definition

A half-space is a division of space along a hyperplane (Figure 6). Here it is defined as the set of points $\mathbf{x}$ such that $\boldsymbol{\omega} \cdot \mathbf{x} + \beta \leq 0$. Note that the inequality is not as will be found in some texts, but is used here for compatibility with the conventions of computer representation software (3D) that the normal vector $\boldsymbol{\omega}$ is oriented so that it points to the outside of the solid object. We also here require that $\boldsymbol{\omega}$ is a unit vector ($\boldsymbol{\omega} \cdot \boldsymbol{\omega} = 1$). A half-space can be denoted by the tuple ($\boldsymbol{\omega}$, β).

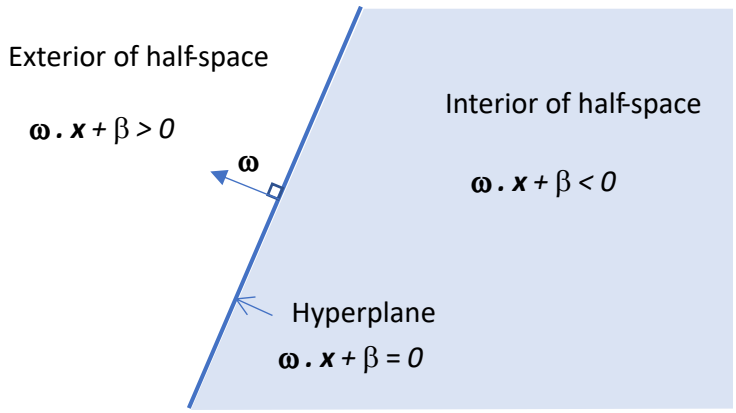


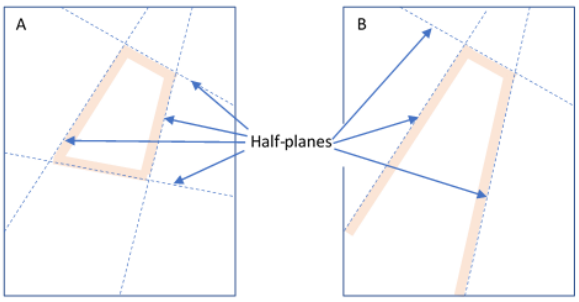
Figure 6: The definition of a 2D half-space which can be generalised to nD

In 2D space, the term half-plane is sometimes used, and it is defined by an infinite straight line — its only boundary. In 3D space, the half-space is defined and bounded by an infinite plane, while in higher dimensions the half-space is defined as all points on a particular side of an $(n-1)$D hyperplane. In all cases, the dividing $(n-1)$D hyperplane has the definition $\boldsymbol{\omega} \cdot \mathbf{x} + \beta = 0$. A convex polytope is defined as the intersection of a finite set of half-spaces, where it is not necessary that the boundaries are closed (Figure 7):

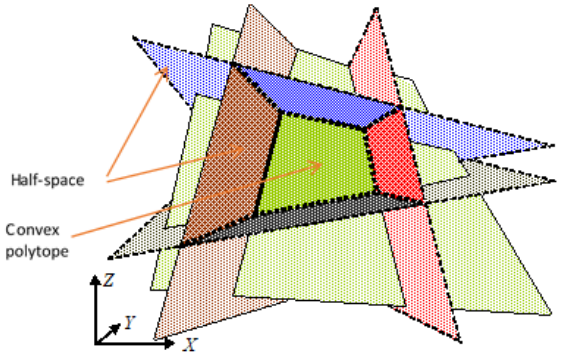
$$C = \cap_{i=1}^{m} H_i$$

where H_i is a set of m half-spaces.

Based on this formulation, we can simply test whether a point is in a half-space by evaluating the value of $\boldsymbol{\omega} \cdot \mathbf{p} + \beta$: if the value is non-positive, the point $\mathbf{p}$ is within the half space. Since $\boldsymbol{\omega}$ and $\mathbf{p}$ are vectors of length n, the operation per point costs O(n) time. Then, computing the relationship between the point and the convex polytope, can maximally cost O(mn) time per point. However, globally traversing all the points for selection is too costly and scales badly with the size of input. Consequently, we adopt the IOT approach to speed up the search.



(a) 2D convex polytopes defined by half-planes. A: a completely bounded convex polytope, B: a convex polytope bounded on three sides only



(b) A 3D convex polytope, from [10]

Figure 7: Convex plytopes in 2D and 3D spaces

3.2 Querying algorithms

The core of the querying algorithm lies in the intersection computation between HistogramTree nodes (or generic Morton nodes) and the convex polytope to generate ranges. Then, all algorithms join the ranges with the IOT (Section 2.2), with a final filter performing the aforementioned point-in-polytope test.

3.2.1 Linear programming method

The rigorous linear programming method detects intersection by finding solutions for a set of equations defined by the $(n-1)$D hyperplanes of the polytope and a node. We realize this by using CPLEX which is a tool developed by IBM to solve linear optimization problems [5]. It provides optimal solutions to an objective function confined by a set of constraints. Using CPLEX, we can create a variable array x (i.e., x_1, x_2, x_3,... x_n), and set their range according to the range of a node (, i.e., $L_1 \leq x_1 \leq U_1$, $L_2 \leq x_2 \leq U_2$, ... $L_n \leq x_n \leq U_n$). Then, we convert all half-spaces to constraints in the form of $\boldsymbol{\omega} \cdot \mathbf{x} + \beta \leq 0$. We set the objective function of the linear model to 0, meaning that once a solution found, the program will stop. In this way, CPLEX detects whether an intersection happens.

3.2.2 SWEEP

SWEEP first identifies the "entry" and "exit" of a node with respect to a half-space (Figure 8): imagine if the half-space were to be moved from a great distance away, towards and across the node so that ultimately the node is within the half-space; the entry and exit are the first and last vertices to cross the boundary. The categorization as entry and exit is only true in relation to a single half-space, and must be re-appraised for others. Then, based on the distance between the half-space's boundary and the entry or the exit, SWEEP determines if an intersection happens. Figure 9 presents the whole workflow of SWEEP.

Figure 10 shows how SWEEP works with HistogramTree. Node N_1 is not within the half-space H_2, and is therefore external to the polytope C, and does not need to be further processed. N_2 is within all of H_1 to H_4, and its range can be exported. In the case of N_3, since it fulfils neither of these cases, it must be further processed before being accepted or rejected. The case of N_4 is significant, because it partially overlaps or falls within each half-space, but in fact it does not intersect C. We refer to this case as a False Positive Node (FPN) and will discuss later. In the process of searching the nodes from the refinement pool, nodes at the next lower levels are processed (2^n of them). These are then applied to the same tests against C. Some sub-nodes are found to be internal, some to be on the boundary, and the rest external.

FPNs exist at crossings of half-spaces (, e.g., N_4 in Figure 10a and N_{42} in Figure 10b). It is a practical proposition to ignore the problem, and allow FPNs to be processed as if they are true positive nodes. In Figure 10, N_4, after first refinement, has three of its sub-nodes eliminated, leaving only N_{42} whose sub-nodes are eliminated at the next level. This appears to be a common event — that as a FPN is decomposed into sub-nodes at one level below, those sub-nodes are largely eliminated. Together with the second filter, points in the remaining FPNs will all be filtered out.

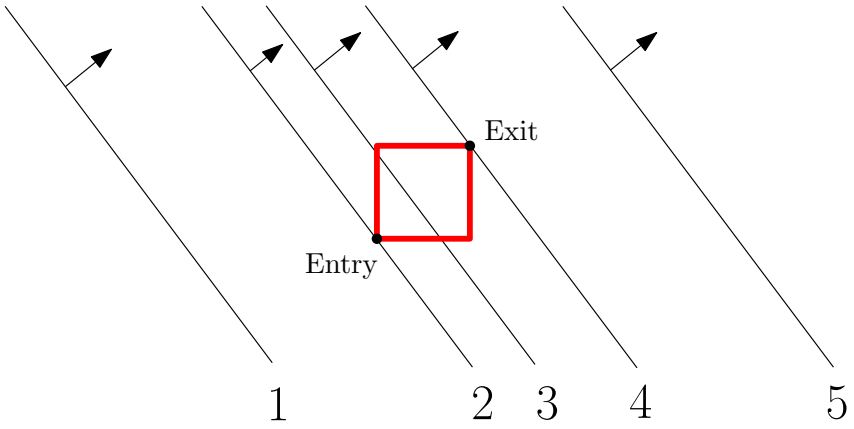


Figure 8: The "sweeping" process: in (1), a half-space starts sweeping, and the node is outside; in (2), (3) and (4), the half-space sweeps over the node, where intersection happens; in (5), the sweeping ends, and the node is fully inside the half-space

