nD-PointClouds
a model for deeply integrating space, time and scale

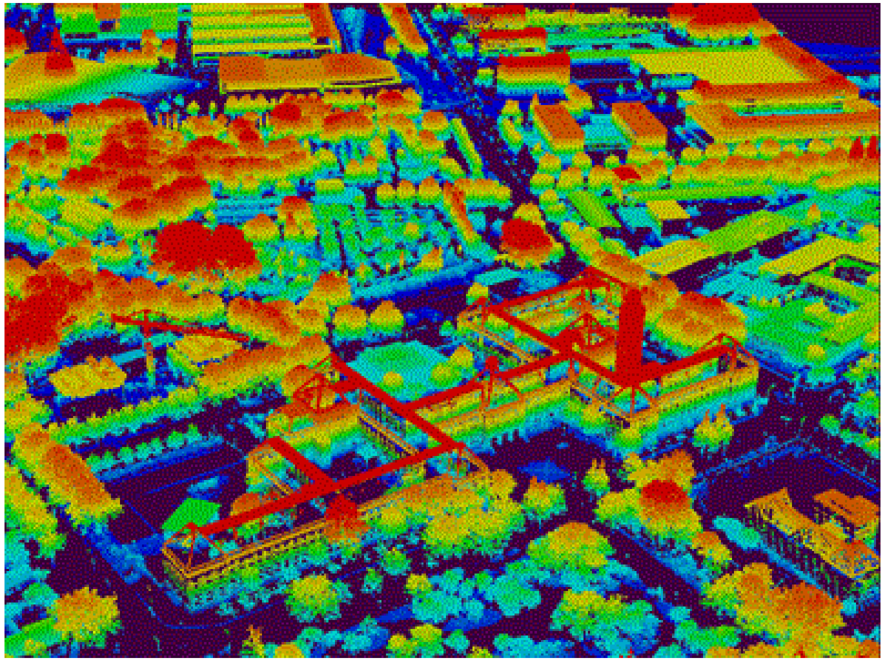
20-10-2016

Peter van Oosterom

Plenary lecture at the Joint 3D Athens Conference,
18-21 October, Athens, Greece

Overview

- motivation
- scale as dimension
- functionality
- data management
- standardization (if time allows)
- conclusion



acknowledgements: based on joint work with Edward Verbree, Theo Tijssen, Oscar Martinez-Rubi, Mike Horhammer, Stella Psomadaki, Xuefeng Guan, …

kind of sequel on last years keynote at JIGC 2015, Kuala Lumpur: Realistic benchmarks for point cloud data management systems

Motivation

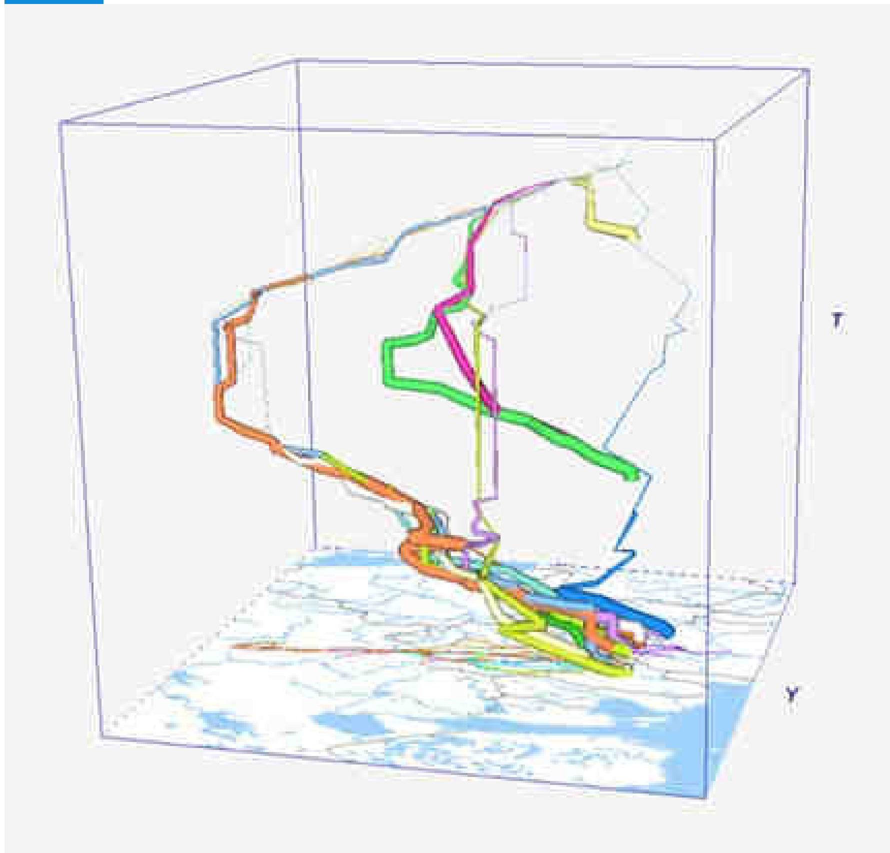
- point cloud data sets are often used for monitoring
  → dynamic point clouds
  → time as additional organizing dimension

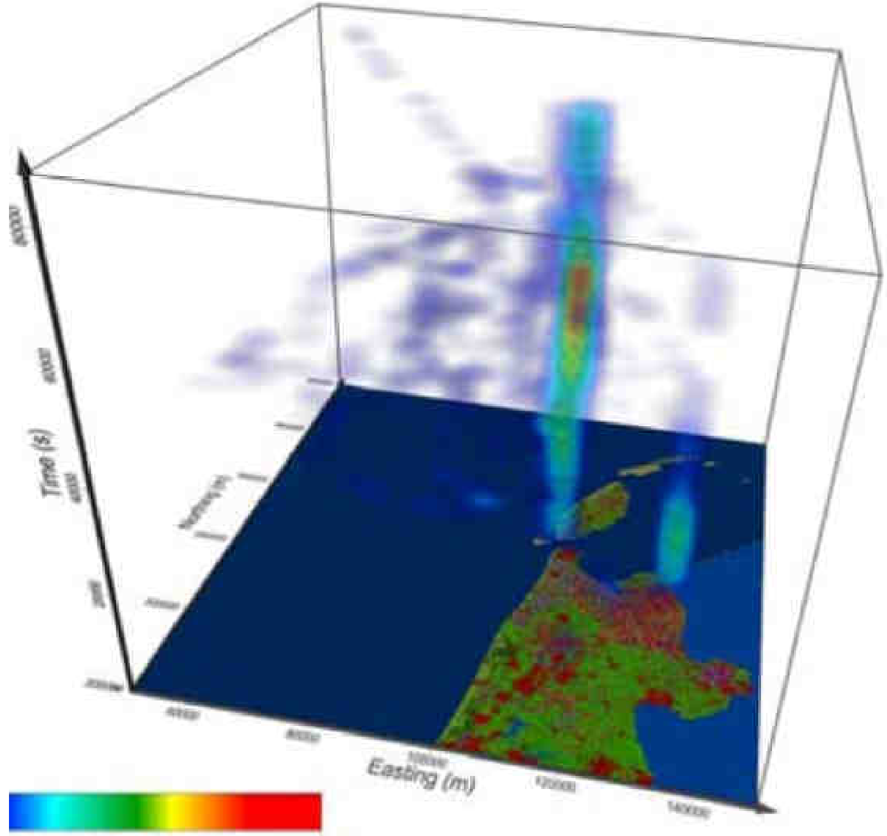
- organizing point cloud data in LoD's/importance levels is an approach to manage large data sets
  → LoD: discrete (multi-scale) or continuous (vario-scale)
  → scale treated as additional organizing dimension

- how to manage higher dimensional point clouds (4D, 5D)

Time as dimension

time more obvious: well-known space-time cubes

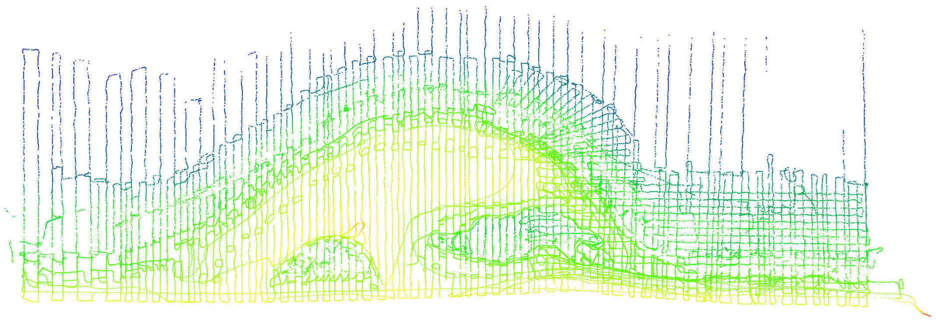




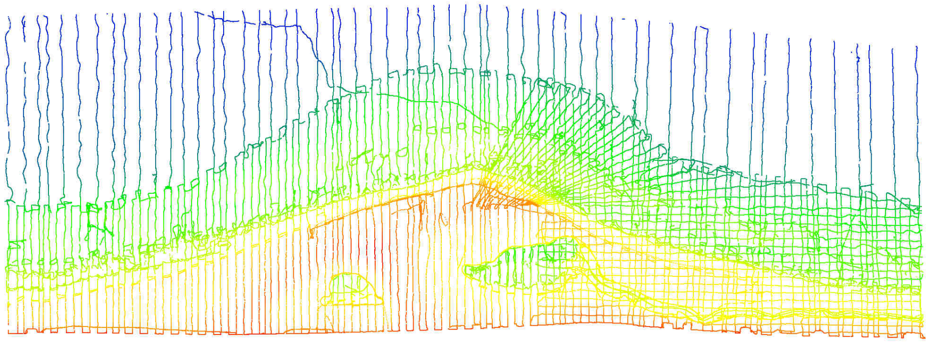
Dynamic Point Clouds

- point clouds are generated every day, hour, minute
- repeated scans of the same area → dynamic
- time as selective as the spatial component or needed in integrated space – time selections
- current DBMS solutions designed for static point clouds
- management is still a challenge
- example Sand Engine, time series Dutch coast, Deltares (see Psomadaki et al, Friday)

