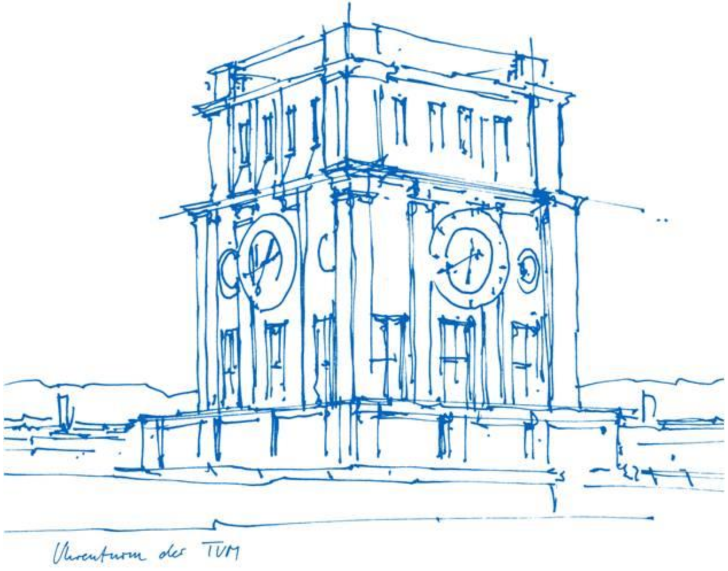
Distributed Point Cloud Data Management and Analysis

Balthasar Teuscher
Technical University of Munich
TUM School of Engineering and Design
Professorship of Big Geospatial Data Management
Bonn, 15. September 2025



Context

- Dissertation topic:
  - *“Distributed Point Cloud Data Management and Analysis”*

- Overarching research question:
  - *“How to manage point cloud data to facilitate efficient and effective visualization and analysis?”*

- Focus on out-of-core computing

Background: Point Cloud Data Management

Data provisioning

- Various text and binary formats such as LAS, LAZ, COPC, and Potree
- Optimized for either simplicity, archival, or visualization

Tooling

- Libraries and CLI such as PDAL, laspy, LAStools and CloudCompare
- Scripting, ETL, batch-processing, analytical workflows
- In-memory or auxiliary indices, single-threaded

Data management

- Monolithic Database Management Systems with dedicated extensions
- Subpar indexing and query capabilities, intransparent storage, slow ingestion and storage amplification

=> Distributed and parallel processing!

Background: Visualization

a) Point Cloud
   – Unordered set of points
b) Layered octree (Gobbetti and Marton, 2004)
   – Points in parent nodes are sampled from child nodes to generate Levels of Detail
c) Dedicated format layout (COPC / Potree)
   – Contains a hierarchical index
d) Analytics format layout (Parquet)
   – No hierarchical relationship between row groups

=> Importance augmentation & partitioning!

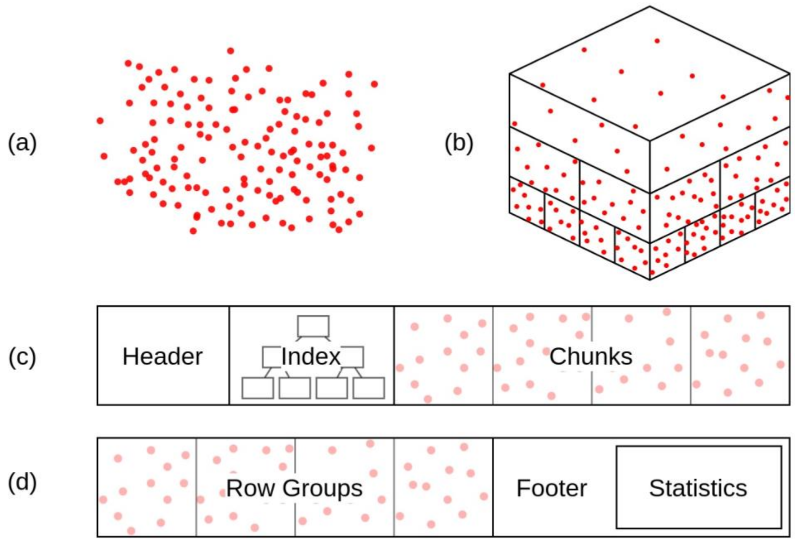


Figure 3. Data organization for point cloud visualization.

