This is the Accepted Manuscript version of an article published by Taylor & Francis in the *International Journal of Geographical Information Science* in 2025, which is available at: https://doi.org/10.1080/13658816.2024.2434606
Cite as:
Yu D, Yue P, Wu B, Biljecki F, Chen M, Lu L (2025): Towards an Integrated Approach for Managing and Streaming 3D Spatial Data at the Component Level in Spatial Data Infrastructures. *International Journal of Geographical Information Science*.

Towards an Integrated Approach for Managing and Streaming 3D Spatial Data at the Component Level in Spatial Data Infrastructures

Dayu Yu[a,b], Peng Yue[a,c,d,e]*, Binwen Wu[a], Filip Biljecki[f,g], Min Chen[b], and Luancheng Lu[a]

[a]*School of Remote Sensing and Information Engineering, Wuhan University, Wuhan, China;* [b]*Key Laboratory of Virtual Geographic Environment (Ministry of Education of PRC), Nanjing Normal University, Nanjing, China;* [c]*Collaborative Innovation Center of Geospatial Technology, Wuhan, China;* [d]*Hubei LuoJia Laboratory, Wuhan, China;* [e]*Hubei Province Engineering Center for Intelligent Geoprocessing (HPECIG), Wuhan University, Wuhan, China;* [f]*Department of Architecture, National University of Singapore, Singapore, Singapore;* [g]*Department of Real Estate, National University of Singapore, Singapore, Singapore*

Correspondence: Peng Yue (pyue@whu.edu.cn)

Toward an Integrated Approach for Managing and Streaming 3D Spatial Data at the Component Level in Spatial Data Infrastructures

Transitions of spatial data infrastructures (SDIs) support applications from 2D landscapes to 3D scenes. The existing methods for describing, managing, and providing services for 3D spatial data often lack coordination and efficiency. Moreover, the added complexity of 3D data structures necessitates novel approaches for component-level management and streaming capabilities. In response, we developed a generic conceptual model suitable for component-level management of diverse 3D spatial data in SDIs and discussed the design rationales and key considerations underlying the model. We formalized the flexible data composition and fine-grained lifecycle management in this model and specified this model at the cloud-optimized encoding level to enable efficient CRUD operations and streaming delivery of massive 3D spatial data. Our approach enabled direct streaming of the managed 3D spatial data without the need for redundant replication. We implemented, evaluated, and discussed the proposed approach in terms of service, accessibility, visualization, analysis cases, and efficiency. The results show that the proposed method is efficient in managing 3D spatial data and enables users to conduct 3D geo-analysis on the basis of specific parts of the data as needed. This work provides a scientific exploration that integrates the management and services of 3D spatial data in SDIs.

Keywords: spatial data infrastructure, information modeling, 3D data management, geographical information system

1. Introduction

Spatial data infrastructures (SDIs) are service-oriented mechanisms designed to efficiently collect, store, manage, and share spatial data (Budhathoki et al., 2008; Hendriks et al., 2012). Many SDIs at the regional, national, or global level have been proposed and implemented for Earth observations (Gao et al., 2022; Gorelick et al., 2017; Izdebski et al., 2021; Lewis et al., 2017). These SDIs have facilitated the discovery and reuse of 2D spatial data, such as raster and vector data, thereby showing

significance in areas such as environmental monitoring, agricultural analysis, and climate change (Owers et al., 2022; Yue et al., 2015, 2016).

As information acquisition technologies advance, detailed and semantically enriched 3D spatial data are being collected worldwide. 3D spatial data convey aesthetic and spatial relationships more effectively than 2D data do (Labetski et al., 2023). Many studies have shown that 3D spatial data constitute an advanced tool for urban analysis, disaster evolution simulation, and microclimate analysis (Biljecki et al., 2015, 2018, 2021). Consequently, high-value 3D data propel the transformation of geographic information systems (GISs) and SDIs from a 2D paradigm to a 3D paradigm at an astounding scale and pace. However, the increase in dimensions results in an amplified geometric volume. Furthermore, as shown in Fig. 1, the diversity and intricate structure introduce greater complexity than initially anticipated.

