Jackpine3D: A Benchmark For Evaluating 3D Spatial Database Features

Mohammadmasoud Shabanijou[1], Zhuliang Jia[1], Suprio Ray[1], Rongxing Lu[2], Pulei Xiong[3]
[1]University of New Brunswick, Fredericton, Canada
Email: {shabani.m, zhuliang.jia, sray}@unb.ca
[2]Queen's University, Kingston, Canada
Email: rongxing.lu@queensu.ca
[3]National Research Council (NRC), Canada
Email: pulei.xiong@nrc-cnrc.gc.ca

***Abstract*—In recent years, due to the advancements in 3D data technologies, and the rise in 3D spatial applications, such as metaverse and digital twins, there has been a growing interest in efficient support of 3D spatial features in database management systems (DBMS). Processing 3D spatial queries can be significantly more complex and computationally intensive than traditional 2D spatial queries. Although a few spatial database benchmarks exist, they support 2D spatial data. Hence, there is a growing need to develop a benchmark to evaluate 3D spatial features.**

To address the aforementioned need, we have developed Jackpine3D, a 3D spatial database benchmark. The main goal of this benchmark is to evaluate the capability of different database systems in processing 3D spatial queries. The benchmark includes a Micro benchmark suite that consists of basic 3D spatial operations, and a Macro benchmark consisting of a few real-world spatial applications. With our benchmark, we have evaluated three database systems and compared their performance. We also identify several gaps for future researchers to address.

I. Introduction

The volume of 3D spatial data has been experiencing rapid growth in recent years. Managing, maintaining, and analyzing this data has become an important consideration [1]. This growth is fueled by advancements in 3D technologies, such as LiDAR and photogrammetry, and their wide applications in urban development, environmental monitoring, and autonomous systems.

Database management systems (DBMS) play a vital role in handling and maintaining data in enterprises and various organizations. There are several features that make database systems attractive, which include logical and physical data independence, a popular query language SQL, and support for various functionalities. Although most databases support querying spatial data, this is usually limited to 2D spatial data. To our knowledge, only a few database systems support 3D spatial querying.

In the past, researchers attempted to compare the performance of different database systems by conducting benchmarking studies. One of these benchmarking efforts was Jackpine [2], a spatial database benchmark to compare the performance of different relational database systems, including both Micro benchmark and Macro benchmark workloads. However, Jackpine did not support 3D spatial data. To our knowledge, there is no existing benchmark to evaluate database systems on 3D spatial queries. To fill this gap, we have developed Jackpine3D, a 3D spatial database benchmark. Jackpine3D shares similar objectives as the original Jackpine benchmark, such as extendability, and its system architecture follows a similar organization.

Since Jackpine3D is a spatial database benchmark, its workloads consist of SQL queries involving 3D spatial operations. The performance of each database system, being benchmarked, is quantified by the execution time of these queries. There are two categories of workloads in the benchmark: Micro and Macro benchmark workloads. The Micro benchmark queries include sets of queries that cover different topological spatial relations and spatial data analysis operations over different 3D spatial data types. The Macro benchmark queries are designed to have real-world applications and serve benchmarking purposes in three application scenarios: Urban planning, Medical analysis, and Metaverse. A review of the current literature on topics, such as smart cities and IoT, Digital twins, and medical analysis was made to determine suitable queries for these applications. The queries were designed with two main objectives: covering 3D spatial functionalities and representing real-world 3D spatial applications. These application scenarios are extendable, as researchers and developers can add new queries or even new scenarios based on their needs.

The spatial data type of the datasets used in Micro benchmark and Macro benchmark workloads is primarily polyhedron, which enables representing solid objects precisely. Besides polyhedron, point cloud, and voxel are two other popular 3D spatial data types. To explore different types of 3D spatial data representations, including polyhedron, point cloud, and voxel, Jackpine3D also includes a workload.

During the design and development of this benchmark, two important considerations were extendibility and comprehensiveness. Extendibility is important because 3D spatial data management is still an emerging field, and it is likely that more database systems with 3D support will be introduced in the near future. The benchmark is also intended to be comprehensive so that as many 3D spatial functionalities as possible across different database systems could be assessed. To that end, we conducted a survey of 3D spatial features in relational databases. We found that 3D spatial support across different database systems is quite limited. Our observations confirm recent findings reported by Belussi et al. [3] and Salleh et al. [4]. There are several database systems that supports 3D data types, however, they provide a very limited set of operations on them. Topological relations are used to describe spatial

association between objects, such as intersects, overlaps, touches and within. Although most databases support 2D spatial topological relations, they offer limited or no support for 3D topological relations. Among the databases we explored, PostgreSQL with PostGIS extension and DB-X (a major commercial DBMS) offered support for many 3D spatial features. Other database systems either did not support 3D functionality or had very limited or incomplete support for 3D. For example, IBM Db2 did not have full native 3D functionality, as it supported 2.5D. To our understanding, SQL Server supports 3D point data type, but there is no support for 3D spatial topological relations.

Jackpine3D Micro benchmark and Macro benchmark benchmarks leverage several real-world and synthetic 3D spatial datasets, details of which are in Section III-B1. We chose to evaluate three relational databases with Jackpine3D benchmark: DB-X, PostgreSQL (with PostGIS) and SpatialLite. Despite its limited 3D spatial support, SpatialLite was included because it extends the SQLite database with spatial functionalities, and SQLite is the most widely deployed embedded database engine [5]. The experimental evaluation of the database systems with Jackpine3D benchmark contained some interesting findings, which are presented in Section V. Based on our evaluation results, there was no single database system that performed better than other systems. These results may provide guidance to the database systems developers regarding opportunities for improvement.

We also conducted a system level analysis of 3D spatial query execution in PostgreSQL, in terms of TLB (Translation Lookaside Buffer) and LLC (Last-Level Cache) cache misses, and provided a detailed breakdown of execution times spent in the query engine. Based on our benchmarking experience, we have outlined a number of future research directions, which would be useful for researchers and practitioners.

The contributions of our paper are as follows.

- We developed Jackpine3D[1], the first benchmark for evaluating 3D spatial features in relations databases, supporting different spatial data types including polyhedron, point clouds, and voxels.
- We designed comprehensive Micro benchmark queries and realistic Macro benchmark scenarios (urban planning, medical, metaverse).
- We evaluated PostgreSQL, DB-X, and SpatiaLite with detailed performance comparisons. We also analyzed memory performance of PostgreSQL.
- We identified and presented several future research directions.

The rest of the paper is organized as follows. Section II discusses related work. Section III provides the details about our benchmark, including data model, datasets and Micro benchmark and Macro benchmark benchmark workloads. Section IV contains information on the experimental setup and in Section V, we discuss the results of the experiments. We present some future research directions for researchers in Section VI. Finally, Section VII relates our conclusions.