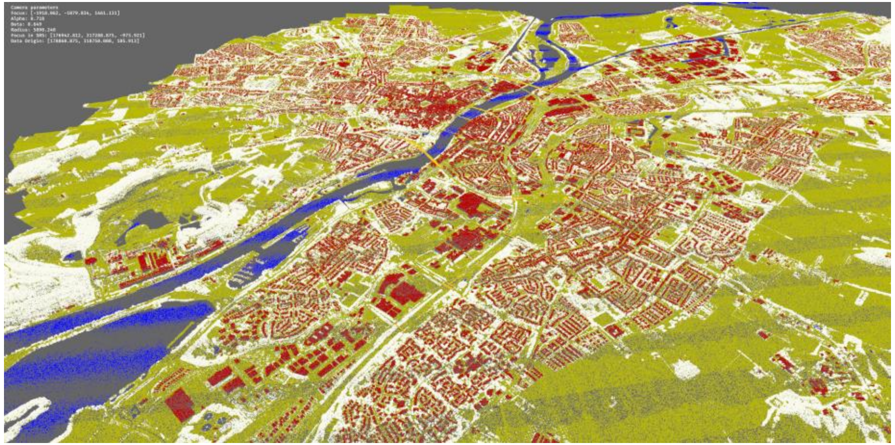
Background: Analysis

- Visualization and analysis generally rely on separate solutions.
- Analytical approaches are becoming more versatile.
  - Increasingly based on machine learning – graph, point set, voxels, grid
- Scalable data management systems are seldom used for modern analytics and visualization.
  - Lack of query capabilities, interoperability
  - Queries are fixed and materialized

=> Data retrieval needs!

Random Data Distribution

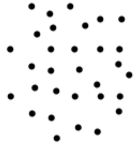
Teuscher, B., & Werner, M. (2024). Random Data Distribution for Efficient Parallel Point Cloud Processing. *AGILE: GIScience Series*, *5*, 15. https://doi.org/10.5194/agile-giss-5-15-2024



Distribution Methodology

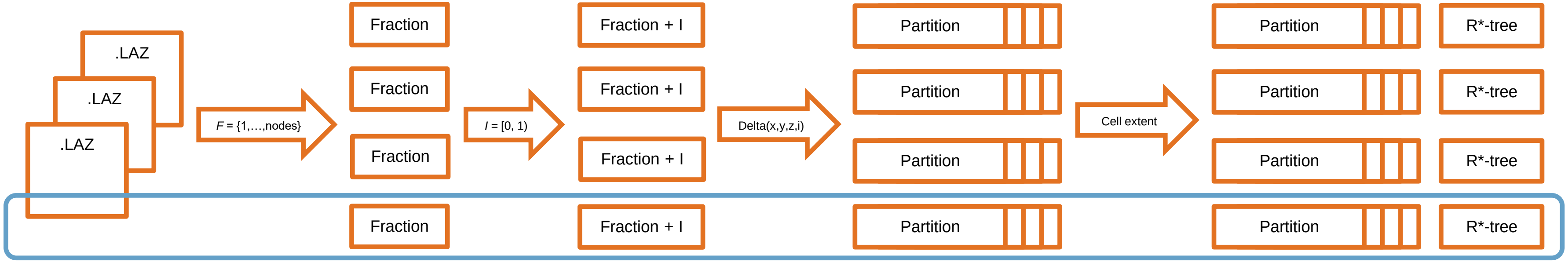
- Coordinator and multiple worker nodes
- Random data distribution
- Random importance value
- Space partitioning (x, y, z, importance)
- R*-tree index of partitions