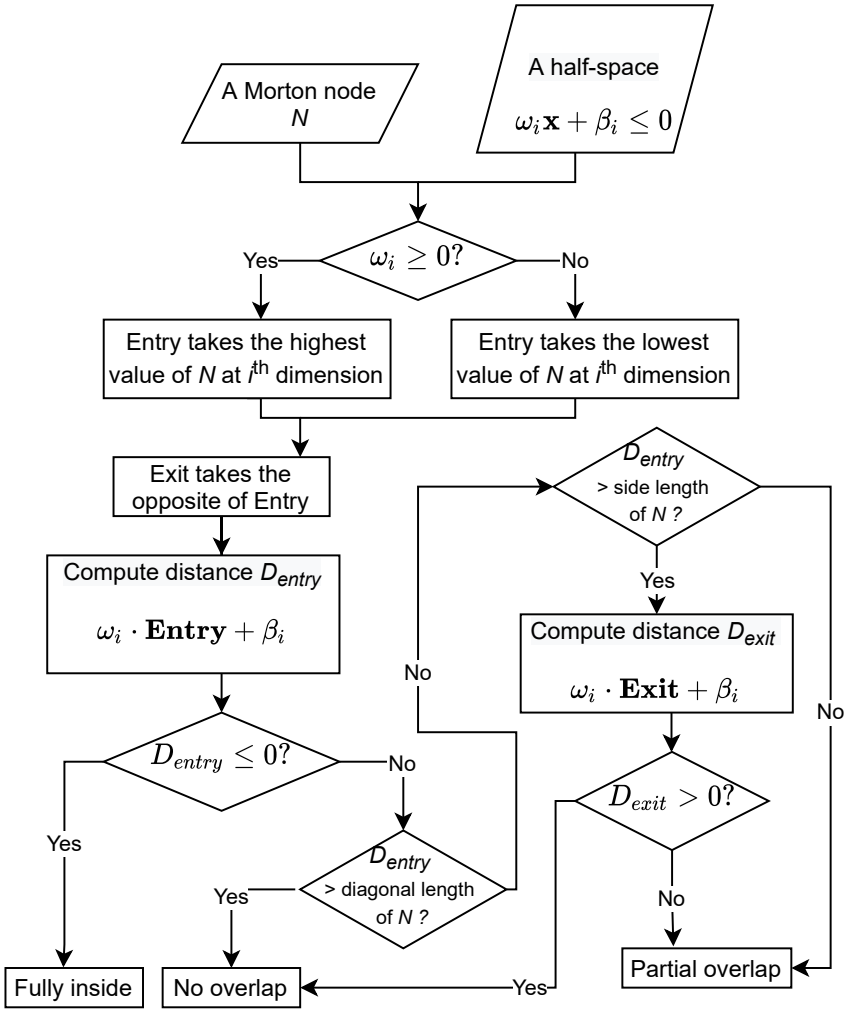
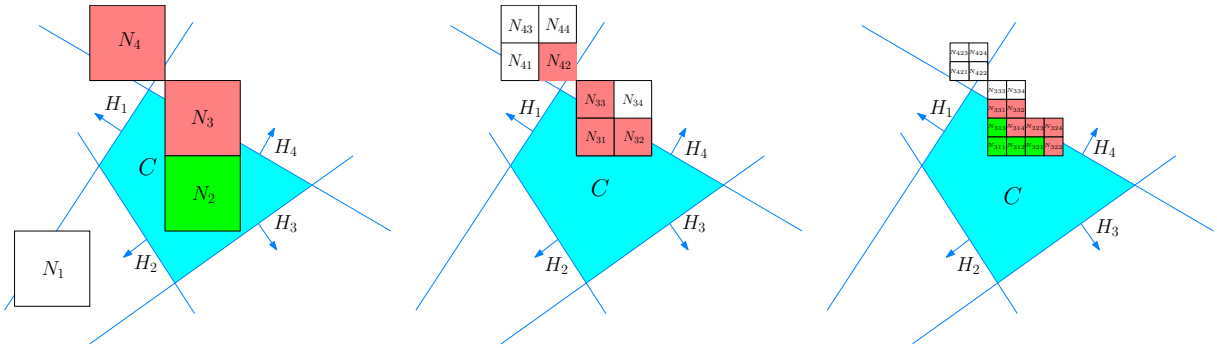


Figure 9: The workflow of SWEEP

3.2.3 SPHERE and VERTEX

SPHERE and VERTEX are two alternatives. They also detect intersection by examining the relationship between a node and all half-spaces.

The SPHERE algorithm first computes the centre of a node. If the centre is in the half-space, or the Euclidean distance between the centre and the half-space is within half of the diagonal length of



(a) HistogramTree leaf nodes (b) Refining once (c) Refining twice

Figure 10: SWEEP selection based on the IOT approach, with white nodes outside, green nodes inside and red nodes on the boundary

the node, then the node will be selected. "inside" or "partial overlap" can be decided depending on the distance. SPHERE only needs a central point for intersection detection, which is favorable. However, as the distance computed is an upper boundary, FPNs will be selected (N_4 in Figure 11).

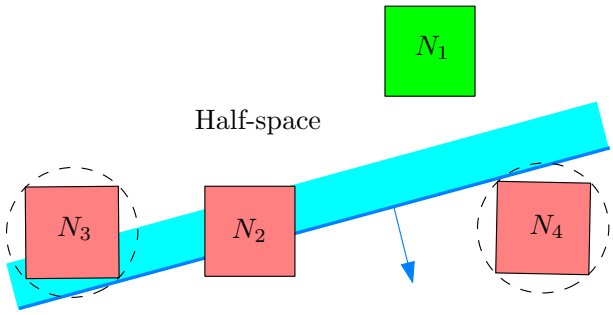


Figure 11: Intersection detection using SPHERE: N_1 is inside; N_2 partially overlaps the half-space; N_3 and N_4 are falsely detected as partial overlap

The VERTEX algorithm is more straightforward, as it examines every vertex of a node to determine whether the node intersects a half-space. If all vertices are outside, then no intersection happens. If all the vertices are in the half-space, the node is inside. For all other cases, a partial overlap is returned. Implementing VERTEX is simple, but the algorithm may degrade drastically in high dimensional spaces. This is because the number of vertices of a node grows exponentially as dimensionality goes up. In addition, false detection will also arise at crossings of half-spaces (, e.g., Figure 10a), which is the same as SWEEP.

3.3 Theoretical complexity

Since these algorithms are all based on the IOT approach, Equation 1 still applies. However, T_{pre} and T_{post} become $\mathcal{O}(mnr \log_B N)$ and $\mathcal{O}(mnk')$. In VERTEX, every vertex of a node has to be examined. Thus, its T_{pre} is bounded by $\mathcal{O}(2^n mnr \log_B N)$. Besides, all algorithms introduce FPNs except CPLEX. So, k' can be varied. To determine the optimal solution in terms of time cost, we should consider all processes as a whole. An accurate first filter may cost more time, but it returns smaller k' which alleviates I/O and post-processing in the second filter. For this purpose, we introduce False Positive Rate (FPR) to indicate I/O and the performance of second filter (Equation 2):

$$FPR = \left| \frac{k' - k}{k} \right| \tag{2}$$

where k is the accurate answer. Sometimes, FPR can be very large, e.g., in high dimensional spaces. Then, the portion of data selected by the first filter is also indicative (Equation 3):

$$selectivity = \frac{k'}{N} \tag{3}$$

The memory cost is mainly determined by the maximum number of ranges for querying and k'.

4 Experiments and analysis

This section describes experiments to evaluate the performance of different algorithms described above. Section 4.2 builds a generic regular nD-simplex model and an nD-prism model for testing. The nD-simplex is the simplest nD polytope which can constitute any other nD-complexes, while the nD-prism is devised to investigate how the number of half-spaces influences the querying efficiency exactly.

We conduct all experiments on "pakhuis" server: a HP DL380p Gen8 server with 2 × 8-core Intel Xeon processors, E5-2690 at 2.9 GHz, 128 GB of main memory, a RHEL6 operating system. The disk storage is a 41 TB SATA 7200 rpm in RAID6 configuration. All tests are "cold", without any caching processes.

4.1 Data sets

To get a generic testing result, we use synthetic data for these two tests. We apply uniform distribution to create every dimension independently. Each dimension uses 12 bits to store its value, so the largest 10D Morton key will occupy 120 bits which is just below the maximum of the Oracle NUMBER type (128 bits). Thus, the value of each dimension is between 0 and 4095. This allows limited number of unique pairs to be created in 2D. So, we only generate 10^4 2D points, while generate another 10^6, 10^7, 10^8 and 10^{10} points for 4D, 6D, 8D and 10D data sets, respectively. Since the data is uniformly distributed, we built the IOT approaches without HistogramTree. With all these data sets, we are able to investigate the querying scalability with respect to dimensionality.

4.2 The regular nD-Simplex query

In the following, we first build the simplex model for querying at different dimensionality. Then, we perform the test and discuss the results.