[1]Available at https://bigdata.cs.unb.ca/projects/jackpine3d/

II. Related Work

As the importance of 3D processing grows rapidly, so have the research efforts in database systems and spatial query processing. This section presents relevant research in these topics: 3D spatial databases, benchmarking studies, and specialized 3D query processing systems.

A. 3D Spatial Databases

Research in 3D spatial databases has witnessed a significant evolution from 2D traditional databases with many developments to handle complex geometries and topological relationships. Ramsey [6] conducted a study of PostGIS, an extension of PostgreSQL that enhances its capabilities to support 3D geometries and point clouds. According to Kothari et al. [7], Oracle Spatial is a capable 3D spatial database engine to handle complex spatial queries. They provided a comparative study of Quadtree and R-tree indexes in Oracle Spatial.

B. Benchmarking Studies

Researchers proposed multiple database benchmarking platforms. These benchmarking studies differ in their methodology and how they evaluate the different databases. Two well-known benchmarking studies are TPC benchmarks [8] and DBT [9]. The Transaction Processing Performance Council (TPC) is a non-profit organization that developed benchmarks tailored to different workload domains, and it has become the industry standard.

In terms of spatial database benchmarking, there have been several contributions, however, these spatial database benchmarks mainly focused on 2D operations. B. Simion et al. discussed how spatial data processing differs remarkably from regular data processing [10]. One of the best-known benchmarking studies on spatial data was SEQUOIA 2000 [11], but this benchmark was studied for 2D data and investigated raster data. Another benchmark for vector datasets was VESPA [12], which compared PostgreSQL and the Rock & Roll deductive object-oriented database. This benchmark contained many queries, but some doubts about its ability to cover spatial functionalities remained. Jackpine benchmark [2] provided a comprehensive assessment for spatial databases but it addressed 2D spatial data. Van Oosterom et al. [13] offered a benchmark for point cloud data management systems focused on big LiDAR datasets.

C. Specialized 3D Query Processing

There is a growing need for efficient processing of 3D spatial queries because of the wide availability of high-resolution imaging instruments. Notably, the application of 3D spatial data can be found in numerous industries, including mineral exploration [14], 3D geographic information systems (GIS) [15], and 3D mapping and navigation [15], and recently augmented reality applications [16]. Compared to 2D data projections of 3D objects, 3D data offers a more precise representation, though it introduces significantly more complexity in terms of shape details and surface-based 3D modeling [17]. Traditional spatial query processing systems follow the Filter-Refinement paradigm [18], which uses approximations of spatial objects to generate an initial set of candidate results in the

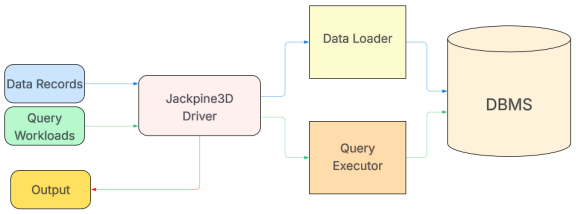

Fig. 1: An overview of Jackpine3D system architecture.

Filter phase and then these candidates are evaluated against their precise geometries to produce accurate answers in the Refinement phase.

Several strategies have been introduced to alleviate the intensive computation involved in the refinement stage at the intra-geometry level. One such strategy is to construct spatial indexes on the polyhedron primitives, including structures like R-tree [19], Orientable Bounding Box trees (OBB-trees) [20], to help determine spatial relations between individual objects.

III. Jackpine3D Benchmark

With Jackpine3D, we aimed to create a benchmark that could be used to evaluate the performance of different databases across different 3D spatial operations by executing SQL queries involving spatial analysis and topological functions. We also attempted to make the benchmarking process more engaging to users and researchers by incorporating real-world application scenarios. The workloads in Jackpine3D are categorized into two groups: Micro and Macro benchmark workloads, consisting of SQL queries involving 3D spatial operations. More details about them are provided in sections III-B and III-C respectively.

An overview of the system architecture is provided in Figure 1. Data records and query workloads get passed on to the Jackpine3D Driver. From there the next components are Data Loader that loads the data into each DBMS, and Query Executor that executes the specific queries by connecting to each DBMS. After execution of each query, Jackpine3D Driver returns the query results. After the completion of all queries an output report can be generated.

A. Spatial Data Models

An important consideration for benchmarking is the supported data types. Spatial data types are based on geometric primitives, which encapsulate the conceptual representation of the underlying spatial objects. Data systems with geospatial functionalities typically support vector and raster representations. 2D spatial vector data types include point (1D), polyline and polygon (2D). 2D spatial raster data constitute an organization of the spatial domain into a grid of cells, with each cell containing a value for a specific feature, such as temperature. While the aforementioned 2D spatial data types have been widely adopted, efficient representation of 3D spatial data is still an ongoing research topic [21]. Three popular 3D spatial data types are: point cloud, polyhedron and voxel. Voxel is the 3D counterpart of 2D spatial raster data, whereas point cloud and polyhedral surface can be considered as 3D spatial vector data types.

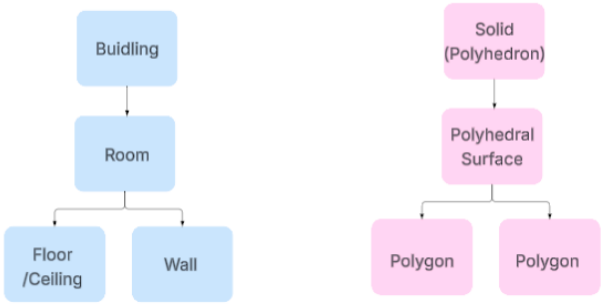

Fig. 2: An illustration of different building components and associated data types.

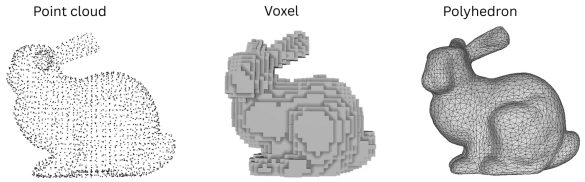

Fig. 3: An example of different 3D spatial data representations [25]

The data type of our primary Micro benchmark dataset is polyhedron (composed of polyhedral surfaces). Figure 3 shows an illustration of different spatial data types. Next, we briefly explain the three 3D spatial data types.

1) Polyhedron: A polyhedron is a 3D geometric structure composed of flat polygonal faces called *polyhedral surface* joined along edges [22]. A *Triangulated Irregular Network (TIN)* is a specific type of polyhedral surface composed of triangles. Polyhedron is one of the most precise ways to represent solid objects in 3D space. In this representation, explicit surface geometry and connectivity is encoded that enables accurate modeling of solid boundaries and volume. This representation is useful in applications that require well-defined topology and surfaces, such as intersection detection. Figure 2 shows a simplified model of different building components and their associated data types, where the data type of the top level object is polyhedron.