$P = \{p_1, \dots, p_N \mid p_i \in \mathbb{R}^3\}$



$p_i = \{a_{1i}, \dots, a_{mi}\}$

a_1	...	a_m
a_{11}	...	a_{m1}
...	...	...
a_{1i}	...	a_{mn}



Query Fragmentation

- Range queries over spatial and importance dimensions
  - Full

    /points?bounds={xmin},{ymin},{zmin},0,{xmax},{ymax},{zmax},1

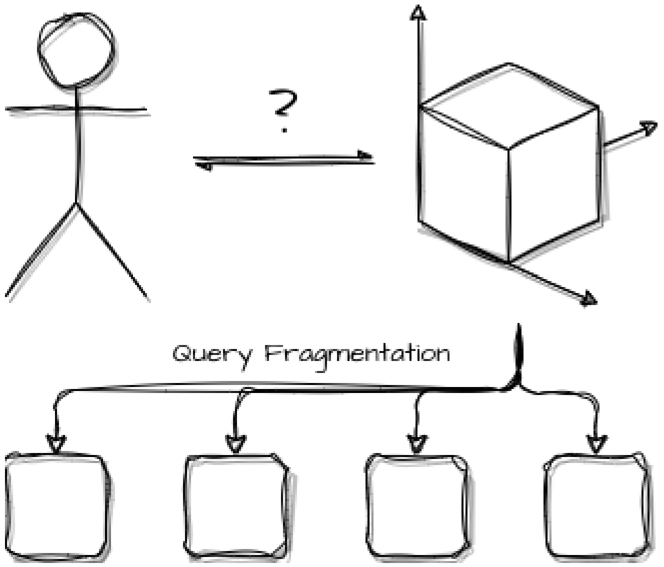
    /points?bounds=174000,315000,0,0,174060,315060,1000,1
  - P-sampling

    /points?bounds={-inf},{-inf},{-inf},0,{+inf},{+inf},{+inf},{p}

    /points?p=0.01
  - Facet-sampling

    /points?bounds={xmin},{ymin},{zmin},0,{xmax},{ymax},{zmax},0.5

    /points?bounds={xmin},{ymin},{zmin},0.5,{xmax},{ymax},{zmax},1



Load and Indexing Performance

File	C_69AZ1.LAZ (743MB, 149 676 342 Points)						
Preprocessing	Load		Index		Total		
	Time (s)	Size (MB)	Time (s)	Size (MB)	Time (s)	Size (MB)	Throughput (Points/s)
Potree	6.2		17.9		24.0	4 141	6 230 284
PCServe	792.0	15 138	318.0	3 362	1110.0	18 500	134 844
Ours (in-memory)	4.1	8 558	1.0	1 267	5.1	9 715	29 114 246
Ours (on-disk)	11.5	7 442	1.9	1 267	13.3	8 709	11 235 275

Range Query Performance

	Dataset size (M Points)	Query type (box,sampling,mixed)	Request duration (ms)	Result size (Points)
PCServe			598	38 981
Ours (in-memory, single node)	150	box	60	58 616
	150	sampling	166	74 343
Ours (on-disk, single node)	150	mixed	196	74 926
Ours (in-memory, 8 workers)	4 500	box	166	59 259
	4 500	sampling	599	45 304

Conlusions

- Random data distribution & materialized importance & 4D range queries work well together
- Simple push down, equal resource usage
- Data representation based on columnar batches proved to be performant
- 4.5 billion in memory – need to go out of core!