2011

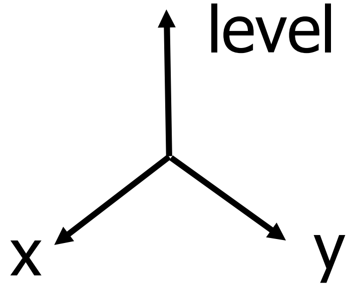


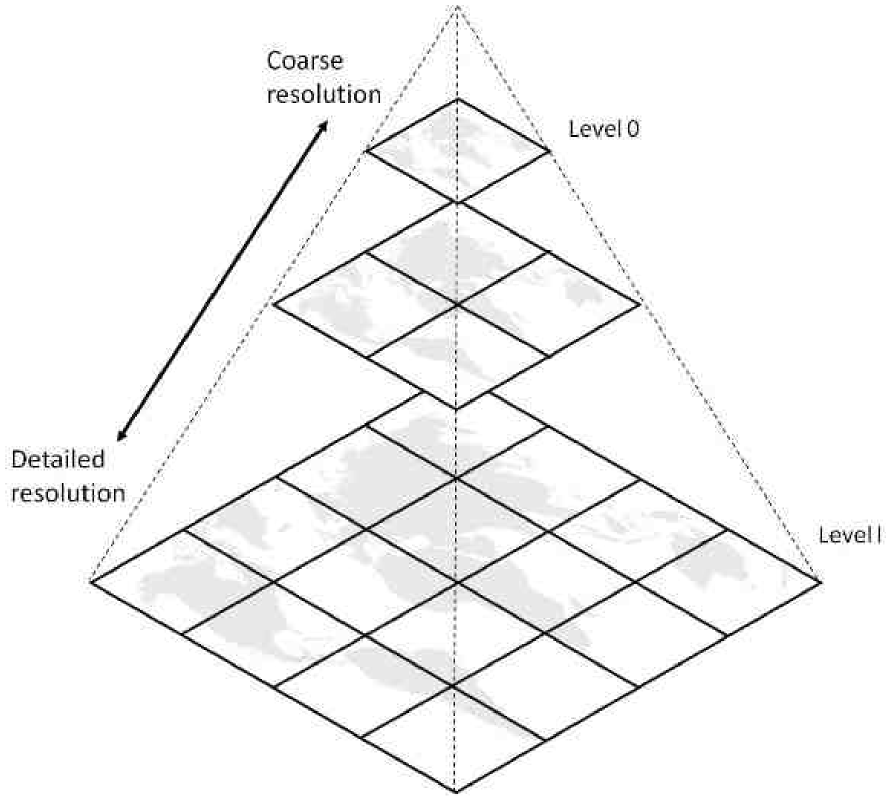
2015



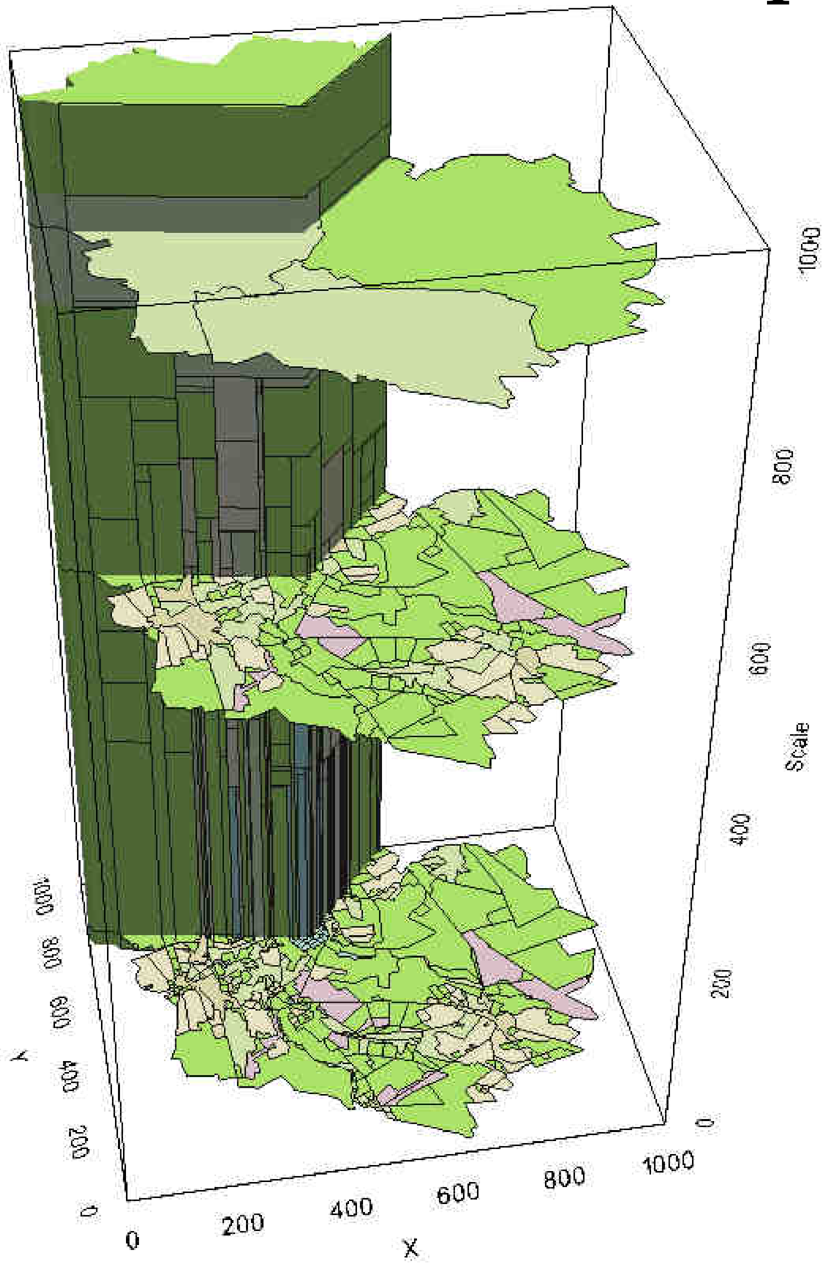
Scale as dimension

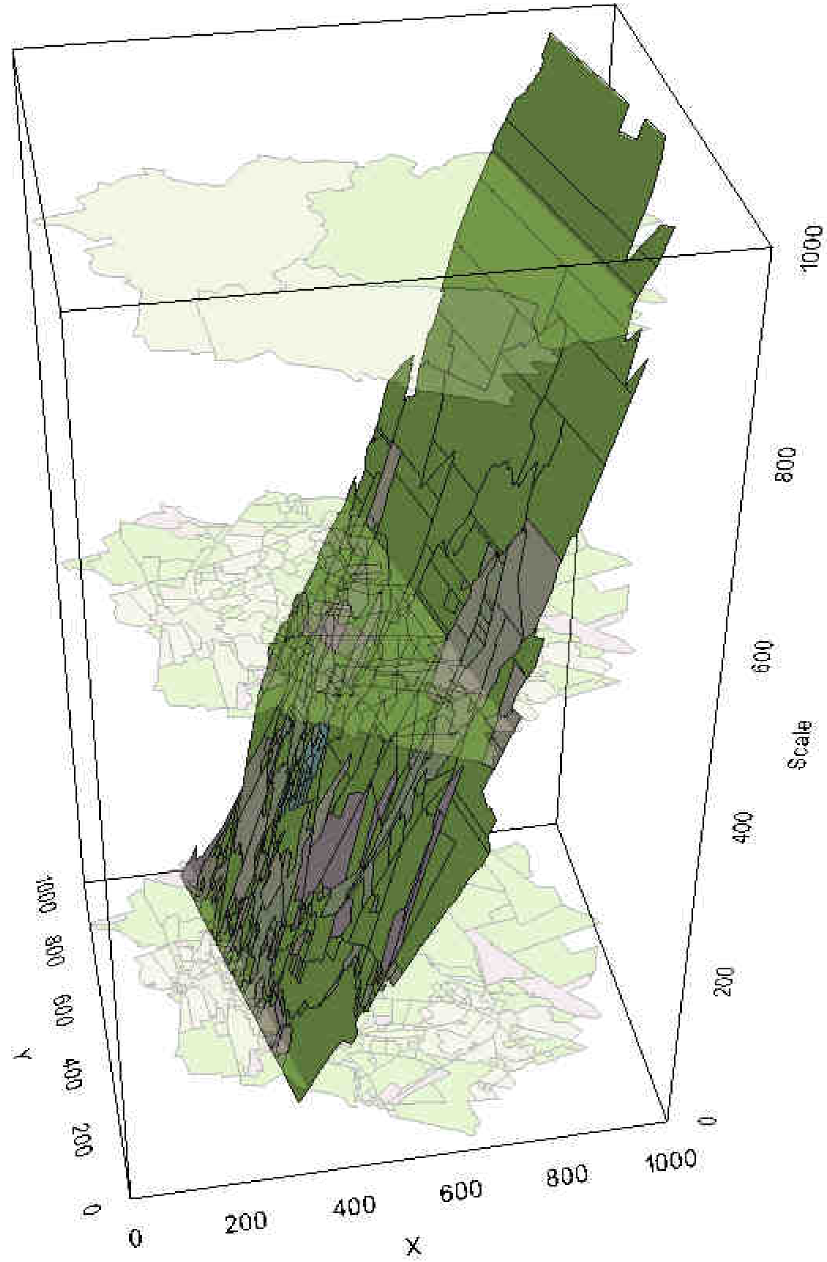
- less obvious than time
- data pyramids
  (Level of Detail/ Multi-scale)
- well-known from raster data
- results in discrete
  number of levels (multi-scale)
- level could be considered
  as additional dimension





Vario-scale with polygonal vector data





Representations of space, time, scale ...after grid/voxel or object/vector

- new 3rd representation: nD point cloud (PC)
- many scientific domains (spatial): geography, medicine, physics, astronomy, hydrology, architecture, archaeology, arts, CAD, social media/ moving objects, gaming…
- deep integration space/time/scale
  1. more efficient, store, exchange, compute
  2. more functionality (smooth zoom/ analysis)
- nD PC in whole processing chain: acquisition, DBMS, analysis, simulation, dissemination, visualization,…
- BIG spatial data: 35 trillion points (in astronomy, geo-info)

Overview

- motivation
- scale as dimension
- functionality
- data management
- standardization
- conclusion