4.2.1 Query geometry construction

An nD-simplex has $n+1$ vertices, from v_0 to v_n, where $v_i = (\nu_{i0}, \nu_{i1}, \ldots, \nu_{i(n-1)})$. We use a unit vector along each axis to represent a vertex so that we can get the first n vertices of the simplex (Figure 12), i.e., $v_0 = (1, 0, 0, \ldots)$, $v_1 = (0, 1, 0, \ldots), \ldots, v_{n-1} = (0, 0, \ldots, 1)$. The last vertex v_n must have the same value for every dimension to make a regular nD-simplex. Assuming $v_n = (\varepsilon, \varepsilon, \ldots)$, we calculate it by solving a quadratic equation on the length of (v_i, v_n), $(i < n)$. So, there are two possible solutions, $\varepsilon = \frac{1 \pm \sqrt{n+1}}{n}$. We take $\varepsilon = \frac{1+\sqrt{n+1}}{n}$. We then compute the mean of all vertices: $M = (\mu, \mu, \ldots)$, where $\mu = \frac{1+\varepsilon}{1+n}$. Then, we shift the whole simplex by moving M to the origin O. In this way, the simplex is centralized at O, while the vertex $v_i = (\nu_{i0}, \nu_{i1}, \ldots, \nu_{i(n-1)})$, where $\nu_{ii} = 1 - \mu$ for $i < n$; $\nu_{ij} = -\mu$ for $i \neq j$ *and* $i < n$; $\nu_{nj} = \varepsilon - \mu$ for $j = 0, 1, \ldots, n-1$.

Afterwards, we normalize the vertices to build the normal vectors (, i.e., $\overrightarrow{a}$, $\overrightarrow{b}$ and $\overrightarrow{c}$ in Figure 12) for a new set of faces. These faces constitute another nD-simplex (with dashed green boundaries). As the simulated data (Section 4.1) are positive numbers, we shift the new simplex to the positive zone: by shifting the first n faces (H_0 to H_{n-1}) to pass through O (, except H_n which is opposite to O), we could build such a "positive" regular nD-simplex for querying. The size of the simplex depends on the position of H_n which is adjustable (Figure 12). Hence, half-spaces constituting this regular nD-simplex are derived:

$$
\begin{aligned}
H_0 &: \frac{1-\mu}{A}x_0 + \frac{-\mu}{A}x_1 + \cdots + \frac{-\mu}{A}x_{n-1} \leq 0 \\
H_1 &: \frac{-\mu}{A}x_0 + \frac{1-\mu}{A}x_1 + \cdots + \frac{-\mu}{A}x_{n-1} \leq 0 \\
&\vdots \\
H_n &: \frac{1}{\sqrt{n}}x_0 + \frac{1}{\sqrt{n}}x_1 + \cdots + \frac{1}{\sqrt{n}}x_{n-1} \leq -\beta
\end{aligned}
$$

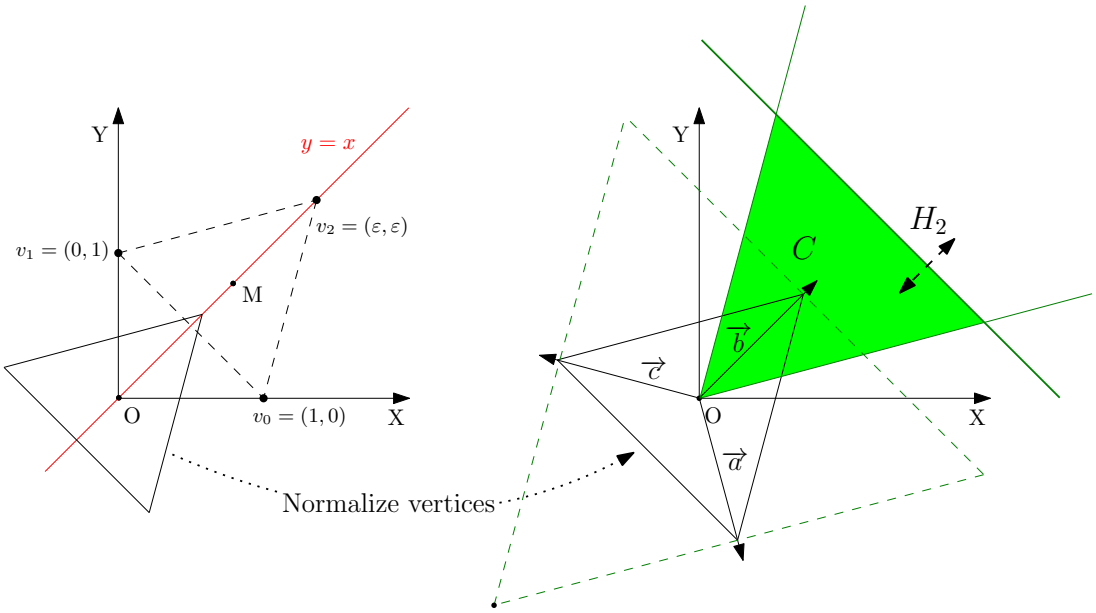


Figure 12: The workflow to build a regular simplex testing model in 2D

where $A = \sqrt{n\mu^2 - 2\mu + 1}$, and $-\beta$ is the distance from O to H_n. To maximize the simplex within the data region, $-\beta$ should be taken as large as possible. Suppose the domain is a unit hypercube with every dimension ranging from 0 to 1. Then, the maximum simplex formulated above leads to an intersection of H_n and H_i $(i < n)$ on the face $x_i = 1$. In other words, when $x_i = 1$, we will get the same expression based on the equation of either H_i or H_n. We just use H_0 and H_n to derive

$$\beta = -\frac{1}{\mu\sqrt{n}} = -\frac{\sqrt{n}}{1 + \frac{1}{\sqrt{n+1}}}$$

However, with such a formulation, all vertices of the simplex are on the boundary faces, and there is no room for SWEEP making false detection at the corners (Figure 10a). So, we raise β to move H_n towards O, which creates some gap between the first n vertices and the boundaries, except the last vertex which coincides with O. Mathematically, this means we adopt

$$\beta = R\sqrt{n} - \frac{1}{\mu\sqrt{n}}, \quad \left(0 \le r \le \frac{\sqrt{n+1}}{1 + \sqrt{n+1}}\right) \tag{4}$$

where R is a ratio indicating the residual gap we created. We call it the residual ratio. A larger R corresponds to larger room for false detection. We multiply R by $\sqrt{n}$ to keep the product a similar decreasing speed as the original β, when n grows.

Based on the formulation above, we can compute the volume of the nD-simplex which indicates the selectivity. Suppose the edge length of the simplex is s, while the height is h, i.e., $-\beta$ in this case, we could derive

$$-\beta = h = s\sqrt{\frac{n+1}{2n}}$$

$$V = \frac{s^n}{n!}\sqrt{\frac{n+1}{2^n}} = \left(-\beta\sqrt{\frac{2n}{n+1}}\right)^n \frac{1}{n!}\sqrt{\frac{n+1}{2^n}} \tag{5}$$

As the test is up to 10D, we substitute $n = 10$ and $R = 0.1$ to Equation 4, and then substitute the resultant β to Equation 5 to compute V: $V_{10} = 0.0010091$. This results in a 1‰ selectivity. Then, we compute R using Equation 6 for other data sets, to achieve the same selectivity:

$$R = \frac{\sqrt{n+1}}{n\sqrt{2}}\left(\frac{n\sqrt{2}}{1 + \sqrt{n+1}} - \left(n!V_{10}\sqrt{\frac{2^n}{n+1}}\right)^{\frac{1}{n}}\right) \tag{6}$$

where $n = 2, 4, 6, 8$. Table 1 shows the computed R. Table 2 presents the ratio between the distance from the origin O to H_n and the diagonal length, i.e., $-\frac{\beta}{\sqrt{n}}$. This ratio indicates how the simplex stretches over the data region as dimensionality rises.

Table 1: Residual ratio for different dimensionality

	2D	4D	6D	8D	10D
R	0.6044127	0.5106438	0.3701990	0.230515	0.1

Table 2: The ratio between the distance from O to H_n and the diagonal length of the simplex

	2D	4D	6D	8D	10D
$-\frac{\beta}{\sqrt{n}}$	0.029562	0.180339	0.355509	0.519485	0.668337

4.2.2 Test setup

The experiment uses 10^6 as the maximum number of ranges for querying. This value on the one hand, guarantees a low FPR in high dimensional spaces; on the other hand, it avoids bloating the memory. CPLEX, SWEEP, SPHERE and VERTEX are tested. We implemented two versions of CPLEX. CPLEX#1 endeavours to distinguish between two types of intersection, i.e., inside or partial overlap, while CPLEX#2 treats all intersections as partial overlap. This means nodes inside the simplex will also be refined iteratively. The test uses a single thread, and records 3 indicators for evaluating the performance:

1. Selectivity of the first filter (Equation 3).
2. Number of iterating cycles (, i.e., for-loops) to generate ranges. A cycle of CPLEX#1 and CPLEX#2 means resolving the optimal problem once (Section 3.2.1). A cycle of SWEEP, SPHERE and VERTEX means computing the distance from a half-space to either the enter or the exit, the center of a node, and a vertex, respectively.
3. Time cost of the first filter and the second filter. The first filter time corresponds to T_{pre} in Equation 1, while the second filter takes $T_{io} + T_{post}$ to accomplish.