Point Cloud Lakehouse

Teuscher, B., & Werner, M. (2025). Point Cloud Data Management for Analytics in a Lakehouse. *AGILE: GIScience Series, 6.* https://doi.org/10.5194/agile-giss-6-47-2025

- NoDB philosophy (Alagiannis et al., 2012)
  - Access raw files directly
  - Combine with a query engine
- Lakehouse pattern (Armbrust, 2021)
  - Cloud environments
  - Decoupling of storage and compute
  - Use optimized formats

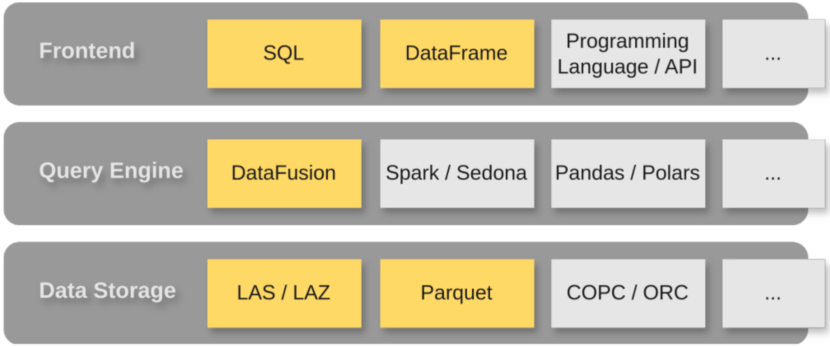


Figure 1. Lakehouse architecture for point cloud data analytics.

Data Formats: Apache Parquet

- Traditional formats
  - None or high compression ratio
  - Record layout (sequential points)
  - Limited access and indexing facilities
- Formats for analytics
  - Inspired by Dremel (Melnik et al., 2010)
  - Partition Attributes Across (Ailamaki et al., 2002)
  - Various encoding and compression schemes
  - Metadata in footer with min-max statistics
- Memory representations
  - Zero copy, self-describing, columnar

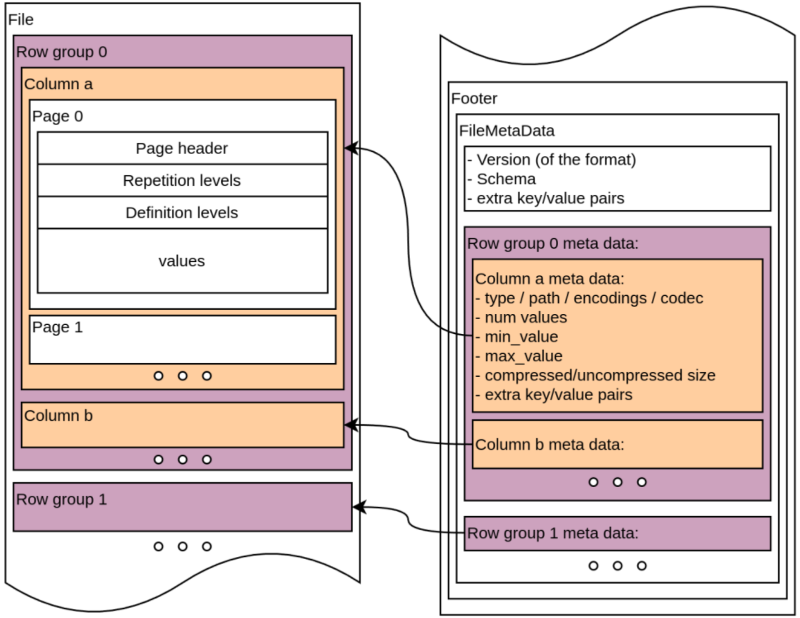


Figure 2. The Apache Parquet file format layout.