2) Point Cloud: A 3D point cloud is a collection of data points in a three-dimensional coordinate system, representing the 3D shape or external surface of an object. Point clouds are usually generated using 3D scanning technologies, such as LiDAR, or photogrammetry. Due to its simplicity, it is becoming a popular representation of 3D data in many applications.

3) Voxel: Voxels are volumetric pixels organized in a regular grid in 3D space and do not have gaps among them [23]. Similar to point clouds, there are also many applications for voxels. Applications for voxels include analysis of medical and scientific data [24] and visualization of terrain in games and simulations [23].

B. Micro Benchmark

In this section, we discuss the Micro benchmark workloads. The queries in the Micro benchmark are presented in Table I. The Micro benchmark workloads are intended to evaluate 3D spatial analysis operations, as well as 3D spatial topological relations. Analysis operations are

TABLE I: Micro benchmark queries

Name	Query	Description
Join Queries		
Building3DIntersectsLine	Check if 3D geometries of Roads intersect with Buildings	This query involves the Roads table and the Buildings table
Building3DIntersectsArea	Check if 3D geometries of Arealandmarks intersect with Buildings	This query involves the Arealandmarks table and the Buildings table
Building3DDistanceLine	Calculate the 3d distance of Roads with Buildings	This query involves the Roads table and the Buildings table
Building3DDistanceArea	Calculate the 3d distance of Arealandmarks with Buildings	This query involves the Arealandmarks table and the Buildings table
Building3DDistanceWithinBuilding	Checks if there are Buildings within 10 units distance of other Buildings	This query involves the Buildings table
Analysis Queries		
3D Bounding Box	Calculate the 3D Bounding box of valid geometries	This query involves the Arealandmarks table
Perimeter3D	Calculate the perimeter of valid 3D geometries	This query involves the Arealandmarks table
Length3D	Calculate the Length of valid 3D geometries	This query involves the Arealandmarks table
Convex Hull	Calculate the Length of valid 3D geometries	This query involves the Arealandmarks table
Dimensions	Returns the dimensions of valid 3D geometries	This query involves the Arealandmarks table

TABLE II: Macro benchmark queries

Name (Scenario)	Query	Description
Subway Station Location (Urban Planning)	Finding suitable locations for subway station	Selects 5 sampled subway locations with the closest nearby locations within 25m based on 3D distance.
Emergency Routes (Urban Planning)	Finding suitable evacuation routes in case of emergency	Finds 5 sampled locations with lowest building density and largest clearance radius within 100m for optimal evacuation routes.
Future Expansion (Urban Planning)	Finding Idea areas for city expansion	Finds 5 sampled locations with low building density ($\leq$ 50) and their distances to nearby buildings within 500m for city expansion analysis.
Cancerous Analysis-Intersection (Medical Analysis)	Intersections of healthy and cancerous cells	Analyzes all intersection points between healthy and cancerous cell structures across the entire dataset to identify tumor-tissue boundary interfaces and cellular interaction patterns.
Perimeter3D (Medical Analysis)	Perimeter3D (for cancerous cells)	Calculates 3D perimeter measurements for all cancerous cells in the dataset to assess overall tumor growth patterns.
Length3D (Medical Analysis)	Length3D (for cancerous cells)	Computes 3D Length for all cancerous cell regions in the complete dataset to assess total tumor volume and cellular density distribution.
Bridge Analysis (Metaverse)	Metaverse bridge analysis	Finds 5 sampled Metaverse locations with over 5 connection points and longest bridge spans within 100m for virtual bridge planning.
Garden Analysis (Metaverse)	Metaverse city planning	Finds 5 sampled Metaverse locations with highest clear space and space score within 150m for virtual garden planning

used to determine basic spatial or geometric features, such as Length and Convex Hull. Topological relations delineate spatial association between objects given the underlying topological constraints. For 2D spatial data, a standard set of topological relations adopted by the Open Geospatial Consortium (OGC) are supported by most database systems. These topological relations are based on a formal model called the Dimensionally Extended Nine-Intersection Model (DE-9IM) [26] and they include topological relations such as, Equals, Intersects, Touches, Contains and Overlaps. Based on these, the original Jackpine benchmark [2] incorporated a minimal set of topological relations for 2D objects.

However, when it comes to 3D spatial topological relations, DBMSs still offer very limited or no support. Existing formal models, such as DE-9IM, struggle to distinguish detailed interactions involving 3D object(s), for instance, whether objects meet or overlap at vertices, edges, or faces. To address these issues, formal models, such as 25 Intersection model [27] and 36 Intersection model [28] have been proposed in recent years. However, there is no formal model for 3D topological relations that is widely adopted and this is an ongoing research area. Due to these constraints, Jackpine3D includes those 3D topological relations that are commonly supported at this time.

1) Datasets: The primary dataset used in the Micro benchmark is from Open City Model [29] in which the main repository contains data for 125 million buildings across different states and counties in the United States. From that, the data corresponding to Riverside County in California was chosen as the baseline dataset. We uploaded the data for the buildings from there into the Buildings table. This dataset contained 776318 records.

To perform join queries with buildings table we created two other tables as explained next. A 2D dataset was obtained from TIGER [30] (US Census Bureau's geographic spatial data) to build a second table (Roads3d

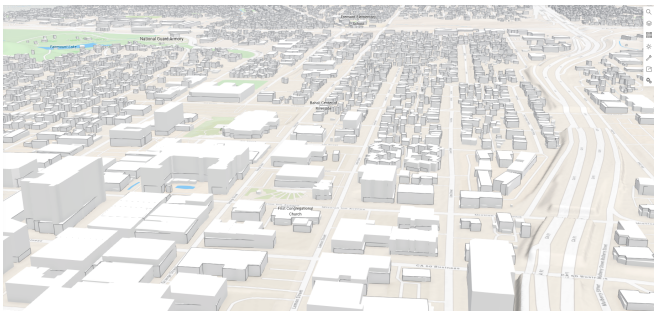

Fig. 4: A 3D illustration of the buildings dataset, which showcases the buildings in Riverside county in California.

table). We combined elevation information from a LiDAR dataset [31] with this 2D dataset to convert it into a 3D dataset. Similarly, another TIGER dataset was used to create AreaLandmarks3d table, by combining the original 2D data with elevation information from a LiDAR dataset [31]. A summary of the Micro benchmark datasets is shown in Table III.

C. Macro Benchmark

This section presents the Macro benchmark workloads. It includes three different real-world workload scenarios, which are explained in detail in the next sections. The queries in the Macro benchmark are presented in Table II.