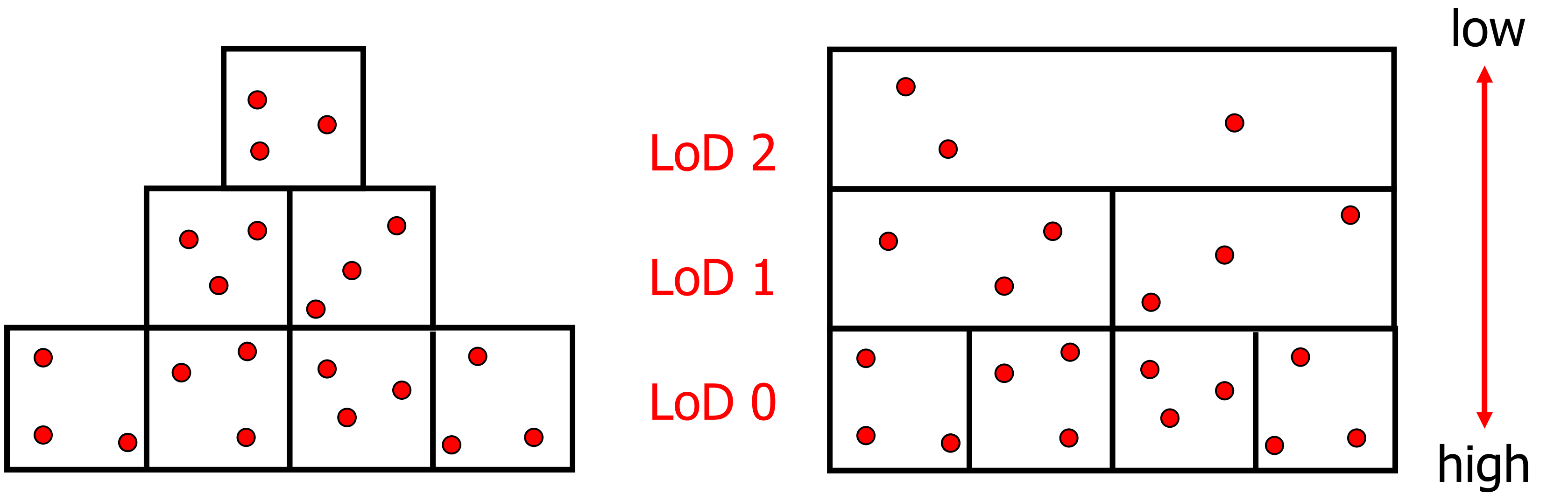
LoD/multi-scale Point Cloud

- data pyramid (Level of Detail/ Multi-scale) in analogy with raster
- imagine a fine 2D (or 3D) bottom level grid to organize the points
- option is after every 4 points in cell move 5th point to parent cell (for 2D organization and every 9th point in case of 3D), recursively bottom-up filling the cell/blocks at higher levels
- results in data pyramid → discrete number of levels (multi-scale)
- Note: depending on input data distribution,
  some areas my reach higher levels than others

Point cloud data pyramid

- overview queries just want top-subset
- detailed queries part of bottom-subset
- organize in data pyramid

2D schematic view, data blocks.... stretched over domain density



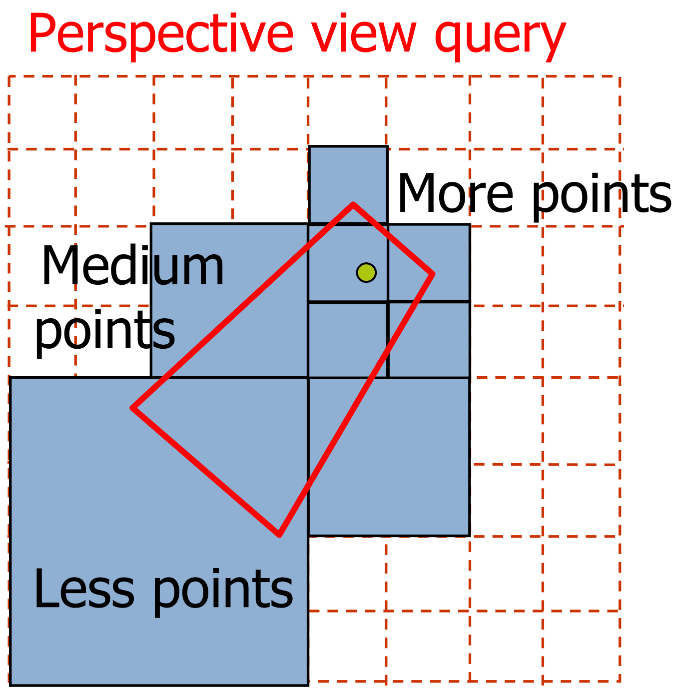
every next higher level, density 2^k times less (2D → 4, 3D → 8)

Data pyramid/multi-scale

- allows fast spatial searching including LoD selection

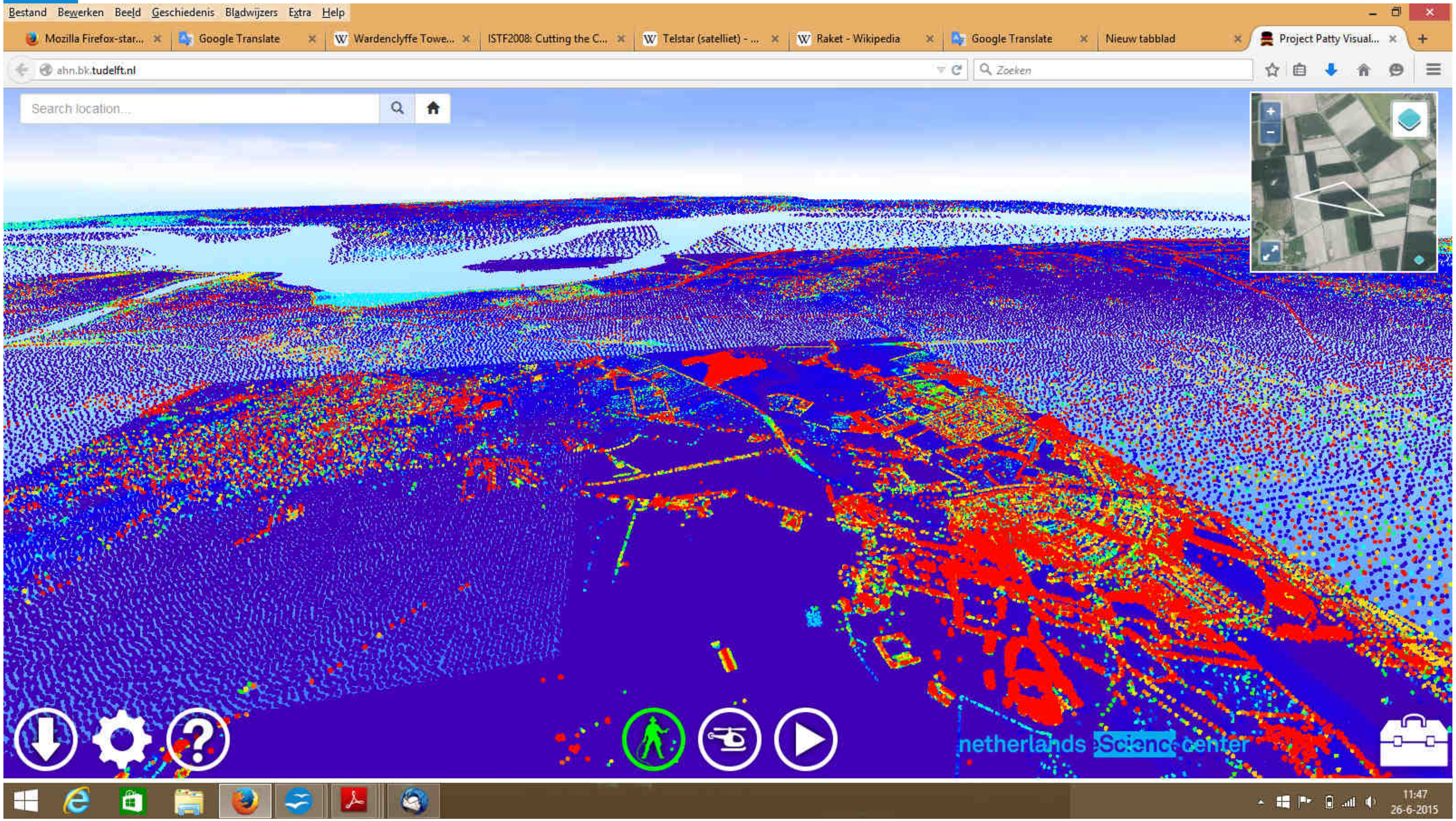
- the further away from viewer the lesser points selected (i.e. the higher level blocks/points)

- drawbacks:
  1. discrete number of levels
  2. bottom-up filling, unbalanced top
  3. point random assigned to level



Discrete LoD's are visible…

http://ahn2.pointclouds.nl: 640.000.000.000 points on-line 3D viewer



Data pyramid alternatives

- not random points, but more characteristic points move up (more important), some analysis needed; e.g.:
  1. compute local data density → more dense less important
  2. compute local surface shape → more flat less important
  3. other criteria, data collection/application dependent (intensity)

  (combine into) one imp_value of point → better than random

- not bottom-up, but top-down population, make sure that top levels are always filled across complete domain (lower levels may not be completely filled)

Further improvements …
beyond discrete levels

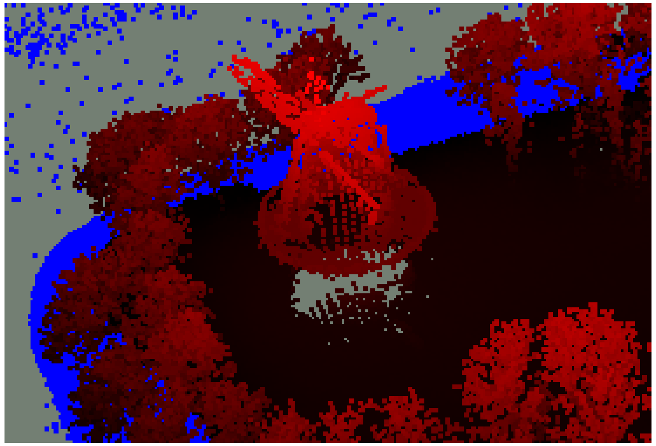
- might result in artefacts when looking at perspective view image (possible ‘see’ blocks of different levels)
- also not optimal within block (near viewer perhaps not enough points, further from viewer perhaps too much points)
- would a true vario-scale option possible?