Use Case: Visualization

- Point cloud
  – Unordered set of points
- Layered octree (Gobbetti and Marton, 2004)
  – Points in parent nodes are sampled from child nodes to generate levels of detail
- Dedicated format layout (COPC)
  – Contains hierarchical index
- Analytics format layout (Parquet)
  – No hierarchical relationship between row groups

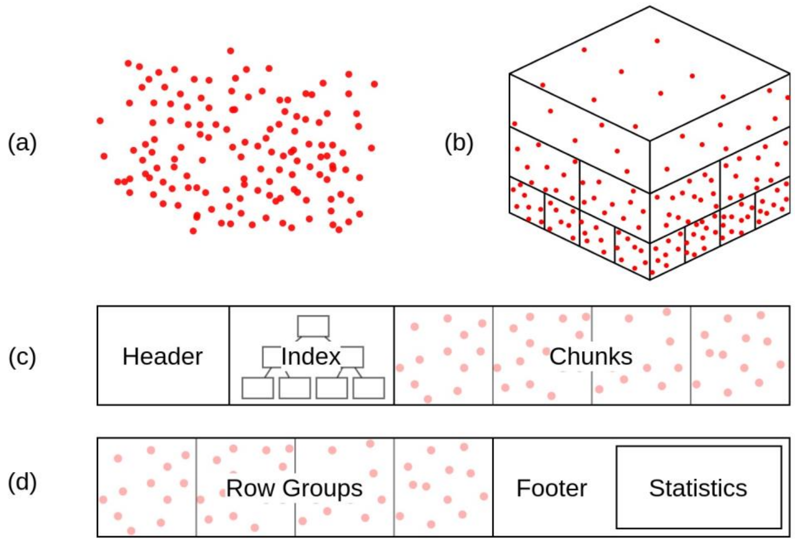


Figure 3. Data organization for point cloud visualization.

Importance Augmentation: Continuous Level of Detail

- Continuous importance augmentation with a random value in the range [0, 1) per point.
  - Given a point cloud with n points, finding out the depth d of the tree respecting a certain amount of points per cell m can be done naively by

$$d = \lceil \log_p(\frac{n}{m}) \rceil$$

    where p is the number of child nodes.
  - From this, the importance range for a certain level can be derived and added as a dimension to the target bounding volumes, which then can be used to sample points to the respective nodes.

Partitioning: Windowed Bounding Volumes

- Windowed bounding volumes partitioning
  - Split the extent of the point cloud into a set of reading windows W and target partitions BV
  - Iteratively process windows by loading overlapping points
  - Mapped to the overlapping bounding volumes in memory
  - Write to disk once they do not intersect with remaining reading windows
- Arbitrary space partitioning
- Parallel and out-of-core through asynchronous stream processing
- Generic over the number of dimensions and hierarchical structure

Algorithm 1 Windowed bounding volumes partitioning.

procedure PARTITION(P) ▷ set of points P
  $E \leftarrow$ extent of P
  $BV \leftarrow$ split E into a set of bounding volumes
  $W \leftarrow$ split E into a set of reading windows
  $C \leftarrow \emptyset$ ▷ cash (in-memory)
  for all $w \in W$ **do** ▷ iterate over windows
    $P_w \leftarrow$ points contained by w
    $BV_w \leftarrow bv \in BV$ intersecting w
    for all $bv \in BV_w$ **do** ▷ map points to volumes
      $P_{bv} \leftarrow \{p \mid p \in P_w\}$ intersecting bv
      $C[bv] \leftarrow C[bv] \cup P_{bv}$
    end for
    $W \leftarrow W \setminus w$ ▷ remove current window
    for all $bv \in BV_w$ **do**
      if bv disjoint W **then**
        evict $C[bv]$ ▷ write to disk
      end if
    end for
  end for
end procedure

Evaluation: Storage Footprint

- Smaller than LAS
  - Even with resolved coordinates and materialized importance
- About 1.5 times larger than LAZ
  - Compression is optimized for speed, not compression ratio
- Comparable to Potree

Table 1. Storage footprint of various coordinate encodings compared relative to LAZ.

Format	Coords	cLoI	Compression	Size (rel.)
LAS	i32[a]	-	-	4.25
LAZ	i32[a]	-	LASzip	1.00
Parquet	i32[b]	-	-	2.76
Parquet	f64		-	3.71
Parquet	f64	f32	-	4.26
Parquet	i32[b]	-	zstd(level3)	1.56
Parquet	f64	-	zstd(level3)	1.59
Parquet	f64	f32	zstd(level3)	2.10
Potree	(missing)	-	-	4.12

[a] Scale and offset in header.
[b] Scale and offset per point.