1) Medical Analysis: In this workload scenario, the aim was to include queries with real-world applications in medical data analysis. The design of these queries was inspired by a study [32]. A tumor dataset from the National Institute of Cancer [33] was obtained, and using that dataset, a synthetic dataset generator produced a set of 10000 3D cancerous and healthy cells.

A set of analysis queries, such as Length (3D) and Perimeter (3D) were implemented to analyze cancerous cells and other join queries, such as intersection, to help understand the relationship between healthy cells and cancerous cells.

2) Urban planning and management: Urban planning and management are important for developers and city planners. In this workload scenario, queries were designed that could help the stakeholders in a city to analyze the infrastructure and the citizens' needs or plan for potential development plans for the future, such as city expansion. For this workload, the dataset used was the open city model dataset [29] for Riverside County.

The queries for this workload scenario involved finding the most suitable areas for a subway station. Parameters, such as, population density and good access to roads and landmarks were used to help someone find the top five optimal areas for a potential subway station. Also similar queries for finding optimal areas for fire stations, and finding good areas for city expansion were included in this workload.

3) Metaverse and Extended Reality (XR): In this workload scenario, queries related to Metaverse and Extended Reality (XR) are explored. Recent advances in 3D game engine software and devices, such as Microsoft Hololens and Meta Quest have opened the door for developing Extended Reality (XR) applications. These applications are expected to usher many novel use cases.

The first query in this workload scenario involved a bridge analysis query in which 5 sampled locations suitable for bridge construction are chosen. The other query involved a garden analysis query to assist with city planning, which finds the five most ideal locations for placing gardens.

IV. Experimental setup

This section describes the details of the experimental setup. The machine we used was an Intel Core i7-7700 CPU (3.60 GHz) with 16 GB of memory and 240 GB disk, running Ubuntu 20.04 LTS. The database systems we evaluated in our benchmark were PostgreSQL, DB-X, and SpatialLite. We installed these databases in their default configuration without any performance optimization. All the tables in these databases have spatial columns. The configuration parameters of each database is shown in Table IV.

The benchmark utilizes a Java Database Connectivity (JDBC) connection pool to connect to the databases. To avoid network delay being included in the query execution times, we ran the benchmark on the same machine as the databases. During each benchmark scenario execution, a warm-up run was first performed, followed by three successive iterations and taking the average of the three runs.

A. Data Loading

To import data from files corresponding to the datasets into database tables, loading tools and other methods were used. For PostgreSQL, GDAL's ogr2ogr tool was used. For DB-X, a shapefile parser [34] was used to parse the datasets and then we created a JDBC data loader to load the parsed data into the database tables. To compare these two methods, ogr2ogr was significantly faster (roughly $1/20th$ the execution time) than the JDBC approach. For DB-X, the process of uploading data took more effort because, besides parsing the datasets and converting them into CSVs, some preprocessing to CSVs had to be performed for the data to be compatible with DB-X SDO.GEOMETRY, which enabled us to perform spatial operations on the data. For SpatialLite, similar to PostgreSQL, a native tool exist, which is called spatial-tool. Also, similar to PostgreSQL for SpatialLite, the process of data uploading was straightforward and did not require any preprocessing. For DB-X a JDBC data loader was then created to read the CSVs and create a connection to the database. Afterwards, a table was created in the database and then the table was populated by inserting data records from the CSV file. For the buildings table, which was the largest table, the process of uploading data took about 70 minutes using this JDBC data loader. The process was much faster for other tables, which were relatively smaller. For PostgreSQL and SpatialLite, the process was much faster and took approximately three minutes for the buildings table. For other tables, which were smaller than the buildings, it took several seconds to upload the data.

TABLE III: Database tables used for micro and macro benchmarks

Database Table	Geometry	Cardinality
Buildings	Polyhedral Surface	776318
AreaLandmarks (TIGER)	MultipolygonZ	8090
Roads (TIGER)	MultipolygonZ	58448
SyntheticCells	MultipolygonZ	10000

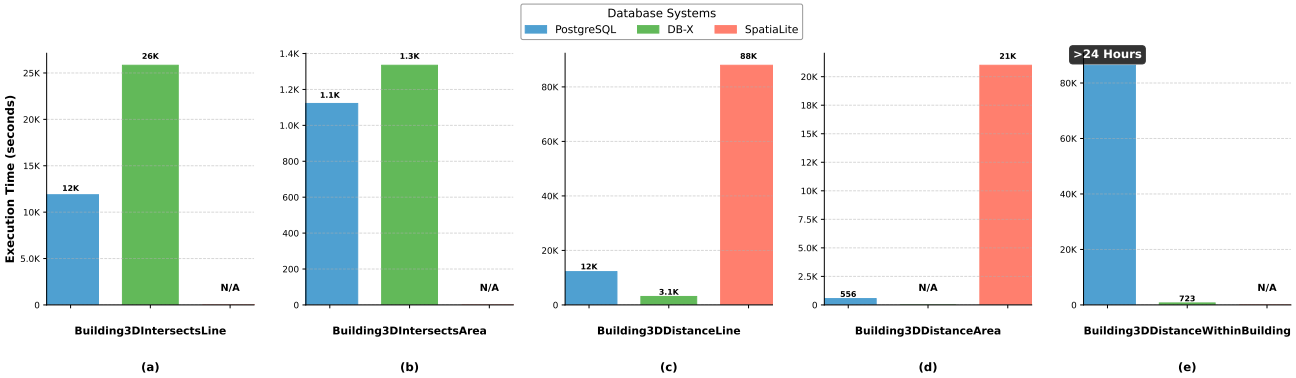

Fig. 5: Pairwise spatial joins between buildings, roads and arealandmarks data tables (N/A: Not Available).

TABLE IV: Database configuration parameters

Database	Version	Data (GB)	Bufferpool (MB)
SpatialLite	3.45.1	4.2	2
PostgreSQL	16.9	4.2	8192
DB-X	21c	4.2	368

V. Benchmarking Results

In this section, the results of the benchmarking experiments are presented. The results show comparison between PostgreSQL, DB-X, and SpatialLite in terms of query execution time. The elapsed time of each query from the workloads was measured and then reported. There are some differences between the mentioned databases. One of them is the index that they use. DB-X uses R-tree spatial indexing, while PostgreSQL uses GiST indexes for spatial operations. SpatialLite, built on SQLite with spatial extensions, uses R*Tree indexes but on the other hand, when compared to PostgreSQL and DB-X, it is not typically considered a full-fledged database. Also, available 3D operations and features in SpatialLite were minimal compared to both PostgreSQL and DB-X. However, because of its unique advantage of being lightweight and very quick deployment time, it may be useful in some spatial applications.

A. Micro Benchmark

This section presents the results of Micro benchmark queries. They are grouped into two categories: join queries and analysis queries.