→ Vario-scale geo-info research at TU Delft

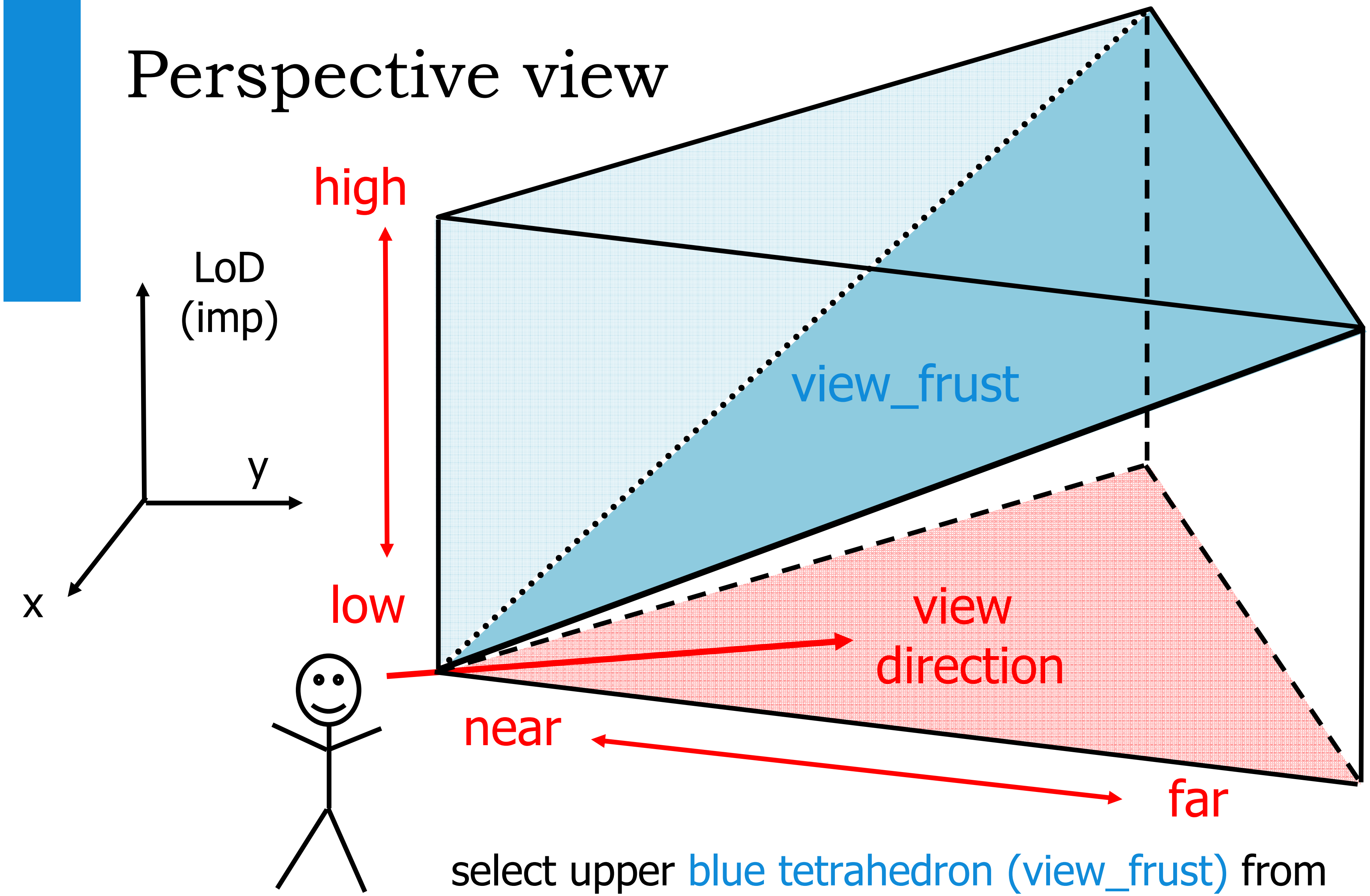
Vario-scale for point cloud data

- lesson from vario-scale research: add one continuous dimension to the geometry to represent scale
  (2D data vario-scale represented by 3D geometry)

- apply this to point cloud data
  1. compute the imp value
  2. add this as dimension, either
     x,y,imp (z and others attributes) or
     x,y,z,imp (and others as attributes)
  3. Cluster/index the 3D or 4D point
  4. Define perspective view selections,
     view frustum with one more dimension:
     the further, the higher imp's



Perspective view



select upper blue tetrahedron (view_frust) from prism-part of vario-scale x,y,imp point cloud data cube

Normal view frustum selection and streaming based on importance

- view frustum selection

```
select point
from point_cloud
where overlaps (point, view_frust)
```

- ordered on importance for streaming

```
select point
from point_cloud
where overlaps (point, view_frust)
order by imp desc;
```

(or distance from tilted plane)

Delta queries for moving and zoom in/out

- select and send new points:
  `point in new_frust and point not in old_frust`

- find and drop old points:
  `point in old_frust and not in new_frust`

- note this works form both
  1. changing view position x,y(,z)
  2. zooming in or out (‘view from above’, imp-dimension)

- optional to work at point or block granularity
  (in selection and server-client communication)

Overview

- motivation
- scale as dimension
- functionality
- data management
- standardization
- conclusion

Point cloud analysis

- benefits:
  - no conversion time
  - no data loss
  - analysis may be better
  - LoD continuous (raster pixels factor 2, vector hard)
  - very realistic representations (e.g tree with leaves)

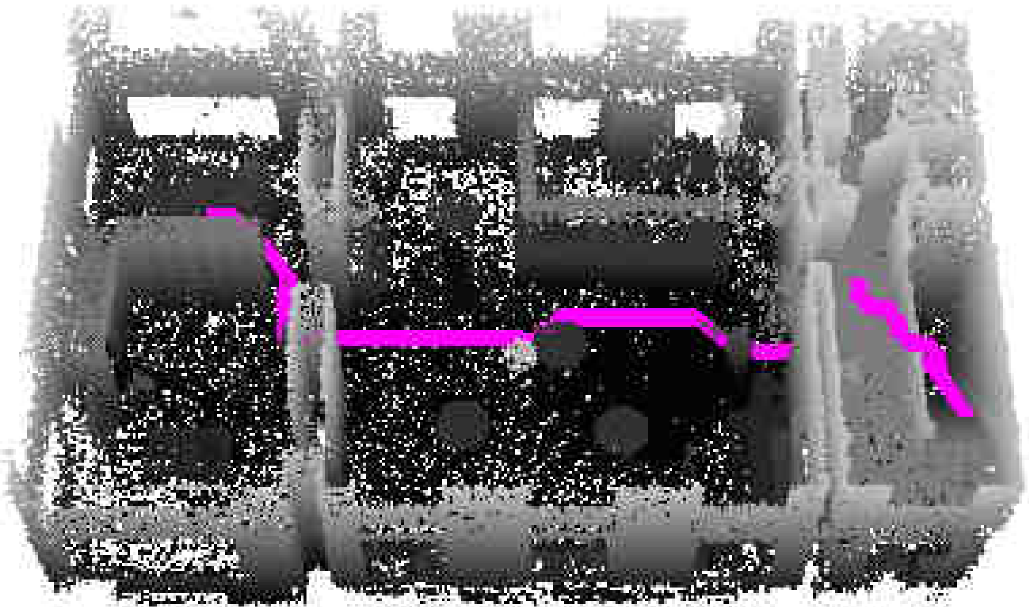
- drawbacks:
  - lot of data
  - redevelop algorithms

- have it as option
  together with conversions PC ←→vector, PC←→ raster

Types of analysis, direct point clouds

- solar energy potential
- viewshed/ line-of-sight
- 3D routing (e.g. drone; see Rodenberg et al, Friday)
- change detection (deformations)
- volume analysis computations
- hydrology/ flow over surface
- vegetation analysis

- continuous LoD
  also for analysis
  not only visualization



Point cloud base functionality (1/2)

1. simple range/rectangle filters (of various sizes)
2. selections based on points along a linear route (with buffer)
3. selections of points overlapping a 2D polygon
4. selections based on the attributes such as intensity I (/RGB)
5. multi-resolution/LoD selection (select top x%)
6. sort points on relevance/importance (support streaming)
7. slope orientation or steepness computation
8. compute normal vector of selected points
9. convert point cloud to TIN representation
10. convert point cloud to Grid (DEM)
11. convert point cloud to contours
12. k-nearest neighbor selection (approx or exact)
13. selection based on point cloud density
14. spatial join with other table; e.g. 100 building polygons
15. spatiotemporal selection queries (specify space+time range)
16. temporal differences computations and selection
17. compute min/max/avg/median height in 2D/3D area

Point cloud base functionality (2/2)

18. hill shading relief (image based on point cloud/DEM/TIN)
19. view shed analysis (directly on point cloud with fat points)
20. flat plane detection (and segmentation point, add plane_id)
21. curved surface detection (cylinder, sphere patches, freeform)
22. compute area of implied surface (by point cloud)
23. compute volume below surface
24. select on address/postal code/geographic names (gazetteer)
25. coordinate transformation RD-NAP - ETRS89
26. compute building height using point cloud (diff in/outside)
27. compute cross profiles (intersect with vertical plane)
28. combine multiple point clouds (Laser+MBES)
29. volume difference between design (3D polyhedral) surface and point could
30. detect break line point cloud surface
31. selection based on perspective view (point cloud density)
32. delta selection of query 31, moving to new position

Overview

- motivation
- scale as dimension
- functionality
- data management
- standardization
- conclusion

nD PC data management

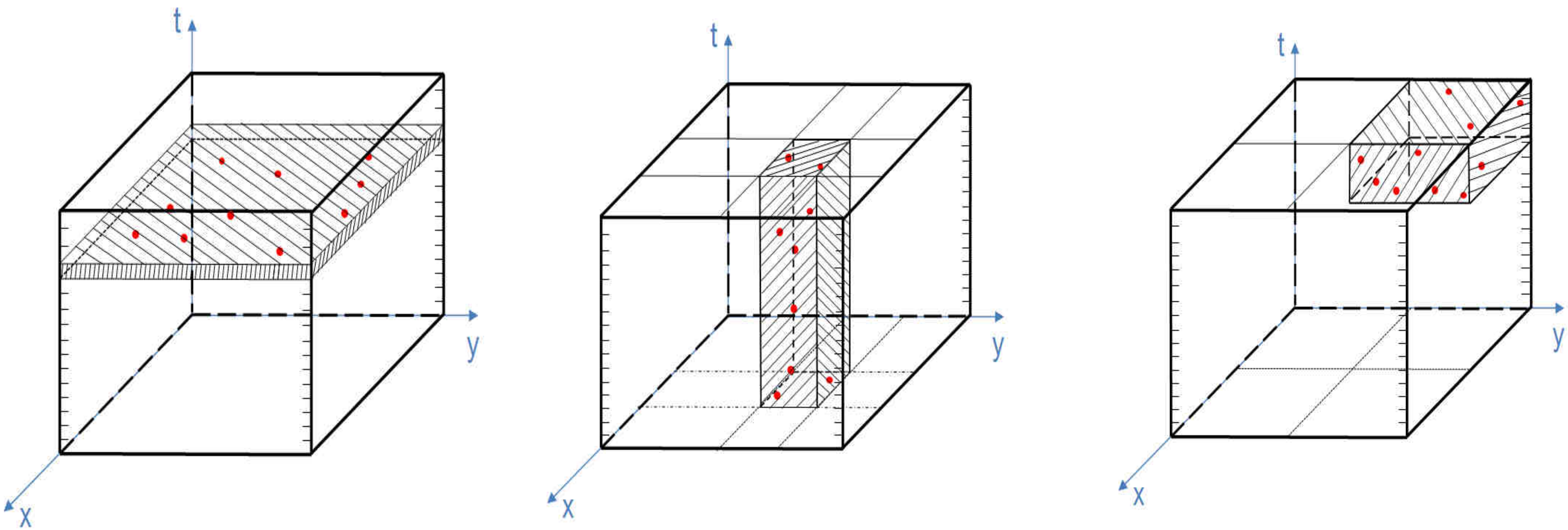
- management of nD PC data, starts by defining
  - dimensions (and their roles/priorities in the points)
  - associated attributes