Evaluation: Data Loading and Indexing

- Instant DBMS with LAS/LAZ files
  - No conversion
- Extracting statistics is a complete scan
  - Decompression is bottleneck
- Conversion to Parquet without partitioning
  - Includes reading, conversion, and writing
- Partitioning with different schemes
  - Scheme is not influencing much
  - Performance comparable to Potree

Table 2. Data loading performance of ~2B points.

Format	Partitioning	Statistics	Time	Throughput
LAZ	-	file	0.0s	-
LAZ	-	chunk	90.6s	22.0MP/s
Parquet	-	page	209.3s	9.5MP/s
Parquet	grid(xy)	page	375.6s	5.3MP/s
Parquet	grid(xyi)	page	358.6s	5.6MP/s
Parquet	quadtree	page	398.2s	5.0MP/s
Potree	octree	node	424.5s	4.7MP/s

Evaluation: Query Performance

Table 3. Query performance of selected queries on ∼200M and ∼2B points; small rectangle (S_RECT), medium rectangle (M_RECT), nearest neighbours (NN_1000) and importance (I_700k).

Dataset	AHN3 extract with ∼200M				AHN3 extract with ∼2B			
Query	S_RECT	M_RECT	NN_1000	I_700k	S_RECT	M_RECT	NN_1000	I_700k
Points returned	74k	726k	1k	700k	74k	726k	1k	700k
Selectivity	0.5‰	4.0‰	0.025‰	3.50‰	0.037‰	0.363‰	0.002‰	0.350‰
LAZ	12.976s	13.003s	13.599s	13.787s	23.304s	23.295s	23.312s	90.415s
LAZ + statistics	0.427s	0.848s	0.378s	13.359s	**0.194s**	0.514s	**0.104s**	114.891s
Parquet (convert)	0.212s	0.282s	0.219s	0.819s	0.311s	0.364s	0.537s	11.130s
Parquet (grid xy)	**0.126s**	**0.173s**	**0.123s**	0.773s	0.288s	**0.321s**	0.313s	12.185s
Parquet (grid xyi)	0.192s	0.264s	0.151s	0.460s	0.328s	0.362s	0.446s	1.305s
Parquet (quadtree)	0.149s	0.221s	0.130s	**0.167s**	0.311s	0.376s	0.333s	**0.488s**

Evaluation: Interactive Visualization

- Below 300ms for typical visualization queries
  - Result set is about 350k points per query

- R*-tree index can speed it up even more
  - The query engine is still improving

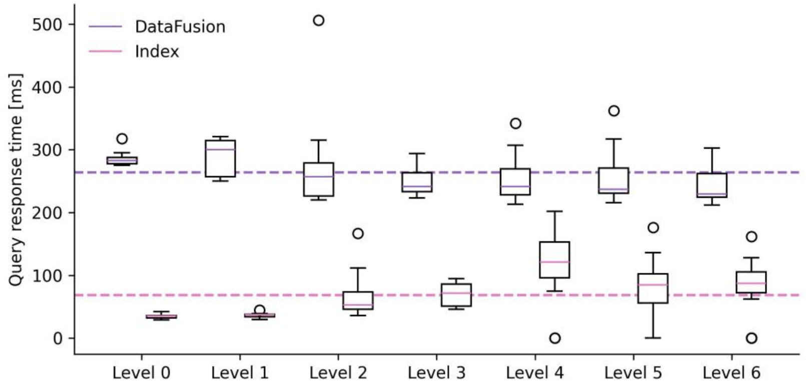


Figure 4. Vizualization query performance on different levels.