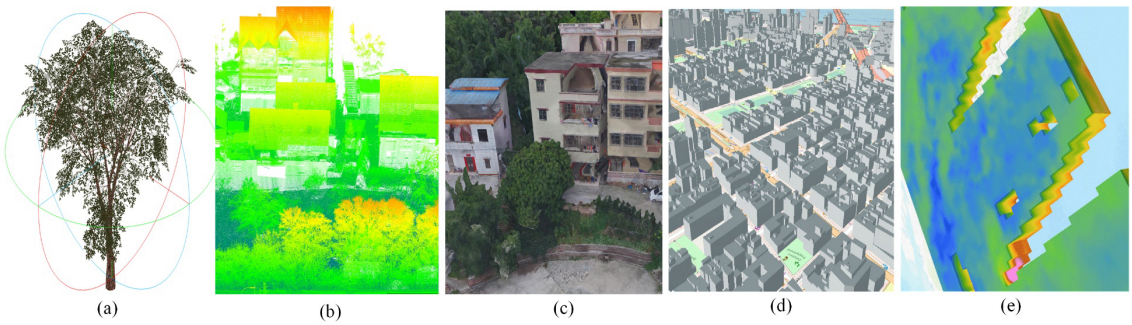


Fig. 1. Different forms of 3D spatial data. Examples of (a) 3D mesh, (b) point cloud, (c) photorealistic 3D mesh, (d) CityGML, and (e) 3D voxel model.

For the construction of an SDI for 3D data, Stoter et al. (2020) identified several primary challenges: data management, data heterogeneity, standardization, interoperability, and data quality. Rapid progress in machine learning has spurred much research on the semantic processing of 3D data (Lai et al., 2022; Yu et al., 2021, 2023; Yuan et al., 2022). To facilitate interoperability for 3D, the Open Geospatial Consortium (OGC) has released multiple specifications, including 3D Tiles[1], Indexed 3D Scene Layers (I3S[2]), City Geography Markup Language (CityGML[3]), and CDB[4].

However, the challenges in 3D data management remain enormous due to the diverse and intricate nature of 3D spatial data. Many 3D spatial data are constructed from numerous components. For example, a complex building is assembled from elements such as tables, walls, and doors, whereas a large geographic scene is composed of nodes representing different areas. Moreover, complex geoanalytics tasks often involve multiple components from different datasets to perform relevant simulations. This inherently requires the SDI to manage and deliver 3D data efficiently at the component level. Recent efforts have focused on the management of specific data formats (Yao et al., 2018) but disregard other data formats. Furthermore, the preview and visualization of objects close to viewpoints necessitate SDIs to deliver voluminous data during streaming, a requirement often overlooked by existing 3D data management studies. Consequently, redundant duplicates are typically required to facilitate streaming delivery, further complicating the management process.

In response to these challenges, this paper introduces an integrated approach for the management and streaming delivery of 3D spatial data at the component level. First, we model a conceptual structure for managing the multigrained representation and fine-grained lifecycle of 3D spatial data. On the basis of this structure, second, we extend the create, retrieval, update, and delete (CRUD) operations for 3D spatial data. Last, this approach encodes the structure into a database schema suitable for efficient CRUD operations. Our goal is to explore efficient management and service methods for 3D spatial data in SDIs. The experiments demonstrate that the proposition is remarkably effective in managing 3D spatial data, streaming it on the web, and enabling users to inspect specific parts of the data as needed. In summary, the contributions of this paper are as follows:

- A generic conceptual model suitable for component-level management of diverse 3D spatial data in SDIs is developed.
- The flexible data composition and fine-grained lifecycle management in this model are formalized.
- This model is encoded with a cloud-optimized architecture to support the extended CRUD and streaming delivery of massive 3D data, avoiding unnecessary replication and conversion.
- The proposed approach is implemented, evaluated and discussed.

2. Background and related works

2.1. Interoperability and standards for delivering 3D data

The International Organization for Standardization (ISO) GL transmission format (glTF[5]) minimizes the 3D content used for rendering and the runtime processing required to parse it. However, glTF only stores the geometric structure, disregarding the semantic and topological aspects of a 3D city. To address this problem, OGC CityGML[4] defines a conceptual model for describing and delivering semantic 3D city models (Gröger & Plümer, 2012). CityGML defines the geometry, semantics, topology, and application domain extension (ADE) of the most important urban objects in a modular way.