4.2.3 Results

Tables 3 - 5 show the results. CPLEX#1 owns the lowest FPR, while SWEEP responses the fastest below 10D with CPLEX#2 fastest in 10D. The low FPR of CPLEX#1 results from its accurate intersection computation, without any FPNs. Consequently, the second filter always receives the smallest k' for post-processing. However, as CPLEX#1 spends significantly more time for each iteration for computing ranges, such advantage is insignificant until 10D (Table 5). CPLEX#2 presents analogous selectivity as CPLEX#1, but is faster. This is because CPLEX#1 decomposes the simplex to individual half-spaces to further compute inside or partial overlap, while ignoring this reduces about 50% computation (Table 4). Besides, it becomes less necessary to distinguish these two intersection types when n increases, as all nodes returned by the first filter tend to fall on the boundary (Table 6). Since the number of nodes at each level of the Morton hierarchy increases exponentially, the final nodes selected mainly reside in higher levels with larger sizes. So, these nodes are more likely to partially intersect the simplex. 2D is an exception as CPLEX#2 keeps refining nodes inside the simplex, while others search to the bottom of the Morton hierarchy.

False positive points selected by CPLEX are caused by boundary nodes intersecting the polytope, as these nodes contains points which are actually outside the polytope. On the other hand, besides the boundary nodes, SWEEP, SPHERE and VERTEX also select FPNs as they apply approximate intersection computation (Table 7). So, larger k' are returned, which causes significant performance degradation in higher dimensional spaces. This makes SWEEP lag behind CPLEX approaches after 8D. SPHERE processes similar number of iterations as SWEEP, and is even faster in generating ranges. However, the ranges generated contain much more false positive points, which greatly undermines SPHERE's overall performance. Especially in 10D, SPHERE nearly selects the whole data set. VERTEX returns the same

Table 3: Selectivity of the first filter

	2D	4D	6D	8D	10D
CPLEX#1	1‰	1.345‰	4.805‰	25.03‰	400.1‰
CPLEX#2	1‰	1.387‰	4.815‰	25.03‰	400.1‰
SWEEP	1‰	1.364‰	9.244‰	164.5‰	605.0‰
SPHERE	1‰	1.386‰	11.93‰	467.1‰	929.5‰
VERTEX	1‰	1.364‰	9.244‰	164.5‰	605.0‰

Table 4: Number of iterating cycles for computing ranges

	2D	4D	6D	8D	10D
CPLEX#1	3 653	4 412 563	4 262 738	4 650 728	6 299 304
CPLEX#2	23 696	1 302 544	1 566 912	2 489 856	3 550 208
SWEEP	2 259	2 223 188	6 451 764	33 842 261	16 336 597
SPHERE	2 243	1 829 719	5 676 139	23 967 213	11 245 251
VERTEX	8 260	28 574 320	316 524 032	1 711 305 472	3 732 959 232

Table 5: Time cost (s) (first filter ; second filter)

	2D	4D	6D	8D	10D
CPLEX#1	0.52 ; 0.001	633.7 ; 0.001	616.8 ; 0.041	660 ; 5.51	845.7 ; 5 502
CPLEX#2	2.078 ; 0.001	120.4 ; 0.001	147.2 ; 0.041	250.2 ; 5.51	360.8 ; 5 502
SWEEP	0.001 ; 0.001	2.129 ; 0.001	3.078 ; 0.082	11.11 ; 31.69	3.388 ; 8 691
SPHERE	0.001 ; 0.001	2.35 ; 0.001	3.331 ; 0.103	9.518 ; 67.32	2.729 ; 11 213
VERTEX	0.001 ; 0.001	2.685 ; 0.001	7.624 ; 0.082	133.8 ; 31.69	434.9 ; 8 691

Table 6: Number of different types of node selected by the first filter (inside; boundary)

	2D	4D	6D	8D	10D
CPLEX#1	323 ; 191	335 290 ; 664 711	78 778 ; 921 227	137 ; 999 941	1 ; 1 000 199
CPLEX#2	0 ; 4 357	0 ; 1 000 003	0 ; 1 000 020	0 ; 1 000 078	0 ; 1 000 200
SWEEP	323 ; 191	333 082 ; 666 930	75 139 ; 924 880	9 ; 1 000 003	1 ; 1 000 074
SPHERE	341 ; 209	367 127 ; 632 884	53 789 ; 946 230	0 ; 1 000 016	0 ; 1 000 210
VERTEX	323 ; 191	333 082 ; 666 930	75 139 ; 924 880	9 ; 1 000 003	1 ; 1 000 074

Table 7: Number of true positive nodes (TPN) and false positive nodes (FPN) selected (TPN; FPN)

	2D	4D	6D	8D	10D
SWEEP	514 ; 0	660 077 ; 6 853	836 021 ; 88 859	370 333 ; 629 670	680 585 ; 319 489
SPHERE	550 ; 0	614 512 ; 18 372	779 259 ; 166 971	230 862 ; 769 154	446 376 ; 553 834

ranges as SWEEP, but takes much more iterations to accomplish due to the excessive vertex-based distance computation. An odd pattern occurs that the iterations of SWEEP and SPHERE decline from 8D to 10D. This is because the simplex's boundary is very close to the boundaries of the data region in 10D. So, the false detection related to this half-space is constrained by the data region and causes less overhead.

In general, the FPRs of all approaches increase drastically as dimensionality goes up, including CPLEX. For one thing, this is because to keep the 1‰ selectivity, the simplex increasingly stretches over the whole data region as n grows (Table 2), so that it intersects an increasing portion of nodes at each height. For another, the maximum number of ranges for querying is set to a constant for all data sets, while a node is decomposed into more children when n increases, which causes nodes mainly at higher levels selected. Thus, approximating the simplex by these nodes becomes less accurate, and introduces more false positive points.

4.3 nD-prism query

In theory, the number of half-spaces constituting the nD-polytope affects the time cost of range generation linearly (Section 3.3). This section uses a simple nD-prism model to verify this. Inscribed regular polygons of a circle are used as the bottom of the nD-prism (Figure 13). To create the prism with $2f$ vertical faces, we apply a rotation formulation: suppose $\theta = \frac{\pi j}{f}$, where $j = -f+1, \ldots, f$. Then, we create each half-space with

$$\boldsymbol{\omega} = (\cos\theta, \sin\theta, 0, 0, \ldots, 0)$$
$$\beta = -\sqrt{\frac{selectivity}{\pi}} \cdot scale - \frac{scale}{2}(cos\theta + \sin\theta)$$

where scale is the length of a dimension which equals 4096 in this case. We do not create half-spaces at the bottom of the prism, as they are implicitly defined by the data region. Based on this formulation, we generate $8i$-gonal ($i \in [1, 8]$) prisms from 2D to 10D. The hyper-volumes of these $8i$-gonal prisms are close to each other, and they approximately equal the product of selectivity 1‰ and the hyper-volume of the data region.

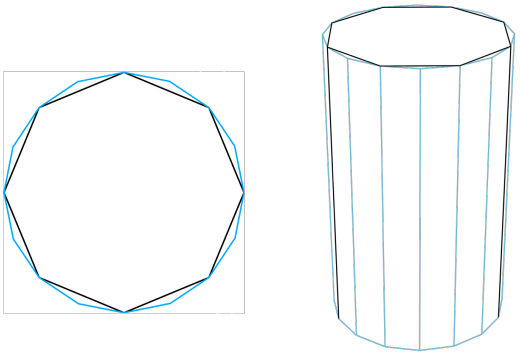


Figure 13: The nD 8-gonal and 16-gonal prisms projected to 2D and 3D spaces

The test still uses the same uniform data sets from 2D to 10D, with a focus on the number of iterations and time cost for computing ranges. The maximum number of ranges for querying is 10^6. The approaches are the same as before. Figures 14 - 23 show the results.