Conclusions

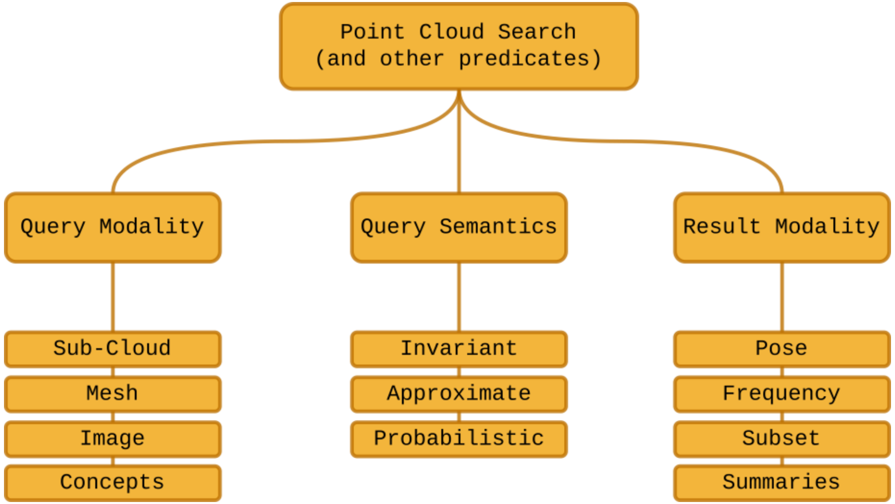
Offers scalable and performant analytics capabilities …
… while simultaneously supporting visualization

- No active components
- Ideal for elastic computing
- No domain-specific adaptations
- Access to a large ecosystem

Taxonomy

Teuscher, B., Walther, P., Wang, J., & Werner, M. (2025). A Taxonomy of Point Cloud Search. *3DGeoInfo (forthcoming).*

- Query Modality
  - How is the query expressed? - focus on the geometric aspect of queries, specifically their encoding
- Query Semantics
  - How is the query interpreted?
- Result Modality
  - How is the query answered?



Evaluation

>20 example queries covering neighborhood, range, sampling, visualization, registration, aggregation, and semantic retrieval tasks.

Query	Query Modality	Query Semantic	Result Modality	Use Cases / Prominent Application
k-Nearest Neighbor (kNN)	Sub-Cloud	I	Subset / Summary	Neighbourhood descriptors and features
Approximate kNN	Sub-Cloud	A	Subset / Summary	Neighbourhood descriptors and features
Radius	Sub-Cloud	I, (A, P)	Subset / Summary	Neighbourhood descriptors and features
Vector	Sub-Cloud	P, (I, A)	Subset / Summary	Similarity search
Range / Box	Mesh	I, (A, P)	Subset	Spatio-Temporal filtering
Slicing	(Mesh)	I, (A, P)	Subset	Spatio-Temporal subsetting
Polygon / Cone / Solid	Mesh	I, (A, P)	Subset	Manifold intersection, Topological analysis
Discrete Sampling	Any	I, (A)	Subset	Environmental Data Retrieval
Random Sampling	Any	P	Subset	continuous Level of Detail, Thinning, Sampling
Poisson Disk Sampling	Any	P	Subset	Disk-/blue noise sampling, Downsampling
Furthest Points Sampling (FPS)	Any	I	Subset	Downsamling, Representative points
View Frustum	Mesh	I, (P)	Subset	Visibility analysis (Field of View), Visualization
Ray	Sub-Cloud	I, (P)	Pose / Frequency	Rendering, Surface intersection, Visibility analysis
Trajectory	Sub-Cloud	I, (P)	Subset	Movement analysis, Collision detection
Co-registration	Sub-Cloud	Any	Pose	Iterative closest point (ICP) / Alignment
Point set registration	Sub-Cloud	Any	Pose / Frequency	Part search, Alignment
Image registration	Image	A, (P)	Pose	Camera position and orientation / Photogrammetry
Aggregate	Any	Any	Summary	Point density, Centroid, Representative points/values
Window	Any	I	Summary	Spatio-temporal change detection
Join	Any	I	Subset	Elevation, Data fusion, Topological analysis
Attribute	Sub-Cloud	I, (A, P)	Subset	Filter by class or label, Semantic search
Text	Concept	P, (I, A)	Any	Natural language, Semantic search