1) Join Queries: The first category of the results includes five pairwise spatial join queries among polyhedral surface and MultipolygonZ objects. MultiPolygonZ is a 3D geometry data type representing a collection of multiple polygons, each with X, Y, and Z (elevation) coordinates to enable modeling of surfaces with elevation in spatial databases. One of the topological predicates is "3DIntersects", which identifies if two geometries intersect in 3D space. This function was executed on two scenarios between buildings and arealandmarks and buildings and roads. The results are plotted in Figure 5. In the first two experiments involving Building3DIntersectsLine and Building3DIntersectsArea (refer to Figure 5(a) and Figure 5(b), PostgreSQL performed better than DB-X. SpatialLite did not have the functions needed for these experiments.

The next topological function in this workload was 3DDistanceWithin, which returns two objects that are within a given distance. The respective results are shown in Figure 5. The queries involved were Building3DDistanceLine and Building3DDistanceArea and also Building3DDistanceWithinBuilding. SpatialLite was also included in these two experiments. In these experiments, DB-X had a much lower (25%) execution time than PostgreSQL. In all cases the highest execution time was for SpatialLite by far (10-20× more execution time than PostgreSQL). This is not surprising because SpatialLite's 3D spatial capabilities are quite limited.

2) Analysis Queries: Next, we investigated a few spatial analysis functions. Figure 6 shows the elapsed times of these queries. The functions used in these queries were for spatial analysis, and unlike the previous join queries, they involve only one table. Some of the functions include Perimeter3D that measures the 3-dimensional perimeter of the geometry, Length3D that returns the 3-dimensional length of a geometry, and a few other analysis functions. For the Perimeter3D, PostgreSQL had a 15% lower execution time than DB-X, which was also observed in the operation Length3D. The difference in execution times in these analysis functions was notably more significant than the difference in execution times in the previous join queries (3D intersection and distance within), which is important to highlight. Another observation was that for operation Length3D, SpatialLite had a much lower execution time than DB-X and PostgreSQL. In the following query, Dimensions, which returns the dimensions of the

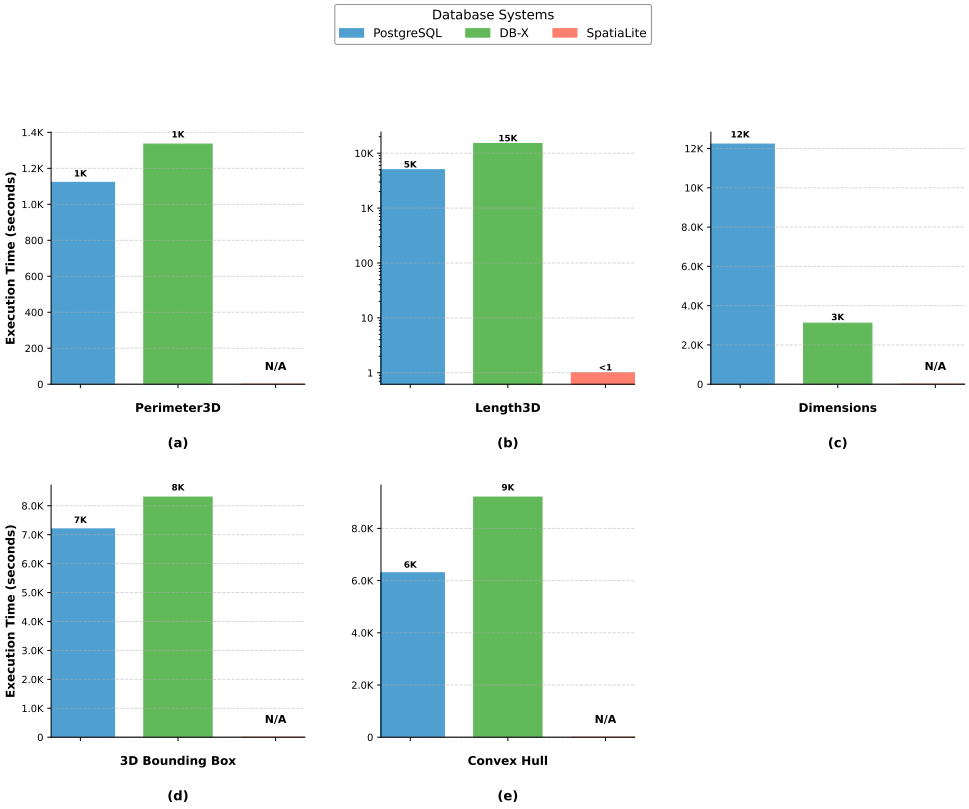

Fig. 6: Spatial analysis operations (N/A: Not Available).

geometry (which, in our case, is 3D), it was interesting that PostgreSQL had a much higher execution time when compared to DB-X, since in the rest of the analysis queries, PostgreSQL had a lower execution time than DB-X. The following query for evaluation was the 3D Bounding Box, which returned the bounding boxes that fully bounded the geometries of the data objects. In this query, similar to the queries of Perimeter3D and Length3D, the difference in execution time was quite notable, and PostgreSQL had a 14% lower execution time than DB-X. PostgreSQL also performed better than DB-X on the Convex Hull query that returns the 3D convex hull of a geometry.

Surprisingly, queries in this workload scenarios showed significant differences in their performance across databases, which demonstrates the merit of benchmarking.

B. Macro Benchmark

In this section, the analysis of the results from running Macro benchmark queries is discussed, along with the insights gained from them.

1) Urban Planning and Management: The urban planning and management was the first workload scenario to evaluate. Initially, our plan was to run the benchmark queries on all the records in the corresponding tables. However, when the queries were run, it became clear that the execution time would be quite high, thus necessitating an adjustment to our approach. So instead, an approach was devised to consider a grid space on the dataset based on longitude and latitude and reduce the cardinality of the query resultset by performing a deterministic random sampling of records from the associated tables. By doing that, the execution time was reduced quite considerably. The results are shown in Figures 7(a), 7(b) and 7(c). In this workload scenario, DB-X outperformed PostgreSQL in terms of execution time. The difference in execution times between the two was quite notable across the three different queries. One possible reasons that likely contributed to this was the use of the 3DDistanceWithin function in those queries. We observed earlier that this function in PostgreSQL had an unusually high execution time compared to similar functions, and this might be the reason that, despite conducting deterministic sampling, the execution time of those queries in PostgreSQL was much higher than in DB-X. The deterministic random sampling approach was adopted for the rest of the Macro benchmark workload scenarios.