Iterations of range computation

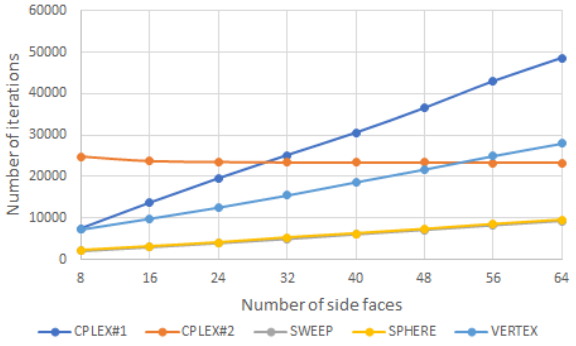


Figure 14: Number of iterations of range computation in the 2D-prism query

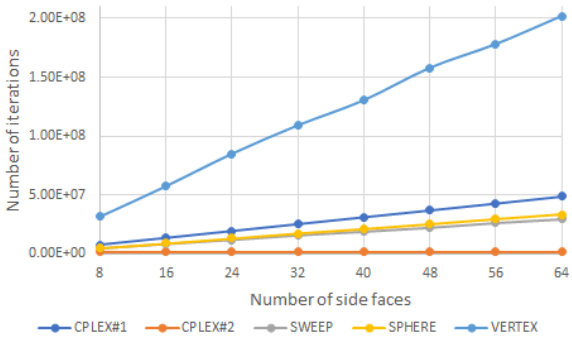
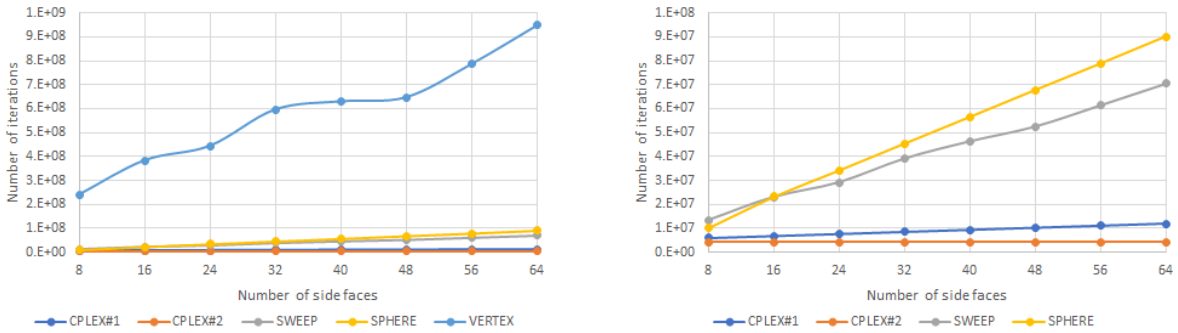
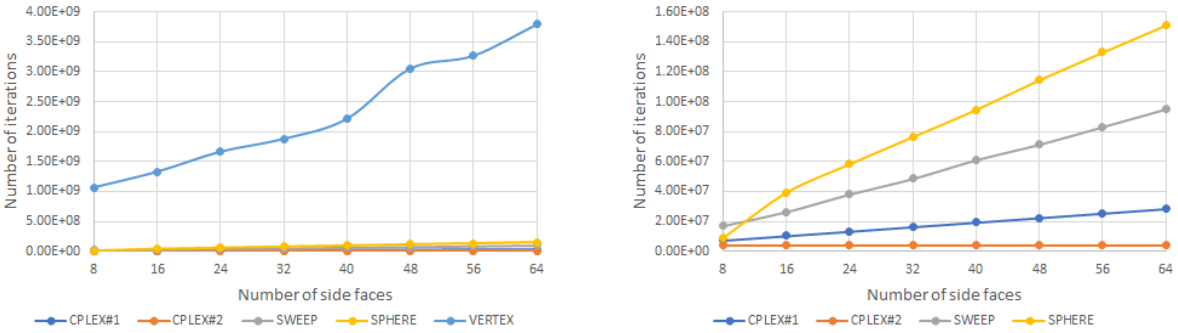


Figure 15: Number of iterations of range computation in the 4D-prism query



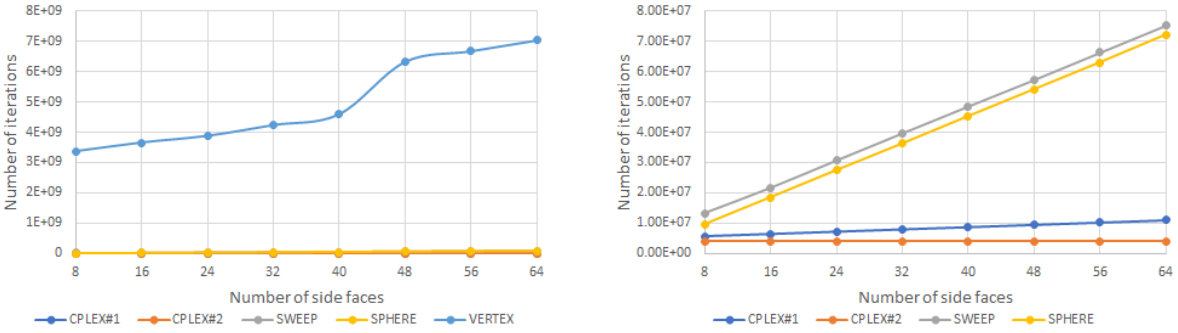
(a) Overall (b) Without VERTEX

Figure 16: Number of iterations of range computation in the 6D-prism query



(a) Overall (b) Without VERTEX

Figure 17: Number of iterations of range computation in the 8D-prism query



(a) Overall (b) Without VERTEX

Figure 18: Number of iterations of range computation in the 10D-prism query

Time cost of range computation

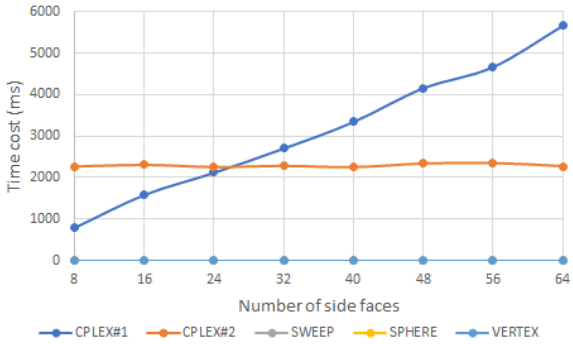
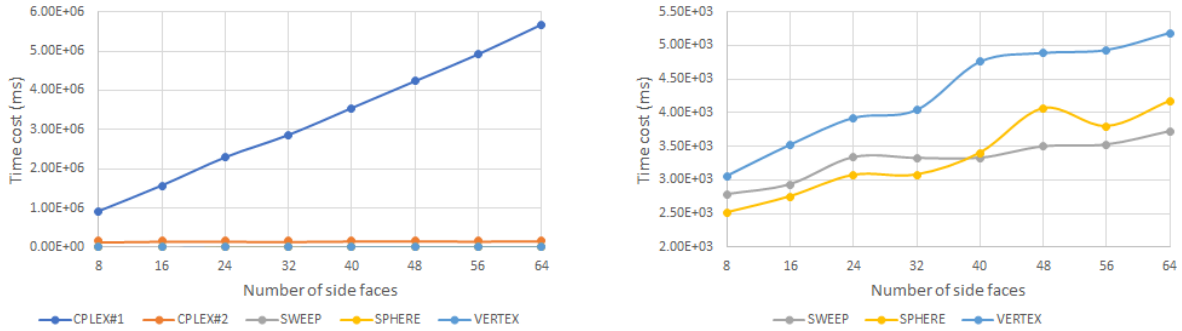
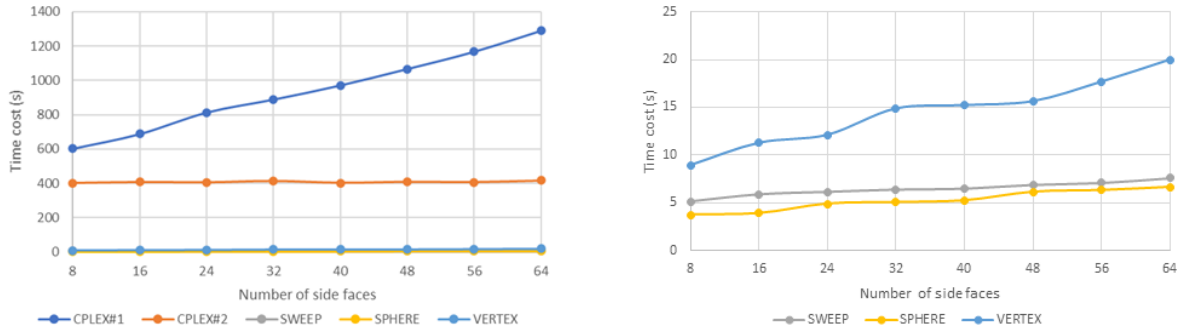