- dimensions are main drivers for data organization, clustering, indexing, subdivision (for parallel processing), compression, blocking/ caching and streaming of data

- investigate various data management options
  - kd-tree based organization (no scaling issues of different dimensions)
  - organization based on simplices (e.g. triangle/tet bins, Sierpinski)
  - *integrate dimension values in 1 value via Space Filling Curve (SFC): Morton, Hilbert, and relation to quadtree*

Different blocking scheme's for space-time (or space-scale) cube

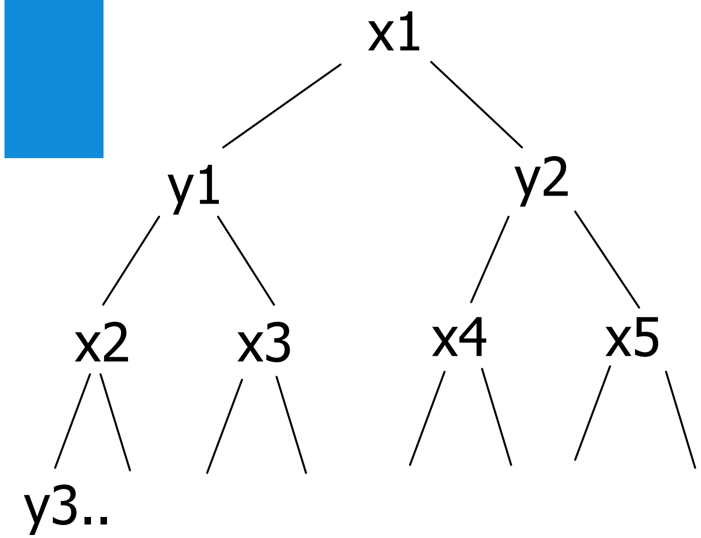
16x16x1 4x4x16 8x8x4

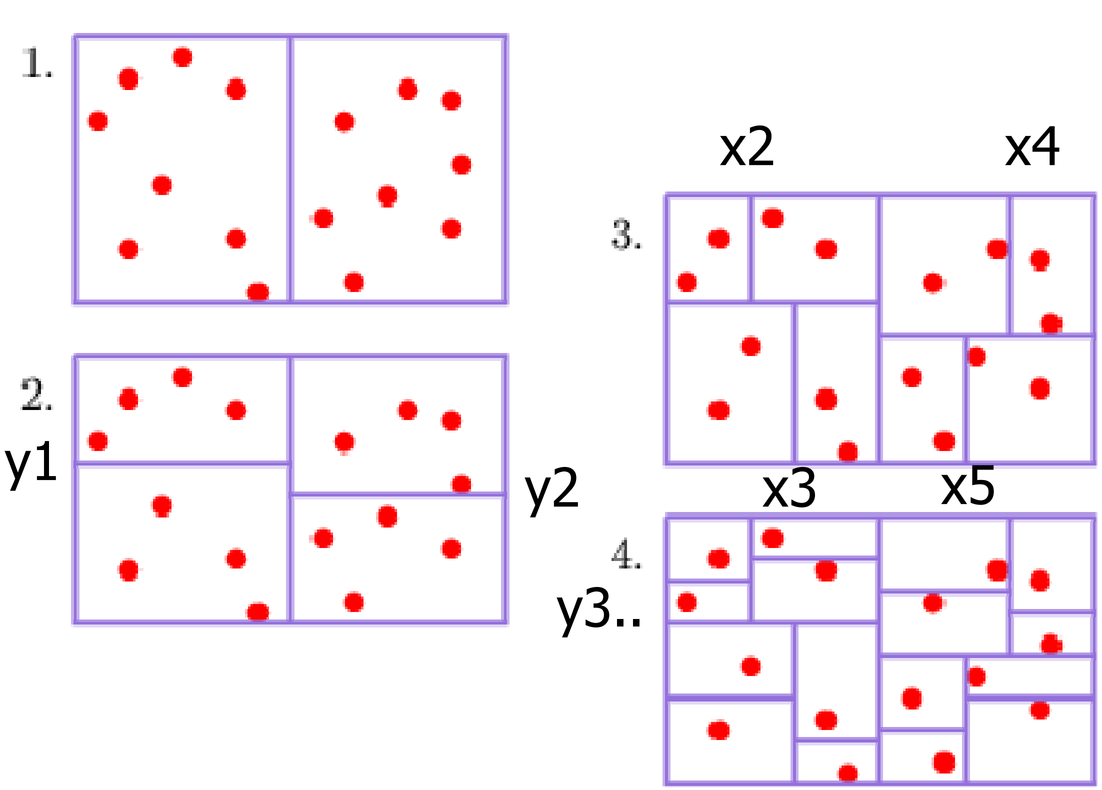


- challenge increases for higher dimensional hyper-cubes:
  - 4D: 2D space-time-scale, 3D space-time, 3D space-scale
  - 5D: 3D space-time-scale

http://graphics.stanford.edu

kd-tree

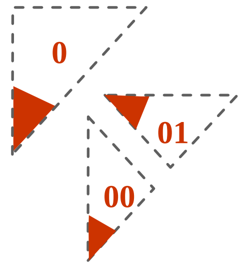


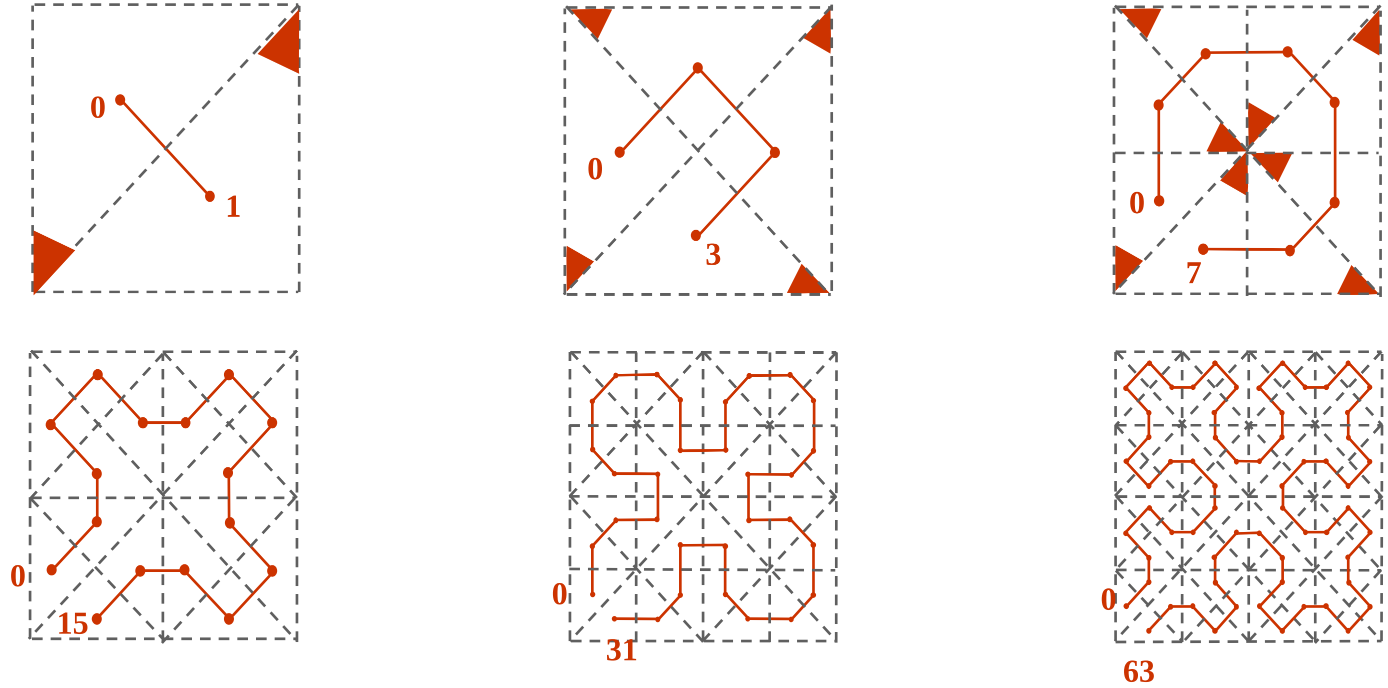


- alternating x, y split
- needs resorting (again and again)
- works in nD (alternative x,y,z split, or x,y,z,t split, or ..)
- may get unbalanced, not dynamic
- dimensions metric independent (scaled, distributed differently)
- used by László Dobos et al (cosmological particles, Bridget Falck)

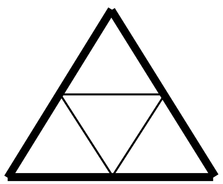
Simplices based

- Sierpinski Curves: start with two triangles (2D) and split recursively
- works in 3D (tets) and higher?





- Elliot Sefton-Hash uses triangle bins with quadsplit (planetary data)



nD-PointClouds data management

- modelling theory for nD point cloud data
- tools to support modelers, developers and users in point cloud data organization design decisions for (given 1. data sets and 2. required functionalities in applications):
  - what are the dimensions,
  - what are the attributes,
  - what type of organization: Morton-code/ kd-tree/ nD simplices-part,
  - what relative scale of various dimensions,
  - parameters such as clustering/ blocking size,
  - what compression,
  - what approach and level of parallelism (incl. hardware aspects),

→ Modeling workbench

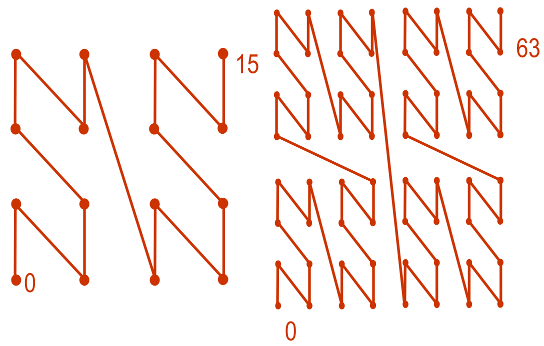
In detail: Space Filling Curves (SFCs)

- apply linear ordering to a multidimensional domain (spatial clustering)
- organize a flat table efficiently
- full resolution keys: avoid storing x,y[,z] + t/l
  → recovered from SFC key

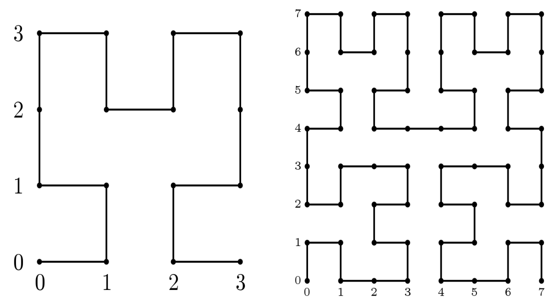
- use Index Organized Table
  (data stored in the B-Tree index)
- queries need to be re-written to SFC-ranges, benefit from spatial clustering → efficient