Search Space

- What is a sub-cloud?
- Searchable space vs. indexable space
  - What is transparent and what is opaque to the system?
- Examples
  - KV store
  - pgpointcloud
  - LAS

$$P \subseteq \prod_{i=1}^{m} D_i,$$

Representations

- Various possible point cloud representations:
  - raw point set
  - voxels,
  - mesh,
  - graph,
  - …
- Tooling and systems are often specialized for a single one
- Can be considered another facet to the taxonomy

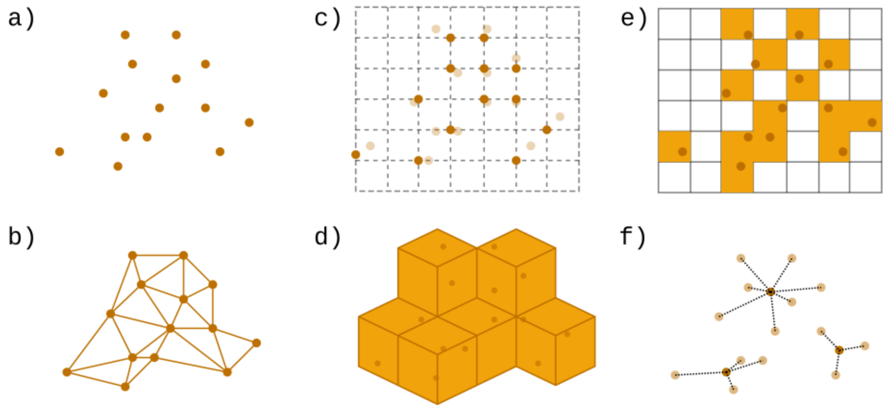


Illustration of various point cloud modalities: a) raw points, b) mesh, c) grid rounded, d) voxel, e) raster, f) skeleton (graph).

Recommendations

- Even though spatial attributes are prominent, allow equal treatment of all attributes for query support.
- Facilitate dynamic schema evolution to support semantic enrichment by adding attributes transparently and readily queryable.
- Support heterogeneous data representations and seamless transformations between them.
- Assume probabilistic and approximate query semantics as the default to incorporate scalability.
- Consider a multi-representation data model tightly integrated with indexing capabilities.

Outlook

- Towards multi-user real-time visualization and editing.
  - Transaction capabilities
  - Investigate suitable table formats or create our own
- Showcase analytical workflows.
  - Support multiple representations
  - Semantic augmentation
  - Spatio-temporal change detection
  - …
- Contextual importance
  - Learned importance based on reconstructive loss with ray tracing

Table 1
Multi-representation point cloud data model schema in a flat table design.

Modality	Attribute	Description
Point	pid	Unique identifier
Point	x	First coordinate
Point	y	Second coordinate
Point	z	Third coordinate
Point	i	Importance
Point	virtual	Flag for virtual points
Point	a_i	Additional point attributes
Graph	gids	Unique graph identifier
Graph	endpoint	Second edge vertex (pid)
Graph	directed	Direction flag (un-/directed)
Graph	weight	Edge weight
Graph	e_i	Additional edge attributes
Mesh	mids	Unique mesh identifier
Mesh	v2_pid	Second triangle vertex (pid)
Mesh	v3_pid	Third triangle vertex (pid)
Mesh	m_i	Additional mesh attributes
Mesh	t_i	Additional triangle attributes
Voxel	code	Well-known voxel code
Voxel	v_x	First voxel coordinate
Voxel	v_y	Second voxel coordinate
Voxel	v_z	Third voxel coordinate
Voxel	size	Voxel size
Voxel	v_i	Additional voxel attributes

Thank you!