Figure 19: Time cost of range computation in the 2D-prism query



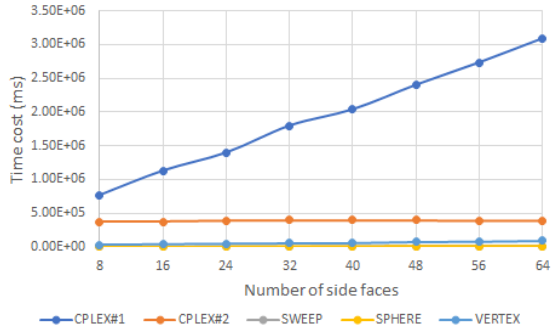
(a) Overall (b) Without CPLEX#1 and CPLEX#2

Figure 20: Time cost of range computation in the 4D-prism query

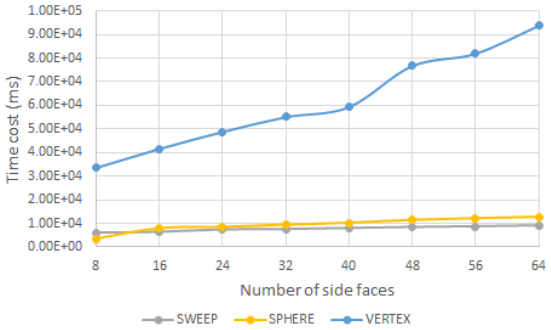


(a) Overall (b) Without CPLEX#1 and CPLEX#2

Figure 21: Time cost of range computation in the 6D-prism query

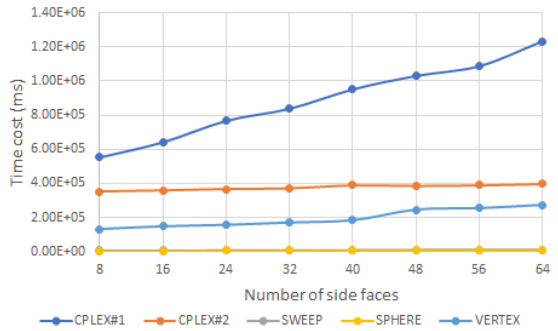


(a) Overall

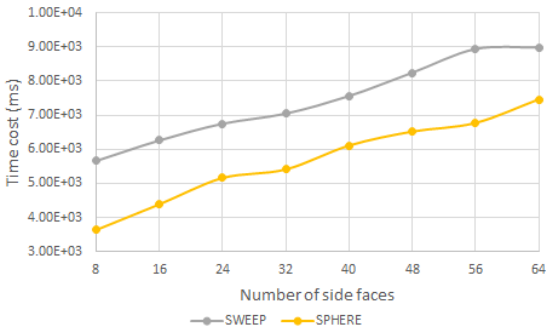


(b) Without CPLEX#1 and CPLEX#2

Figure 22: Time cost of range computation in the 8D-prism query



(a) Overall



(b) Without CPLEX#1, CPLEX#2 and VERTEX

Figure 23: Time cost of range computation in the 10D-prism query

These figures indicate that for all solutions, the number of half-spaces influences the time cost of range computation linearly, although the gradients vary. CPLEX#2 holds the best scalability. The approach spends one iteration to determine whether an intersection occur. So, the number of iteration it takes equals the number of nodes visited during the search. As prisms are very close to each other, nodes involved in all cases are the same, which results in nearly constant time cost. On the other hand, CPLEX#1 decomposes the prism to individual half-spaces to further detect inside or partial overlap. Consequently, its iteration and time cost increase. SWEEP and SPHERE hold the superiority over others in time cost. However, because of more FPNs, SPHERE takes more iterations than SWEEP. This gap becomes larger when the dimensionality goes high, as Figure 17 shows. In 10D, although iterations do not vary much, the FPNs from SPHERE is much larger than that of SWEEP (Table 8). This will greatly slow down the second filter.

Table 8: Selectivity of the first filters in the nD-prism test

	2D	4D	6D	8D	10D
CPLEX#1	1‰	1.857‰	13.39‰	71.29‰	247.9‰
CPLEX#2	1‰	1.932‰	13.39‰	71.29‰	247.9‰
SWEEP	1‰	1.857‰	13.39‰	71.29‰	247.9‰
SPHERE	1‰	1.938‰	15.33‰	71.29‰	749.4‰
VERTEX	1‰	1.857‰	13.39‰	71.29‰	247.9‰

5 Discussion

The strong aspect of nD-simplex model for query tests lies in the pseudo-randomness in terms of faces' directions. This results in diverse intersection angles between axis-parallel nodes and the simplex, which makes the result more generic and convincing. We achieved constant selectivity for both nD-simplex and nD-prism test, facilitating analysis of querying efficiency dependency on dimensionality and the number of half-spaces.

As shown in Tables 3 and 5, SWEEP becomes less competitive after 8D. FPNs occur at the acute corners where boundaries meet (Figure 10a), and this occurs increasingly frequently in higher-dimensional simplices. The nD-simplex is effectively a worst-case for the generation of FPNs, as SWEEP do not return any FPNs in the nD-prism test (Table 8).

The clipping method [3] on the other hand, clips the nodes intersecting each half-space. In this way, the intersection detection becomes a joint determination from all half-spaces, and FPNs are expected to be reduced. However, computation of the clipping position has to be improved in high dimensional spaces. The intersection will be an nD-plane which maximally results in 2^{n-1} possible vertices for clipping. With the original method, several iterations of clipping has to be performed to detect accurately whether a node intersects the polytope, which costs significant amount of time.

Rigorous method such as CPLEX takes more time in each iteration, but it holds a more constant performance over dimensionality thanks to accurate intersection computation. So, CPLEX remains to be a competitive solution when dimensionality is high.

In general, all approaches suffers from the 'the curse of dimensionality' that the FPR becomes very high. As different types of geometry (, e.g., triangle, cube and sphere) develop differently as dimensionality increases (, e.g., Figure 24), using hypercubes to approximate the polytope in high dimensional spaces may not be easy and can cause significant error. In fact, such approximating process is also a discretizing process, as the de-facto way to discretize in 2D is to use pixels. Consequently, we need to develop new methodologies for discretizing geometries in high dimensional spaces, rather than extending low dimensional schemes.

Additionally, our querying framework can solve more abstract queries whose constraints on combinations of different dimensions are expressible as a polytope model. For a convex curved face, a generic method is to generate a set of tangent planes which together form a superset of the geometry for approximation. More specifically, given a convex curved face: $f(x_0, x_1, \ldots, x_{n-1}) = 0$, we first randomly generate m points on f, from p_0 to p_{m-1}. Then, we compute the gradients at these points:

$$\nabla f(p_i) = \left(\frac{\partial f}{\partial x_0}(p_i), \frac{\partial f}{\partial x_1}(p_i), \ldots, \frac{\partial f}{\partial x_{n-1}}(p_i) \right)$$

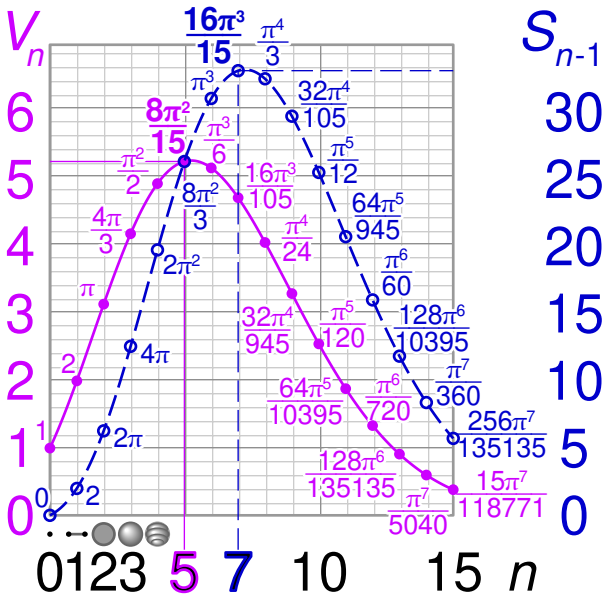


Figure 24: Volumes (V) and surface areas (S) of nD-balls of radius 1. Image source: Wikipedia

where p_i is one of the random points. These gradients serve as normal vectors of a set of hyperplanes which are what we need. With this, more complicated convex geometry queries can be resolved by directly adopting the querying framework established. Future work will further develop and verify this method.

References

[1] Pankaj K Agarwal et al. "Efficient searching with linear constraints". In: *Journal of Computer and System Sciences* 61.2 (2000), pp. 194–216.

[2] Bernard Chazelle. "Lower bounds on the complexity of polytope range searching". In: *Journal of the American Mathematical Society* 2.4 (1989), pp. 637–666.

[3] Jonathan Goldstein et al. "Processing queries by linear constraints". In: *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. 1997, pp. 257–267.

[4] Arijit Khan et al. "Towards indexing functions: Answering scalar product queries". In: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. 2014, pp. 241–252.

[5] Ricardo Lima. "IBM ILOG CPLEX - What is inside of the box?" In: *EWO Seminar*. Carnegie Mellon University. Pittsburgh, PA, USA, 2010.