- SFCs based on hyper-cubes
  - Morton/Hilbert both **nD and quadrant recursive**
  - Consider relative scaling of dimensions
  - Space reserved on the hypercube for future data

Morton (Peano)



Hilbert

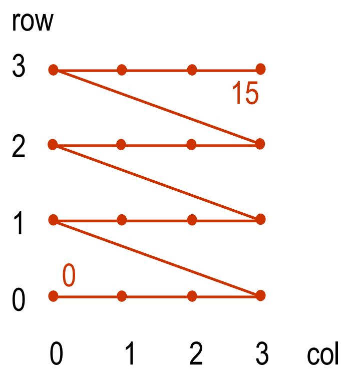


Some Space Filling Curves

space filling curve used for block/cell creation

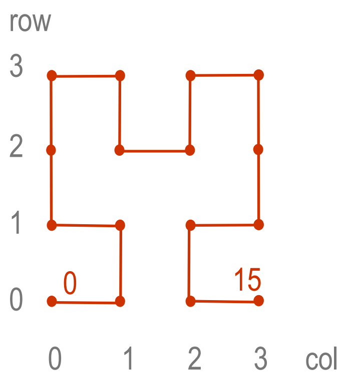
- ordering or numbering of cells in kD into 1D using bi-jective mapping

Row (first y, then x)

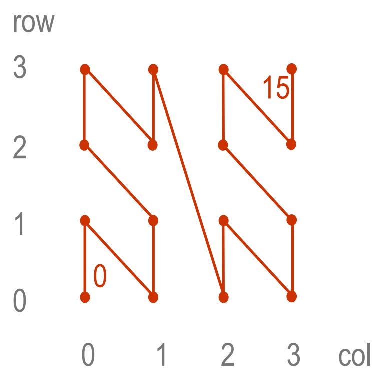


Default nD-array
(non clustering)

Hilbert



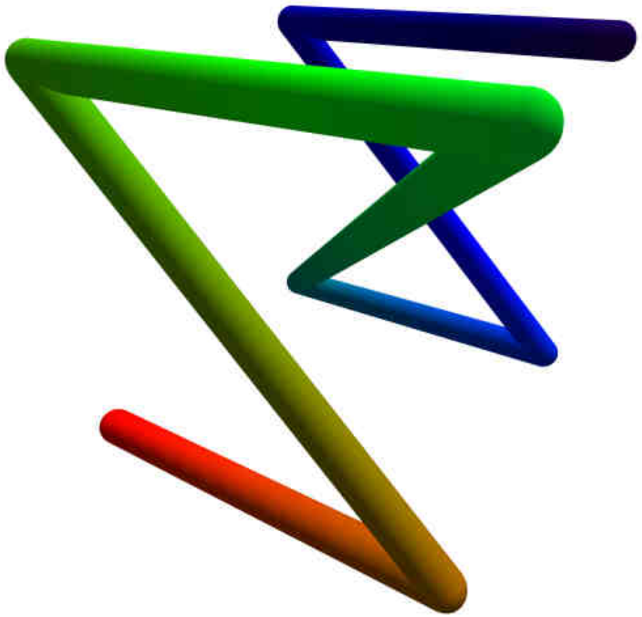
Peano



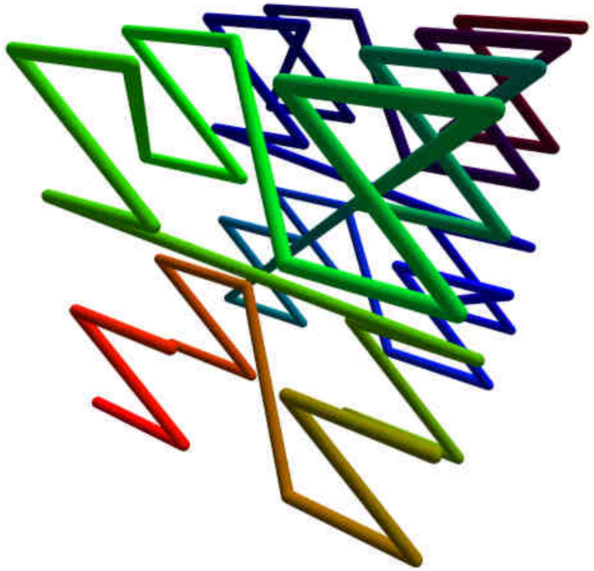
3D Morton curve

illustrations from http://asgerhoedt.dk

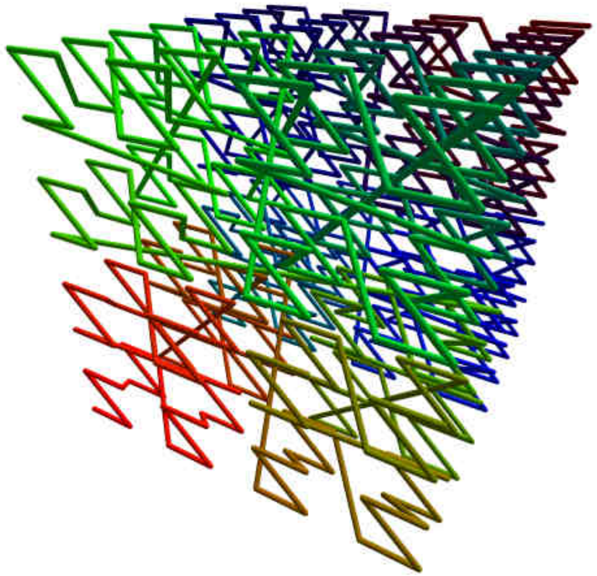
2x2x2



4x4x4



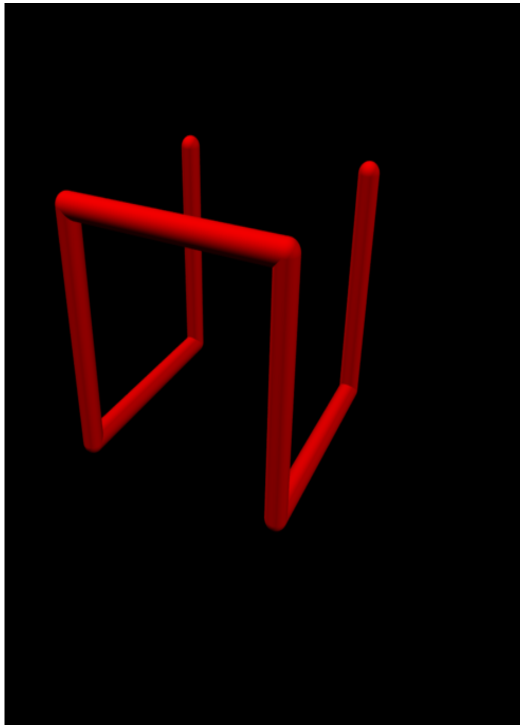
8x8x8

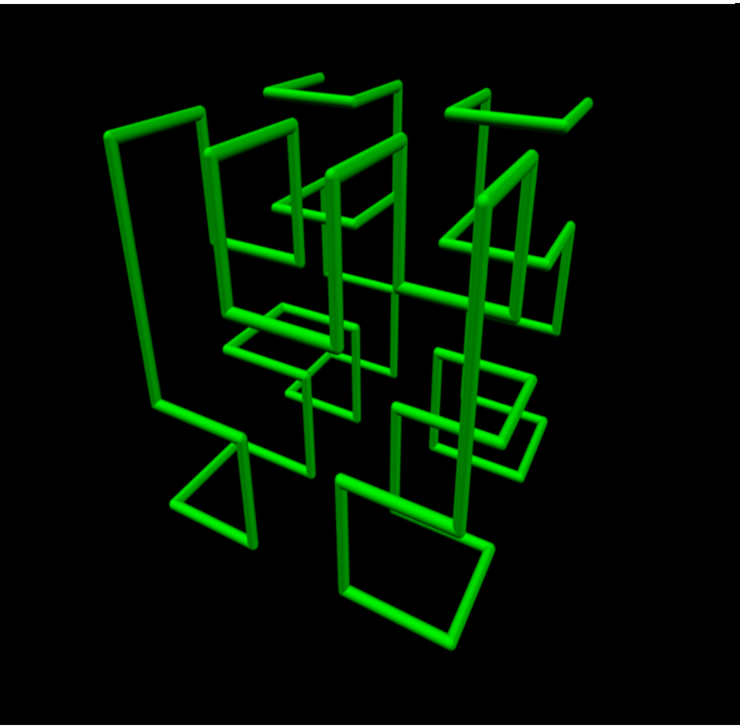


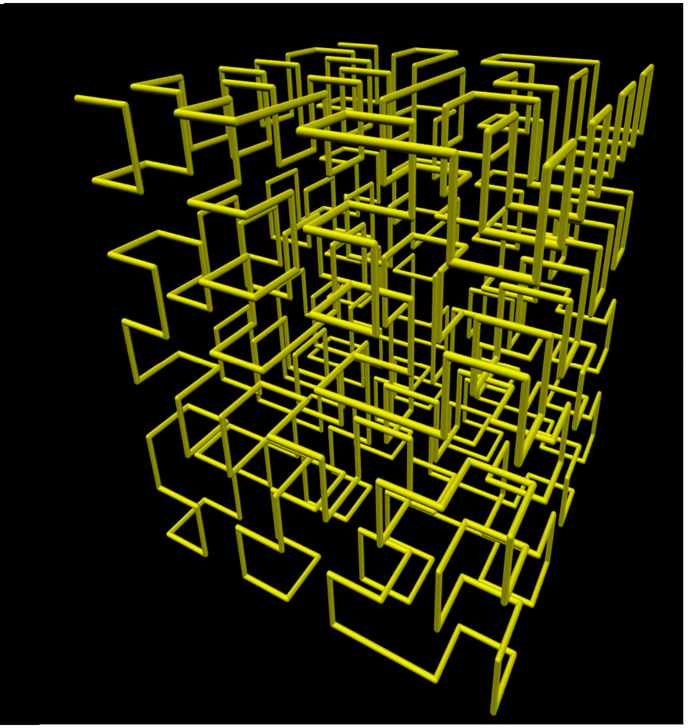
3D Hilbert curve

illustrations from Wikimedia Commons

2x2x2 4x4x4 8x8x8





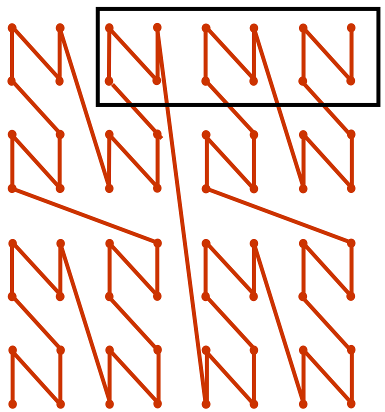


Average number of clusters for all possible range queries

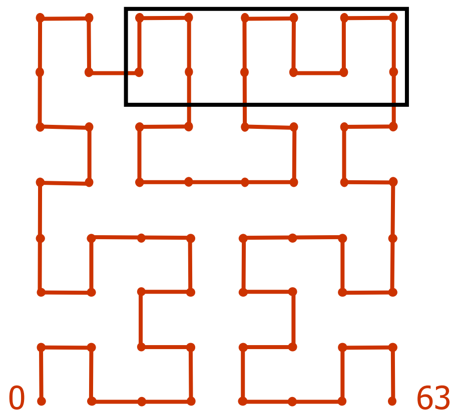
- Faloutsos and Roseman, 1989

N*N GRID	HILBERT	PEANO	I
2*2	1.11	1.22	.11
4*4	1.64	2.16	.52
8*8	2.93	4.41	1.48
16*16	5.60	9.29	3.69

- N=8, number of Clusters for a given range query:



Peano (3 ranges)



Hilbert (2 ranges)

Use Hilbert/Morton code

- two options:
  1. flat table model create b-tree index on SFC code
  2. walk the curve create point cloud blocks

- better flat table model (tested with Oracle):
  - not use the default heap-table, but an indexed organized table IoT (issue with duplicate values → CTAS distinct)
  - no separate index structure needed → more compact, faster

- best (as no redundancy):
  - not x, y, z, time, LoD attributes, but just high-res SFC code (as x, y, z coordinates and time, LoD can be obtained from code)

SQL DDL for index organized table

- Oracle:

```
CREATE TABLE PC_demo (hm_code NUMBER PRIMARY KEY)
    ORGANIZATION INDEX;
```

- PostgreSQL, pseudo solution, not dynamic (better available?):

```
CREATE TABLE PC_demo (hm_code BIGINT PRIMARY KEY);
CLUSTER pc_demo ON pc_demo_pkey;
```

SFC code technique outline

A. define functions for given square/cubic/… nD domain:

1. Compute_Code(point, domain) → Code; (for storage)
2. Overlap_Codes(query_geometry, domain) → Ranges; (for query)

B. add SFC Code during bulk load or afterwards

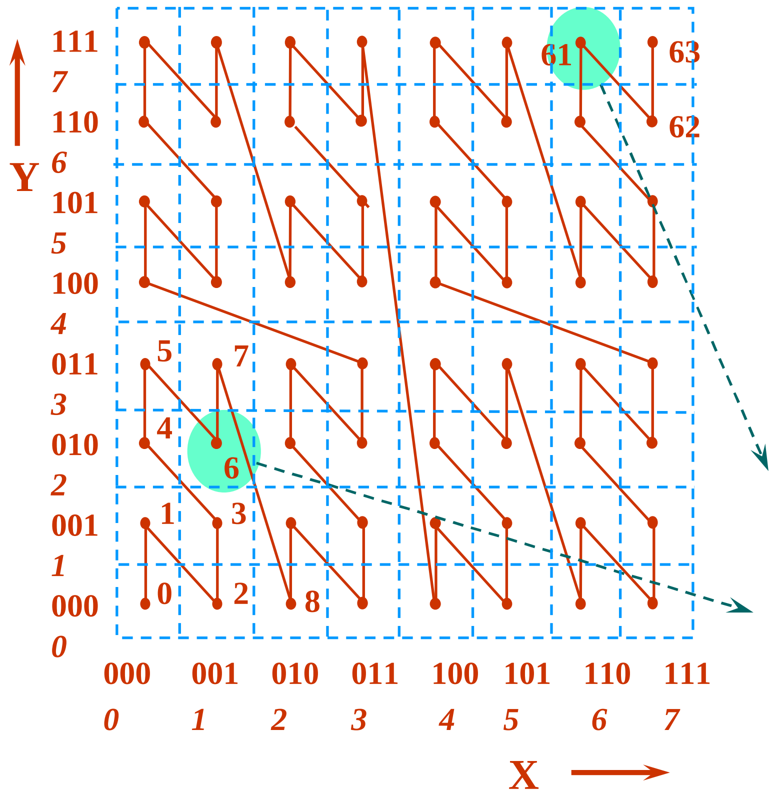
- or even replace point coordinates

C. modify table from default heap to b-tree on Code

SFC code (corresponds to Quadtree in 2D, Octree in 3D, …)

Compute_Code (point, domain) → Morton_code / Peano key / Z-order

- bitwise interleaving x-y coordinates
- also works in higher dimensions (nD)



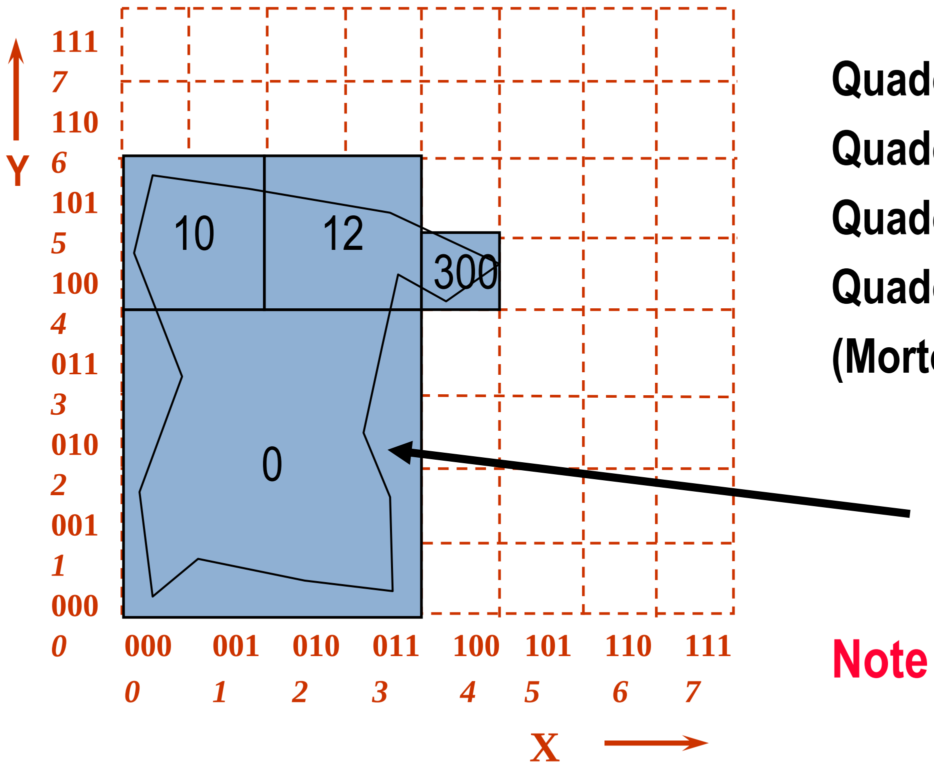
two examples of Morton code:

x= 110, y=111 → xy= 111101 (decimal 61)

x= 001, y=010 → xy= 000110 (decimal 6)

Overlap_Codes (query_geometry, domain) → Morton_code_ranges

- based on concepts of Region Quadtree & Quadcodes
- works for any type of query geometry (point, polyline, polygon)
- also works in 3D (Octree) and higher dimensions



Quadcode 0: Morton range 0-15

Quadcode 10: Morton range 16-19

Quadcode 12: Morton range 24-27

Quadcode 300: Morton range 48-48

(Morton code gaps resp. 0, 4, 20)

query_geometry, polygon

Note : SW=0, NW=1, SE=2, NE=3

Overlap_Codes(), recursive function Pseudo code

```
Overlap_Codes(query_geometry,domain,parent_quad) def
for quad = 0 to 3 do
  quad_domain = Split(domain, quad);
  curr_quad_code = parent_quad+quad;
  case Relation(query_geometry, quad_domain) is
    quad_covered:  write Range(curr_quad_code);
    quad_partly:   Overlap_Codes(query_geometry,
                     quad_domain, curr_quad_code);
    quad_disjoint: done;
```

notes:
- number of quads 2^k (for 2D: 4, for 3D: 8, etc.)
- quad_covered with resolution tolerance
- Range() translates quadcode to Morton range: start-end
- above algorithm writes ranges in sorted order (eg linked list)

Create ranges & post process (glue)

```
Overlap_codes(the_query, the_domain, '');
Glue_ranges(max_ranges);
```

Overlap_codes() creates the sorted ranges (in linked list).
result can be large number of ranges, not pleasant for DBMS
query optimizer gets query with many ranges in where-clause

reduce the number of ranges to 'max_ranges' with Glue_ranges()
(which also adds unwanted codes)

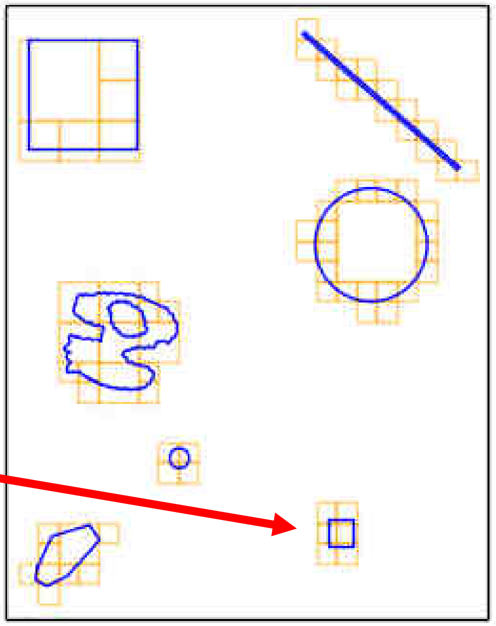
```
Glue_ranges(max_ranges) def
  Num_ranges = Count_ranges();
  Remove_smallest_gaps(num_ranges – max_ranges);
```

notes: - gaps size between two ranges may be 0 (no codes added)
- efficient to create gap histogram by Count_ranges()

Quadcells / ranges and queries

```
CREATE TABLE query_results_1 AS (
SELECT * FROM
  (SELECT x,y,z FROM ahn_flat WHERE
  (hm_code between 1341720113446912 and 1341720117641215) OR
  (hm_code between 1341720126029824 and 1341720134418431) OR
  (hm_code between 1341720310579200 and 1341720314773503) OR
  (hm_code between 1341720474157056 and 1341720478351359) OR
  (hm_code between 1341720482545664 and 1341720503517183) OR
  (hm_code between 1341720671289344 and 1341720675483647) OR
  (hm_code between 1341720679677952 and 1341720683872255)) a
WHERE (x between 85670.0 and 85721.0)
   and (y between 446416.0 and 446469.0))
```

Query 1 (small rectangle)



Drawback of high dimensional SFC?