2) Medical Analysis: For the medical analysis workload scenario, a set of join and analysis queries was implemented. Multiple experiments were made to determine a suitable number of cells from the synthesized cell generator for our queries. The results of the experiments are shown in Figures 7(d), 7(e) and 7(f). The queries that were tested in this scenario consisted of a join query (3DIntersection of healthy and cancerous cells) and a couple of analysis, namely, Perimeter3D and Length3D. For the join query involving the intersection operation, the execution time of PostgreSQL was lower than DB-X (approximately 20% of the execution time of DB-X). For the Perimeter3D query, the difference in execution time between PostgreSQL and DB-X was quite considerable (higher than the previous intersection query), and also for the Length3D query execution time in PostgreSQL was lower.

3) Metaverse and Extended Reality (XR): In this workload scenario, the queries were designed for visibility analysis, height analysis, and navigation in Metaverse. Like the previous scenarios (urban planning and management),

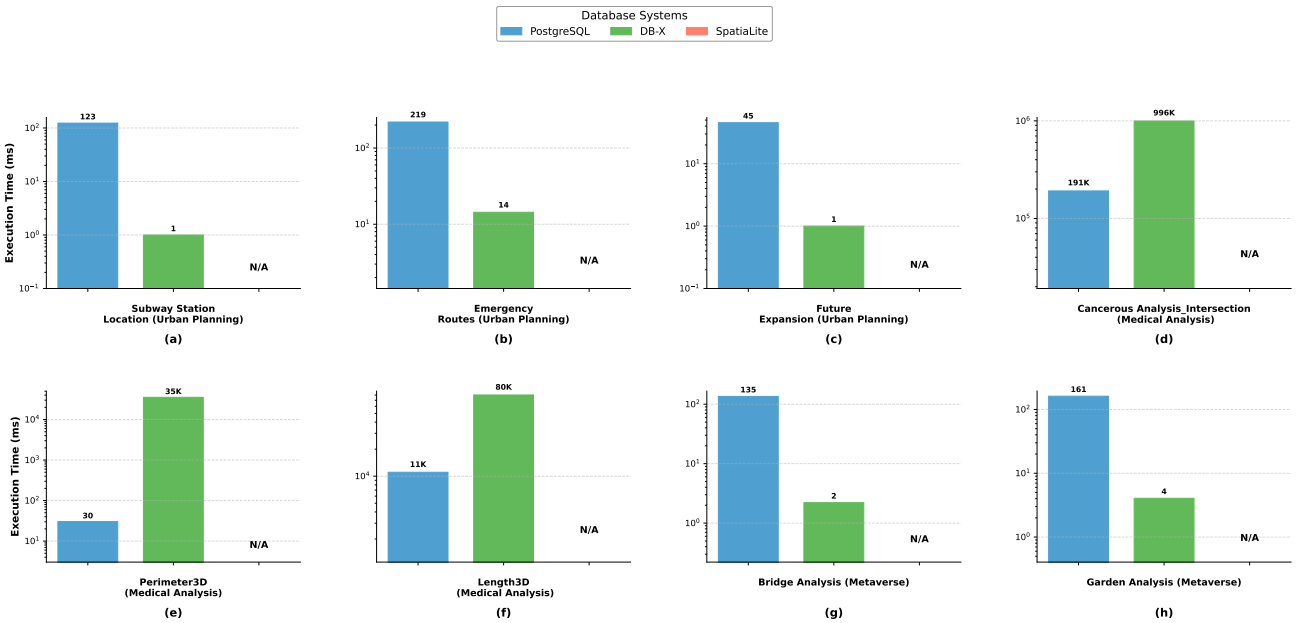

Fig. 7: Macro benchmark scenarios (N/A: Not Available).

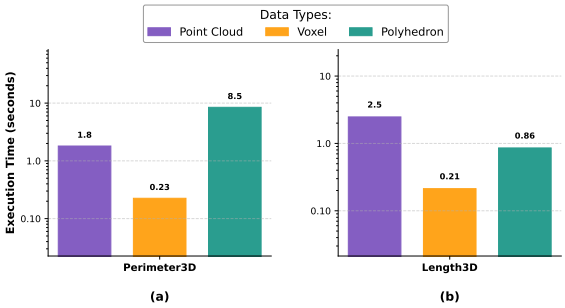

Fig. 8: Performance comparison with different 3D spatial representations: point cloud, voxel and polyhedron.

since 3D operations are computationally intensive for these queries, a deterministic random sampling approach was used to reduce the resultset cardinality of the queries. The results from these queries, as shown in Figures 7(g) and 7(h), also exhibited that queries on DB-X had a lower execution time than on PostgreSQL. The difference in execution times between DB-X and PostgreSQL was relatively high and unusually higher than previous queries and experiments. As mentioned earlier, a likely reason for this was the use of the 3DDistanceWithin function, which had a high execution in PostgreSQL leading to much higher execution time in PostgreSQL than in DB-X.

C. 3D Spatial Data Types

In addition to the Micro benchmark and Macro benchmark workloads, we wanted to evaluate the performance of databases due to different 3D spatial data types. To that end, experiments were conducted on different 3D spatial representations, polyhedral surface (polyhedron), point cloud and voxel, introduced earlier. The extent of these experiments was not as broad as in previous sections. However, expanding the scope of these experiments and doing more analysis on these different representations of 3D could be a good future direction for researchers and a good direction for expanding this benchmark.

The Buildings dataset utilized in the Buildings3d table uses polyhedral surface (polyhedron) to represent 3D spatial objects. For these experiments, we converted the polyhedral surface into point cloud and voxel. For conversion to voxel, the work of Li et al. [35] was found to be very useful, in which the authors explained voxel data management in PostgreSQL. After the conversion, experiments were conducted on these data types by running a few queries involving spatial operations. Some operations on these data types included Length3D and Perimeter3D. As shown in Figure 8, for both of these operations, the execution time for voxel was lower than that of point cloud. Both data types are efficient in representation and processing. The queries with polyhedron (polyhedral surface) took longer than the others, because this data type is used to represent 3D objects precisely. On the other hand, this demonstrates the need for further research to improve query execution performance with polyhedron data.

D. Memory Performance Analysis (TLB and LLC Misses)

Understanding why some queries perform sub-optimally necessitates going beyond their execution times and investigating low-level system behavior. One critical aspect influencing performance is how the queries interact with the system memory hierarchy. To explore this, we conducted a detailed memory performance analysis using the Linux `perf` tool. This analysis focused on TLB (Translation Lookaside Buffer) misses and LLC (Last Level Cache) misses to uncover potential memory bottlenecks. These experiments were conducted only with PostgreSQL, as it is a mature open-source database with good available documentation. We plotted the results in Figure 9. Next, we discuss these results.