[6] Haicheng Liu et al. "An efficient nD-point data structure for querying flood risk". In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLIII-B4-2021 (2021), pp. 367–374.

[7] Haicheng Liu et al. "HistSFC: Optimization for nD massive spatial points querying". In: *International Journal of Database Management Systems (IJDMS)* 12.3 (June 2020), pp. 7–28. DOI: 10.5121/ijdms.2020.12302.

[8] Jiři Matoušek. "Geometric range searching". In: *ACM Computing Surveys (CSUR)* 26.4 (1994), pp. 422–461.

[9] Oracle. *Indexes and Index-Organized Tables*. 2013. URL: https://docs.oracle.com/database/121/CNCPT/indexiot.htm.

[10] Rodney James Thompson. "Towards a Rigorous Logic for Spatial Data Representation". Published as NCG Publications on Geodesy no. 65. PhD thesis. Delft University of Technology, Dec. 2007, p. 333.

Reports published before in this series

1. GISt Report No. 1, Oosterom, P.J. van, Research issues in integrated querying of geometric and thematic cadastral information (1), Delft University of Technology, Rapport aan Concernstaf Kadaster, Delft 2000, 29 p.p.
2. GISt Report No. 2, Stoter, J.E., Considerations for a 3D Cadastre, Delft University of Technology, Rapport aan Concernstaf Kadaster, Delft 2000, 30.p.
3. GISt Report No. 3, Fendel, E.M. en A.B. Smits (eds.), Java GIS Seminar, Opening GDMC, Delft 15 November 2000, Delft University of Technology, GISt. No. 3, 25 p.p.
4. GISt Report No. 4, Oosterom, P.J.M. van, Research issues in integrated querying of geometric and thematic cadastral information (2), Delft University of Technology, Rapport aan Concernstaf Kadaster, Delft 2000, 29 p.p.
5. GISt Report No. 5, Oosterom, P.J.M. van, C.W. Quak, J.E. Stoter, T.P.M. Tijssen en M.E. de Vries, Objectgerichtheid TOP10vector: Achtergrond en commentaar op de gebruikersspecificaties en het conceptuele gegevensmodel, Rapport aan Topografische Dienst Nederland, E.M. Fendel (eds.), Delft University of Technology, Delft 2000, 18 p.p.
6. GISt Report No. 6, Quak, C.W., An implementation of a classification algorithm for houses, Rapport aan Concernstaf Kadaster, Delft 2001, 13.p.
7. GISt Report No. 7, Tijssen, T.P.M., C.W. Quak and P.J.M. van Oosterom, Spatial DBMS testing with data from the Cadastre and TNO NITG, Delft 2001, 119 p.
8. GISt Report No. 8, Vries, M.E. de en E. Verbree, Internet GIS met ArcIMS, Delft 2001, 38 p.
9. GISt Report No. 9, Vries, M.E. de, T.P.M. Tijssen, J.E. Stoter, C.W. Quak and P.J.M. van Oosterom, The GML prototype of the new TOP10vector object model, Report for the Topographic Service, Delft 2001, 132 p.
10. GISt Report No. 10, Stoter, J.E., Nauwkeurig bepalen van grondverzet op basis van CAD ontgravingsprofielen en GIS, een haalbaarheidsstudie, Rapport aan de Bouwdienst van Rijkswaterstaat, Delft 2001, 23 p.
11. GISt Report No. 11, Geo DBMS, De basis van GIS-toepassingen, KvAG/AGGN Themamiddag, 14 november 2001, J. Flim (eds.), Delft 2001, 37 p.
12. GISt Report No. 12, Vries, M.E. de, T.P.M. Tijssen, J.E. Stoter, C.W. Quak and P.J.M. van Oosterom, The second GML prototype of the new TOP10vector object model, Report for the Topographic Service, Delft 2002, Part 1, Main text, 63 p. and Part 2, Appendices B and C, 85 p.
13. GISt Report No. 13, Vries, M.E. de, T.P.M. Tijssen en P.J.M. van Oosterom, Comparing the storage of Shell data in Oracle spatial and in Oracle/ArcSDE compressed binary format, Delft 2002, .72 p. (Confidential)
14. GISt Report No. 14, Stoter, J.E., 3D Cadastre, Progress Report, Report to Concernstaf Kadaster, Delft 2002, 16 p.
15. GISt Report No. 15, Zlatanova, S., Research Project on the Usability of Oracle Spatial within the RWS Organisation, Detailed Project Plan (MD-NR. 3215), Report to Meetkundige Dienst – Rijkswaterstaat, Delft 2002, 13 p.
16. GISt Report No. 16, Verbree, E., Driedimensionale Topografische Terreinmodellering op basis van Tetraëder Netwerken: Top10-3D, Report aan Topografische Dienst Nederland, Delft 2002, 15 p.
17. GISt Report No. 17, Zlatanova, S. Augmented Reality Technology, Report to SURFnet bv, Delft 2002, 72 p.
18. GISt Report No. 18, Vries, M.E. de, Ontsluiting van Geo-informatie via netwerken, Plan van aanpak, Delft 2002, 17p.
19. GISt Report No. 19, Tijssen, T.P.M., Testing Informix DBMS with spatial data from the cadastre, Delft 2002, 62 p.

20. GISt Report No. 20, Oosterom, P.J.M. van, Vision for the next decade of GIS technology, A research agenda for the TU Delft the Netherlands, Delft 2003, 55 p.
21. GISt Report No. 21, Zlatanova, S., T.P.M. Tijssen, P.J.M. van Oosterom and C.W. Quak, Research on usability of Oracle Spatial within the RWS organisation, (AGI-GAG-2003-21), Report to Meetkundige Dienst – Rijkswaterstaat, Delft 2003, 74 p.
22. GISt Report No. 22, Verbree, E., Kartografische hoogtevoorstelling TOP10vector, Report aan Topografische Dienst Nederland, Delft 2003, 28 p.
23. GISt Report No. 23, Tijssen, T.P.M., M.E. de Vries and P.J.M. van Oosterom, Comparing the storage of Shell data in Oracle SDO_Geometry version 9i and version 10g Beta 2 (in the context of ArcGIS 8.3), Delft 2003, 20 p. (Confidential)
24. GISt Report No. 24, Stoter, J.E., 3D aspects of property transactions: Comparison of registration of 3D properties in the Netherlands and Denmark, Report on the short-term scientific mission in the CIST – G9 framework at the Department of Development and Planning, Center of 3D geo-information, Aalborg, Denmark, Delft 2003, 22 p.
25. GISt Report No. 25, Verbree, E., Comparison Gridding with ArcGIS 8.2 versus CPS/3, Report to Shell International Exploration and Production B.V., Delft 2004, 14 p. (confidential).
26. GISt Report No. 26, Penninga, F., Oracle 10g Topology, Testing Oracle 10g Topology with cadastral data, Delft 2004, 48 p.
27. GISt Report No. 27, Penninga, F., 3D Topography, Realization of a three dimensional topographic terrain representation in a feature-based integrated TIN/TEN model, Delft 2004, 27 p.
28. GISt Report No. 28, Penninga, F., Kartografische hoogtevoorstelling binnen TOP10NL, Inventarisatie mogelijkheden op basis van TOP10NL uitgebreid met een Digitaal Hoogtemodel, Delft 2004, 29 p.
29. GISt Report No. 29, Verbree, E. en S.Zlatanova, 3D-Modeling with respect to boundary representations within geo-DBMS, Delft 2004, 30 p.
30. GISt Report No. 30, Penninga, F., Introductie van de 3e dimensie in de TOP10NL; Voorstel voor een onderzoekstraject naar het stapsgewijs introduceren van 3D data in de TOP10NL, Delft 2005, 25 p.
31. GISt Report No. 31, P. van Asperen, M. Grothe, S. Zlatanova, M. de Vries, T. Tijssen, P. van Oosterom and A. Kabamba, Specificatie datamodel Beheerkaart Nat, RWS-AGI report/GIST Report, Delft, 2005, 130 p.
32. GISt Report No. 32, E.M. Fendel, Looking back at Gi4DM, Delft 2005, 22 p.
33. GISt Report No. 33, P. van Oosterom, T. Tijssen and F. Penninga, Topology Storage and the Use in the context of consistent data management, Delft 2005, 35 p.
34. GISt Report No. 34, E. Verbree en F. Penninga, RGI 3D Topo - DP 1-1, Inventarisatie huidige toegankelijkheid, gebruik en mogelijke toepassingen 3D topografische informatie en systemen, 3D Topo Report No. RGI-011-01/GISt Report No. 34, Delft 2005, 29 p.
35. GISt Report No. 35, E. Verbree, F. Penninga en S. Zlatanova, Datamodellering en datastructurering voor 3D topografie, 3D Topo Report No. RGI-011-02/GISt Report No. 35, Delft 2005, 44 p.
36. GISt Report No. 36, W. Looijen, M. Uitentuis en P. Bange, RGI-026: LBS-24-7, Tussenrapportage DP-1: Gebruikerswensen LBS onder redactie van E. Verbree en E. Fendel, RGI LBS-026-01/GISt Rapport No. 36, Delft 2005, 21 p.
37. GISt Report No. 37, C. van Strien, W. Looijen, P. Bange, A. Wilcsinszky, J. Steenbruggen en E. Verbree, RGI-026: LBS-24-7, Tussenrapportage DP-2: Inventarisatie geo-informatie en -services onder redactie van E. Verbree en E. Fendel, RGI LBS-026-02/GISt Rapport No. 37, Delft 2005, 21 p.
38. GISt Report No. 38, E. Verbree, S. Zlatanova en E. Wisse, RGI-026: LBS-24-7, Tussenrapportage DP-3: Specifieke wensen en eisen op het gebied van plaatsbepaling, privacy en beeldvorming, onder redactie van E. Verbree en E. Fendel, RGI LBS-026-03/GISt Rapport No. 38, Delft 2005, 15 p.