- nD SFC keys have benefits: space-time-scale (and perhaps even other attributes) in compact organization
- may select on multiple dimensions at same time efficiently

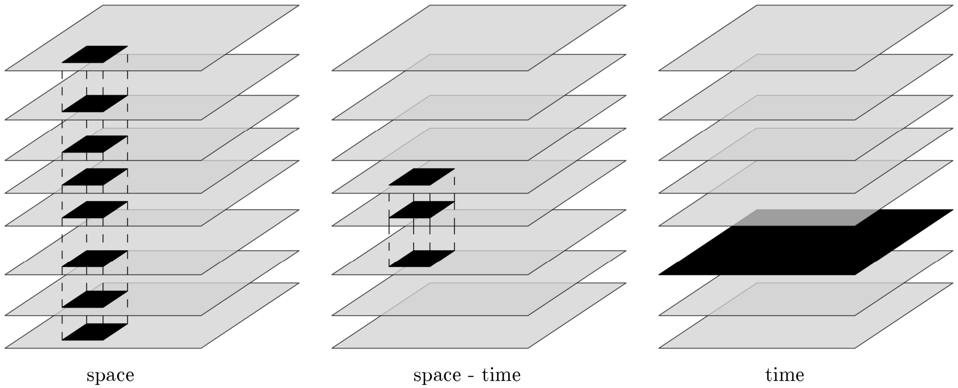
- possible drawbacks of high dimensional point cloud:
  1. need big SFC code (128 bits number or other encoding, like varchar)
  2. if just limited number of dimensions are specified for selection → other dimensions then range form min-to-max: 'tall prisms'
     many (empty?) cells, what are the query performance consequences

- needs further exploration
  (as the relative scaling of dimensions need attention → basis for defining cross-dimension distance → actual grouping/ clustering)

Storage model balancing

'best' organization is dependent on data and queries; e.g.

- asking for time slice (map of one moment in time)
- performing time needle query (one location trough time)
- selecting data for time interval in limited area



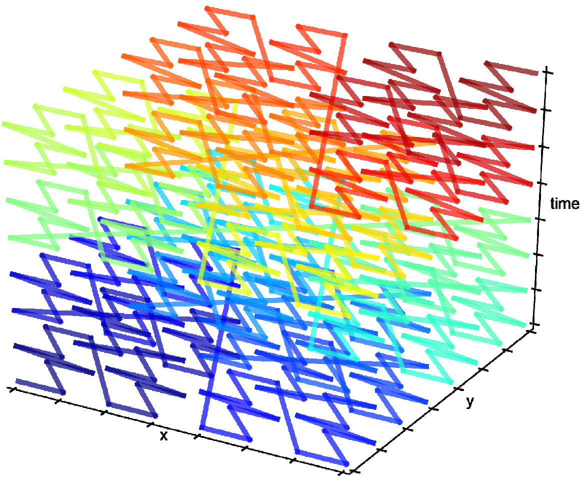
dynamic data optimizing for space/time queries contradicts:

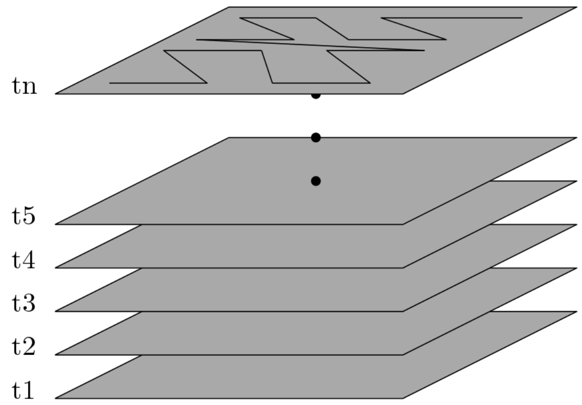
1. Points close in space and time should be stored (to some extent) close in memory for fast spatio-temporal retrieval
2. Already organized points should not be reorganized when inserting new data to achieve fast loading

Storage Model

storage of space and time:

1. integrated space and time approach: space and time have an equal role in the SFC code
2. non-integrated space and time approach: time dominates over space (and used first in organization)





second option easier to add data (dynamic scenario), no reorganization

Overview

- motivation
- scale as dimension
- Functionality
- data management
- standardization
- conclusion

OGC Domain Working Group PC

DWG PC is active for about 1 year
chairs: Stan Tillman (Intergraph), Jan Boehm (UCL), myself

first, conducted Point Cloud Survey (use, tools, needs,…)
received 188 responses:
https://docs.google.com/spreadsheets/d/1_6389UlkIblWyneY5WbbONMMJ-ZiNaeUmcs_iG6olS0/edit?usp=sharing

next, following priorities for the DWG are identified:

1. further collaborate with ASPRS on LAS (OGC community standard)
2. explore HDF5 as format for Point Cloud data
3. interoperable steaming Point Cloud webservices

Standardization of point clouds?

- ISO/OGC spatial data:
  - at abstract/generic level, 2 types of spatial representations: features and coverages
  - at next level (ADT level), 2 types: vector and raster, but perhaps points clouds should be added
  - at implementation/ encoding level, many different formats (for all three data types)

- nD point cloud:
  - points in nD space and not per se limited to x,y,z (n ordinates of point which may also have m attributes)
  - make fit in future ISO 19107
  - note: nD point clouds are very generic; e.g. also cover moving object point data: x,y,z,t (id) series.

Characteristics of possible standard point cloud data type

1. xyz (a lot, use SRS, various base data types: int, float, double,..)
2. attributes per point (e.g. intensity I, color RGB or classification, or imp or observation point-target point or…)
   → correspond conceptually to a higher dimensional point
3. fast access (spatial cohesion) → blocking scheme (in 2D, 3D, …)
4. space efficient storage → compression (exploit spatial cohesion)
5. data pyramid (LoD, multi-scale/vario-scale, perspective) support
6. temporal aspect: time per point (costly) or block (less refined)
7. query accuracies (blocks, refines subsets blocks with/without tolerance value of on 2D, 3D or nD query ranges or geometries)
8. operators/functionality (next slides)
9. options to indicate use of parallel processing

Grouping of functionality

a. loading, specify conversion / organization
b. selections
c. LoD use/access
d. analysis I (not assuming 2D surface in 3D space)
e. analysis II (some assuming a 2D surface in 3D space)
f. conversions (some assuming 2D surface in 3D space)
g. towards reconstruction, classification, segmentation
h. updates: insert, delete, modify

(grouping of functionalities from user requirements)

Loading, specify conversion / organization

- input format
- storage blocks based on which dimensions (2, 3, 4,…)
- data pyramid, block dimensions (level: discrete or continuous)
- compression option (none, lossless, lossy)
- spatial clustering (morton, hilbert,…) within and between blocks
- spatial indexing (rtree, quadtree) within and between blocks
- validation (more format, e.g. no attributes omitted, than any geometry or topological validation; perhaps outlier detection)?

Webservices

- better not try to standardize point clouds at database level (not much support/ partners expected), but rather focus on webservices level (more support/ partners expected)

- there is overlap between WMS, WFS and WCS...

- OGC point cloud DWG should explore if WCS is good start for point cloud services:
  - If so, then analyse if it needs extension
  - If not good starting point, consider a specific WPCS, web point cloud service standards (and perhaps further increase the overlapping family of WMS, WFS, WCS,...)

Overview

- motivation
- scale as dimension
- functionality
- data management
- standardization
- conclusion

Related projects and PhD theses

- Massive Point Clouds (NL): NL eScience Center, Oracle, RWS, Fugro, CWI/MonetDB, TUD Harvest4D (EU): Uni Wien, TUD computer graphics
- IQumulus (EU): UCL, TUD, many more

- Ahn Vu Vo: Spatial Data Storage and processing Strategies for Urban Laser Scanning, PhD thesis, University College Dublin, October 2016.
- Remi Cura: Inverse Procedural Street Modelling from interactive to automatic reconstruction, PhD thesis, University Paris Est (IGN/Thales), September 2016.

Conclusion

- nD-PointClouds as 3rd representation: direct use (storage, analysis, visualization) or conversation to vector or raster
- develop functionality inside the database: encoding and decoding SFC, SFC ranges generation
- investigate different space-time-scale relative dimension representations in hypercube (for surface PC data, but also for more dynamic data: moving object trajectories)
- investigate other SFCs (Morton/Hilbert, less ranges) and/or other organizations (kd-tree, simplex based)
- generation of blocks using the same integrations of space, time and scale (more efficient: less rows, block compression, …)
- standardize streaming, progressive nD-PointCloud web-services

Implementation / code

- Python code Dynamic Point Cloud available at:
  https://github.com/stpsomad/DynamicPCDMS

- C++ code for Morton/Hilbert encode/decode/range generation
  https://github.com/kwan2004/SFCLib

- eScience Massive Point Cloud code (database/ viewer) & docu
  http://pointclouds.nl

- Oracle Database 12c
  (Enterprise Edition Release 12.1.0.1.2 – 64 bit)
  - Use of Index Organized Table (IOT)
  - NUMBER data type for 128 bit Morton/Hilbert keys

Thanks for your attention

- time for questions?

OGC actions in more detail
ASPRS: LAS file format

- American Society for Photogrammetry and Remote Sensing (ASPRS) developed LAS 1.4; https://www.asprs.org/committee-general/laser-las-file-format-exchange-activities.html (with Domain Profile)

- 2 nov'15: OGC and ASPRS to *collaborate* on geospatial standards, invite participation in Point Cloud work; http://www.opengeospatial.org/pressroom/pressreleases/2313

- Ongoing effort to bring the LAS 1.4 point cloud format into the OGC as a community standard

- Attention points (for the future): Attribute flexibility, Other sources than laser, Compression, Organization (clustering)

HDF5 for Point Cloud data

- *Explore capabilities*: test/ benchmarks, assess tools
- Hierarchical Data Format (HDF): file format to store /organize large amounts of data, originally by National Center for Supercomputing Applications (NCSA)

- Hierarchical, filesystem-like data format, 2 types of objects:
  - Datasets: nD arrays
  - Groups: container structures for datasets and other groups

- See HDF5 for point cloud data
  - Chauhan et al (jun' 15): National Geospatial Intelligence Agency (NGA) Sensor Independent Point Cloud (SIPC)
  - Ingram (mar'16): Advanced Point Cloud Format Standards
- Note: also NetCDF 4 (more grid oriented) is based on HDF5

Steaming Point Cloud webservices

- Web services protocol (request/selection, response)
- Data format
- Streaming, ordering, compression
- Caching
- Progressive refinement
- Support LoD's
- Visualization

- Fitting in existing WXS (WCS, WFS) or new service needed (WPCS)?
- Earlier work of OS Geo pointdown
  - https://lists.osgeo.org/mailman/listinfo/pointdown
  - https://github.com/pointdown/protocol