1) Query Performance Variation: An interesting but not unexpected trend from the TLB and LLC miss results is that there was a high degree of variation among the four queries on PostgreSQL:

- Query1 (Dimensions): 22,663 TLB misses, 131,318 LLC misses (moderate performance)

- Query2 (Convex Hull): 78,789 TLB misses (worst performance), 176,976 LLC misses
- Query3 (Bounding Box): 39,393 TLB misses, 97,490 LLC misses (best performance)
- Query4 (Building3DIntersectsArea): 76,146 TLB misses, 374,527 LLC misses (worst performance)

2) Memory Access Efficiency: All four queries generated substantial LLC cache misses, which indicate that PostgreSQL's spatial query execution struggles with memory locality. The number of TLB misses shows frequent page table lookups because of scattered memory access patterns , while the high number of LLC misses indicates the working sets exceed last-level cache capacity.

3) Spatial Index Utilization: The varying miss rates across all these queries suggest that PostgreSQL's spatial indexing effectiveness is query-dependent. Some of the operations seem to benefit more from R-tree or GiST indexing than others, which leads to inconsistent memory access across different operations.

4) Overall Assessment: PostgreSQL demonstrates poor cache efficiency for some spatial workloads with high variability. The system appears to load and process geometric data in ways that are not optimized for cache hierarchy, and this could be further improved.

E. Execution Time Breakdown

In this section, an analysis regarding the query execution time breakdown of different PostgreSQL operations is provided. These statistics are plotted in Figure 10. Next we discuss these results.

1) Query-Specific Bottlenecks: To better understand the internal behavior of different spatial queries, we conducted a second set of performance profiling analyses using the Linux `perf` tool. This allowed us to identify which operations dominated execution time and where specific overheads originated. Each of the spatial queries shows distinct performance characteristics. ConvexHull is geometry-computation heavy (48.65%), Dimension is validation-dominated (82.93%), Bounding Box suffers from high decompression costs (29.26%), and 3DDistance incurs a large proportion of overhead in non-primary operations (58.42%), suggesting significant time spent on ancillary computation and data handling. Decompression overhead refers to unpacking compressed geometric data before processing them, while validation includes verifying geometric integrity and spatial reference conformance.

The Bounding Box query operations show the highest decompression cost at 29.26% of execution time, indicating compressed geometric data creates significant performance bottlenecks for certain spatial operations compared to other operations. Bounding Box and 3DDistanceWithin queries show substantial I/O overhead (34.63% and 42.3% respectively) indicating that these operations need more data movement or result materialization than geometry data processing. The varying performances seen in these operations show operation-specific inefficiencies in PostgreSQL's spatial processing and show opportunities to reduce decompression overhead for bounding operations and minimize ancillary costs for distance calculations.

VI. Future Research Directions

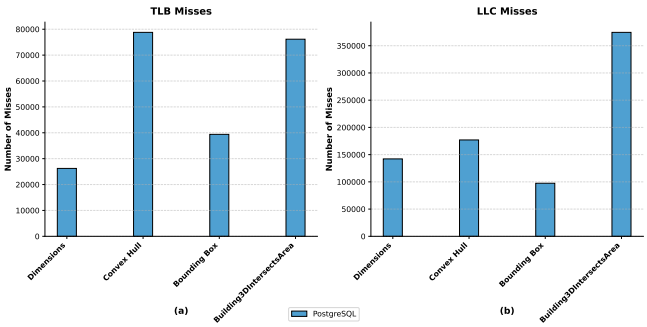

Fig. 9: Memory performance of spatial queries on PostgreSQL. Each bar shows the relative contribution of different operations.

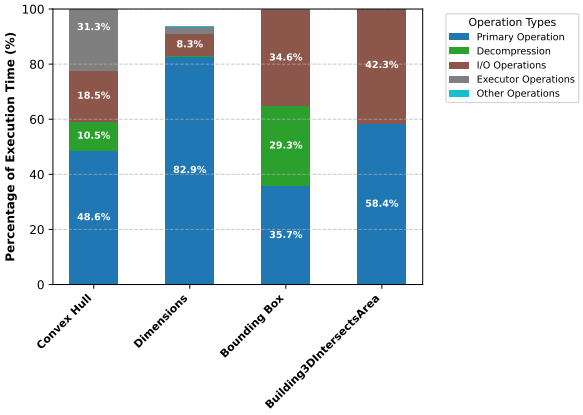

Fig. 10: Performance breakdown comparing four PostgreSQL spatial queries. Each bar shows the relative contribution of different operations.

Due to rising popularity of spatial applications, such as Location-Based Services (LBS), mapping and navigation, urban planning, environmental management, and emergency response, most relational database systems support 2D spatial features. However, these database systems offer limited or no 3D spatial support. Hence, there are significant opportunities for research, and some of them have been identified through our benchmarking effort.

- *Formal model for 3D topological relations.* While for 2D spatial data, DE-9Im [26] was adopted by database systems to implement spatial topological relations, there is yet no widely adopted formal model for 3D spatial topological relations Although models, such as 25 Intersection model [27] and 36 Intersection model [28] have been proposed, there is still significant rooms for improvement.
- *Support for 3D spatial features.* As revealed in our benchmarking study, most of the database systems do not support 3D spatial data types, such as polyhedron, spatial analysis functions or spatial topological relations. The databases that do support 3D spatial features, their current offering is quite limited. For instance, PostgreSQL currently only supports ST_3DIntersects topological relation and that too for a few 3D geometry types only.
- *Efficient execution of 3D spatial queries.* Spatial queries are resource intensive to process, and a two step Filter-Refinement approach is utilized by

database systems. Due to their complexity, 3D spatial queries involving topological relations are even more expensive. As we found in our experimental evaluation, for some of the queries it was too time consuming to process all data, leading us to resort to a deterministic sampling. Recently researchers have proposed a Filter-Progressive-Refine approach, which performs progressive compression of 3D objects [17]. However, this approach has some limitations and there is a significant opportunity in this area.

- *Efficient indexing and access methods for 3D spatial queries.* To improve the performance of the Filter step of the two step Filter-Refinement, database systems use spatial index structures like R-tree [19], Quadtree [7] or their variants. However, they are not optimized for indexing 3D objects. Although, a few techniques, such as Orientable Bounding Box trees (OBB-trees) [20], have been proposed, they are yet to be adopted for a Filter-Progressive-Refine paradigm. Moreover, how to efficiently store and retrieve multiple levels of details of objects of different 3D spatial data types, such as polyhedron and voxel, is an open research topic.

VII. Conclusion

Due to recent technological advancements, the volume of 3D spatial data has been experiencing rapid growth. Because of this and their wide application and availability, the support for 3D spatial data and functionality across different database systems, is vital. In this paper, we proposed Jackpine3D benchmark, the first 3D spatial database benchmark. Jackpine3D comprises Micro benchmark workloads containing basic spatial topological relations and data analysis functions, and Macro benchmark workloads that include several real-world application scenarios.