39. GISt Report No. 39, E. Verbree, E. Fendel, M. Uitentuis, P. Bange, W. Looijen, C. van Strien, E. Wisse en A. Wilcsinszky en E. Verbree, RGI-026: LBS-24-7, Eindrapportage DP-4: Workshop 28-07-2005 Geo-informatie voor politie, brandweer en hulpverlening ter plaatse, RGI LBS-026- 04/GISt Rapport No. 39, Delft 2005, 18 p.
40. GISt Report No. 40, P.J.M. van Oosterom, F. Penninga and M.E. de Vries, Trendrapport GIS, GISt Report No. 40 / RWS Report AGI-2005-GAB-01, Delft, 2005, 48 p.
41. GISt Report No. 41, R. Thompson, Proof of Assertions in the Investigation of the Regular Polytope, GISt Report No. 41 / NRM-ISS090, Delft, 2005, 44 p.
42. GISt Report No. 42, F. Penninga and P. van Oosterom, Kabel- en leidingnetwerken in de kadastrale registratie (in Dutch) GISt Report No. 42, Delft, 2006, 38 p.
43. GISt Report No. 43, F. Penninga and P.J.M. van Oosterom, Editing Features in a TEN-based DBMS approach for 3D Topographic Data Modelling, Technical Report, Delft, 2006, 21 p.
44. GISt Report No. 44, M.E. de Vries, Open source clients voor UMN MapServer: PHP/Mapscript, JavaScript, Flash of Google (in Dutch), Delft, 2007, 13 p.
45. GISt Report No. 45, W. Tegtmeier, Harmonization of geo-information related to the lifecycle of civil engineering objects – with focus on uncertainty and quality of surveyed data and derived real world representations, Delft, 2007, 40 p.
46. GISt Report No. 46, W. Xu, Geo-information and formal semantics for disaster management, Delft, 2007, 31 p.
47. GISt Report No. 47, E. Verbree and E.M. Fendel, GIS technology – Trend Report, Delft, 2007, 30 p.
48. GISt Report No. 48, B.M. Meijers, Variable-Scale Geo-Information, Delft, 2008, 30 p.
49. GISt Report No. 48, Maja Bitenc, Kajsa Dahlberg, Fatih Doner, Bas van Goort, Kai Lin,Yi Yin, Xiaoyu Yuan and Sisi Zlatanova, Utilty Registration, Delft, 2008, 35 p.
50. GISt Report No 50, T.P.M. Tijssen en S. Zlatanova, Oracle Spatial 11g en ArcGIS 9.2 voor het beheer van puntenwolken (Confidential), Delft, 2008, 16 p.
51. GISt Report No. 51, S. Zlatanova, Geo-information for Crisis Management, Delft, 2008, 24 p.
52. GISt Report No. 52, P.J.M. van Oosterom, INSPIRE activiteiten in het jaar 2008 (partly in Dutch), Delft, 2009, 142 p.
53. GISt Report No. 53, P.J.M. van Oosterom with input of and feedback by Rod Thompson and Steve Huch (Department of Environment and Resource Management, Queensland Government), Delft, 2010, 60 p.
54. GISt Report No. 54, A. Dilo and S. Zlatanova, Data modeling for emergency response, Delft, 2010, 74 p.
55. GISt Report No. 55, Liu Liu, 3D indoor " door-to-door" navigation approach to support first responders in emergency response – PhD Research Proposal, Delft, 2011, 47 p.
56. GISt Report No. 56, Md. Nazmul Alam, Shadow effect on 3D City Modelling for Photovoltaic Cells – PhD Proposal, Delft, 2011, 39 p.
57. GIST Report No. 57, G.A.K. Arroyo Ohori, Realsing the Foundations of a Higher Dimensional GIS: A Study of Higher Dimensional Data Models, Data Structures and Operations – PhD Research Proposal, Delft, 2011, 68 p.
58. GISt Report No. 58, Zhiyong Wang, Integrating Spatio-Temporal Data into Agent-Based Simulation for Emergency Navigation Support – PhD Research Proposal, Delft, 2012, 49 p.
59. GISt Report No. 59, Theo Tijssen, Wilko Quak and Peter van Oosterom, Geo-DBMS als standard bouwsteen voor Rijkswaterstaat (in Dutch), Delft, 2012, 167 p.
60. GISt Report No. 60, Amin Mobasheri, Designing formal semantics of geo-information for disaster response – PhD Research Proposal, Delft, 2012, 61 p.
61. GISt Report No. 61, Simeon Nedkov, Crowdsourced WebGIS for routing applications in disaster management situations, Delft, 2012, 31 p.

62. GISt Report No. 62, Filip Biljecki, The concept of level of detail in 3D city models – PhD Research Proposal, Delft, 2013, 58 p.
63. GISt Report No. 63, Theo Tijssen & Wilko Quak, GISt activiteiten voor het GeoValley project – Projectnummer: GBP / 21F.005, Delft, 2013, 39 p.
64. GISt Report No. 64, Radan Šuba, Content of Variable-scale Maps – PhD Proposal, Delft, 2013, 36 p.
65. GISt Report No. 65, Ravi Peters, Feature aware Digital Surface Model analysis and generalization based on the 3D Medial Axis Transform, - PhD Research Proposal, Delft 2014, 55 p.
66. GISt Report No. 66, Sisi Zlatanova, Liu Liu, George Sithole, Junqiao Zhao and Filippo Mortani, Space subdivision for indoor applications, Delft, 2014, 38 p.
67. GISt Report No. 67, Sisi Zlatanova, Jakob Beetz, Joris Goos, Albert Mulder, Anne-Jan Boersma, Hans Schevers, Marian de Vries and Tarun Ghawana, Spatial Infrastructure for the Port of Rotterdam, Delft, 2014, 35 p.
68. GISt Report No. 68, Edward Verbree en Peter van Oosterom, Exploratieve Puntenwolken, Delft, 2015, 60 p.
69. GISt Report No. 69, Wilko Quak, Storage of spatial properties of NWB-vaarwegen in Neo4j, Delft, 2016, 13 p.
70. GISt Report No. 70, Edward Verbree and Peter van Oosterom, Standaardisatie van Puntenwolken, Delft, 2016, 66 p.
71. GISt Report No. 71, Martijn Meijers, Peter van Oosterom and Wilko Quak, Management of AIS messages in a Geo-DBMS, Delft, 2018, 33 p.
72. GISt Report No. 72, Martijn Meijers and Peter van Oosterom, Clustering and Indexing historic AIS data with Space Filling Curves, Delft, 2018, 33 p.
73. GISt Report No. 73, Abdullah Allatas, Supporting indoor navigation using access rights to spaces based on an integrated IndoorGML and LADM model, - PhD Research Proposal, Delft, 2019, 54 p.
74. GISt Report No. 74, Mingxue Zheng, Classification of Point Clouds of Urban Scenes exploiting a hierarchy of Point Densities, - PhD Research Proposal, Delft, 2019, 59 p.
75. GISt Report No. 75, Haicheng Liu, Towards 10^{15} Points Management – an nD Point Cloud Approach, PhD Research Proposal, Delft, 2018, 67 p.
76. GISt Report No. 76, Agung Indrajit, 4D Open Spatial Information Infrastructure supporting Participatory Urban Planning Monitoring, PhD Research Proposal, Delft, 2019, 112 p.
77. GISt Report No. 77, Kalogianni Eftychia, 3D Land Administration System within the Spatial Development Lifecycle, PhD Proposal, Delft, 2010, 56 p.

Chair GIS technology
Faculty of Architecture and the Built Environment, TU Delft
Julianalaan 134, 2628 BL Delft
Postbus 5043, 2600 GA Delft

Phone: +31 (0)15 2786950
E-mail: P.J.M.vanOosterom@tudelft.nl
www.gdmc.nl