We evaluated three relational database systems with Jackpine3D. Based on our observations, we observed that execution times of 3D queries are generally long. We have outlined a number of future research directions for the research community.

VIII. Acknowledgment

We acknowledge the support of the NRC (National Research Council) DHGA program for this research.

References

[1] H. Shin, K. Lee, and H.-Y. Kwon, "A comparative experimental study of distributed storage engines for big spatial data processing using geospark," *The Journal of supercomputing*, vol. 78, no. 2, pp. 2556–2579, 2022.

[2] S. Ray, B. Simion, and A. D. Brown, "Jackpine: A benchmark to evaluate spatial database performance," in *ICDE*, 2011, pp. 1139–1150.

[3] A. Belussi, S. Migliorini, and M. Negri, "A framework for evaluating 3d topological relations based on a vector data model," *Geoinformatica*, vol. 24, no. 4, pp. 915–950, 2020.

[4] S. Salleh, U. Ujang, and S. Azri, "Topology models and rules: a 3d spatial database approach," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 48, pp. 117–122, 2023.

[5] "SQLite," https://en.wikipedia.org/wiki/SQLite.

[6] P. Ramsey, "Lidar in postgresql with pointcloud," 2013.

[7] R. K. V. Kothuri, S. Ravada, and D. Abugov, "Quadtree and R-tree indexes in Oracle Spatial: a comparison using GIS data," in *ACM SIGMOD*, 2002, pp. 546–557.

[8] T. P. P. Council, "Transaction processing performance council," *Web Site, http://www. tpc. org*, 2005.

[9] "DBT - Database Test Suite," http://osdldbt.sourceforge.

[10] B. Simion, S. Ray, and A. D. Brown, "Surveying the landscape: an in-depth analysis of spatial database workloads," in *ACM SIGSPATIAL*, 2012, pp. 376–385.

[11] M. Stonebraker, J. Frew, K. Gardels, and J. Meredith, "The sequoia 2000 storage benchmark," *ACM SIGMOD Record*, vol. 22, no. 2, pp. 2–11, 1993.

[12] N. W. Paton, M. H. Williams, K. Dietrich, O. Liew, A. Dinn, and A. Patrick, "Vespa: A benchmark for vector spatial databases," in *BNCOD*, 2000, pp. 81–101.

[13] P. van Oosterom, O. Martinez-Rubi, T. Tijssen, and R. Gonçalves, "Realistic benchmarks for point cloud data management systems," *Advances in 3D Geoinformation*, pp. 1–30, 2017.

[14] E. De Kemp, T. Monecke, M. Sheshpari, E. Girard, K. Lauzière, E. Grunsky, E. Schetselaar, J. Goutier, G. Perron, and G. Bellefleur, "3d gis as a support for mineral discovery," 2011.

[15] Esri, "3d gis — 3d mapping software - arcgis - esri," 2023, accessed: 2025-03-02. [Online]. Available: https://www.esri.com/en-us/capabilities/3d-gis/overview

[16] B. C. Ooi, G. Chen, M. Z. Shou, K.-L. Tan, A. Tung, X. Xiao, J. W. L. Yip, B. Zhang, and M. Zhang, "The metaverse data deluge: What can we do about it?" in *ICDE*, 2023, pp. 3675–3687.

[17] D. Teng, Y. Liang, F. Baig, J. Kong, V. Hoang, and F. Wang, "3DPro: querying complex three-dimensional data with progressive compression and refinement," in *EDBT*, vol. 25, no. 2, 2022, p. 104.

[18] H.-H. Park, K.-Y. Whang, and I.-Y. Song, "Early separation of filter and refinement steps in spatial query optimization," in *ICDE*, 1998.

[19] P. K. Agarwal, M. de Berg, J. Gudmundsson, M. Hammar, and H. J. Haverkort, "Box-trees and r-trees with near-optimal query time," in *Proceedings of the seventeenth annual symposium on Computational geometry*, 2001, pp. 124–133.

[20] S. Gottschalk, M. C. Lin, and D. Manocha, "Obbtree: A hierarchical structure for rapid interference detection," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 171–180.

[21] B. Lonneville, C. Stal, B. De Roo, A. De Wulf, and P. De Maeyer, "Determining geometric primitives for a 3d gis easy as 1d, 2d, 3d?" in *GISTAM*. IEEE, 2015, pp. 1–6.

[22] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy, *Polygon Mesh Processing*. CRC Press, 2010.

[23] W. Li, S. Zlatanova, and B. Gorte, "Voxel data management and analysis in postgresql/postgis under different data layouts," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 6, pp. 35–42, 2020.

[24] C. D. Hansen and C. R. Johnson, *The Visualization Handbook*. Elsevier, 2005.

[25] M. Gao, N. Ruan, J. Shi, and W. Zhou, "Deep neural network for 3d shape classification based on mesh feature," *Sensors*, vol. 22, no. 18, p. 7040, 2022.

[26] E. Clementini and P. Di Felice, "A model for representing topological relationships between complex geometric features in spatial databases," *Information sciences*, vol. 90, no. 1-4, pp. 121–136, 1996.

[27] M. Zhou and Q. Guan, "A 25-intersection model for representing topological relations between simple spatial objects in 3-d space," *ISPRS International Journal of Geo-Information*, vol. 8, no. 4, p. 182, 2019.

[28] S. Salleh, U. Ujang, and S. Azri, "Representing 3d topological adjacencies between volumes using a 36-intersection model," *Geomatics and Environmental Engineering*, vol. 16, no. 2, pp. 127–155, 2022.

[29] O. Contributors, "Opencitymodel," n.d., accessed: 2025-03-10. [Online]. Available: https://github.com/opencitymodel/opencitymodel

[30] U.S. Census Bureau, "Geographic shapefiles," 2025. [Online]. Available: https://www.census.gov/cgi-bin/geo/shapefiles/

[31] California Natural Resources Agency, "California state geoportal - lidar data," https://gis.data.ca.gov/search?layout=grid&tags=lidar, 2025, accessed: 2025-09-08.

[32] A. Aji, F. Wang, and J. H. Saltz, "Towards building a high performance spatial query system for large scale medical imaging data," in *ACM SIGSPATIAL*, 2012, pp. 309–318.

[33] National Cancer Institute, "Genomic Data Commons Portal," 2025, accessed: 2025-03-10. [Online]. Available: https://portal.gdc.cancer.gov/

[34] D. IM, "Java shapefile parser," n.d., accessed: 2025-03-10. [Online]. Available: https://github.com/delight-im/Java-Shapefile-Parser/tree/master

[35] W. Li, S. Zlatanova, and B. Gorte, "Voxel data management and analysis in postgresql/postgis under different data layouts," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 6, pp. 35–42, 2020.