The conceptual model of CityGML is continuously evolving. To address the limitation that interior building structures can be represented only by the level of detail (LOD) 4, Boeters et al. (2015) extended LOD2 to encompass simplified indoor geometries. Geometrically, the 5 LODs in CityGML are often too generalized. Hence, a set of 16 LODs is proposed to provide geometric supplements (Biljecki, Ledoux, & Stoter, 2016). In response to the redundancy of XML[6] encodings, CityJSON[7] was

introduced to reduce the complexity of application development for CityGML(Ledoux et al., 2019). Recently, CityGML 3.0 was approved as an OGC standard. This updated version includes indoor representations at all LODs and introduces new elements such as versioning, dynamic attributes, and point clouds (Kutzner et al., 2020).

The OGC common database (CDB) is designed as an interoperable container for geospatial data among 3D geo-simulation applications. CDB serves as a runtime database for geospatial information essential to 3D geo-simulation scenes, including rasters, vectors, and 3D models (Saeedi et al., 2017). While OGC CDB is frequently employed within the United States government for the modeling of 3D synthetic environments, its utilization in academic research remains relatively limited.

Web-side 3D rendering faces pressure from the increasing scale and complexity of 3D geospatial scenes. Streaming delivery is a compelling solution for alleviating this pressure, with OGC having established 3D Tiles and I3S standards for web stream optimization of 3D spatial data (Lu et al., 2021). These standards employ a fragmentation strategy for delivering 3D models, thereby alleviating the local loading pressure experienced on the web. The Logic4DCity model subsequently integrates the temporal dimension into 3D tiles, enabling interactive visualization of time series 3D city models on the web (Jaillot et al., 2020). To facilitate interoperability among these solutions, 3D GeoVolumes[8] is proposed. 3D GeoVolumes defines a unified API for requesting, receiving, and transmitting 3D content across different data providers, thereby simplifying the development of 3D applications.

In building information modeling (BIM), the industry foundation class (IFC)[9] was developed to store and exchange construction project information, including geometry, materials, schedules, quantities, and spatial relationships among building elements. To integrate GIS with BIM data, OGC LandInfra[10] offers a conceptual model

for representing land and civil engineering infrastructure facilities. OGC IndoorGML[11] integrates indoor building objects for navigation purposes. In addition, Liu et al. (2016) proposed a framework for converting IFC and CityGML data to integrate BIM and GIS. Wang and Xie (2022) proposed a function integration method for 3D GIS and BIM, which is based on IFC and CityGML, with applications in the visual detection of concealed facilities.

In summary, various solutions for delivering 3D spatial data exist, but harmonizing and converting between them remains challenging owing to differences in features and suitability for specific scenarios. Table 1 highlights key formats: CityGML and IFC are commonly used for modeling and exchanging semantic data but lack streaming capabilities and are not web friendly (Schilling et al., 2016; Jaillot et al., 2021). The Open Scene Graph (OSG) format is primarily used for delivering photorealistic 3D meshes on desktops but not on the web (Jiang et al., 2017). Like glTF, 3D tiles, and I3S can stream 3D data online but do not support temporal dimensions or versioning (Jaillot et al., 2020). These variations present significant challenges for the integrated management and service of 3D data.

Table 1. Feature comparison of popular 3D specifications or formats

Specifications	*Asset type*	*Semantic extension*	*LOD*	*Asset referencing*	*Versioning*	*Encoding*	*Net stream optimization*
glTF	Model	No	No	No	No	JSON/binary	Yes
3D Tiles 1.1	Scene	application extras	HLOD	External reference	No	JSON+glTF	Yes
I3S 1.1	Scene	No	Mesh pyramid	Node reference	No	SLPK	Yes
OSG	Scene	No	PagedLOD	shallow_copy	No	ASCII/binary	No
CityGML 3.0	Scene	ADE	LOD0-3	Implicit geometry	Yes	GML	No
Ours	Multi scene	ADE	LOD	Node reference	Yes	Database schema	Yes

2.2. SDIs and management of 3D spatial data

Building SDIs for 3D data has been an important topic in GIS research for years. Several studies have attempted to construct SDIs for 3D data. For example, Basanow et al. (2008) discussed the implementation of 3D SDI based on open standards for

Heidelberg. The 3D testbed in the Netherlands resulted in the proof of concept of the 3D SDI (Stoter et al., 2011). Alizadehashrafi (2019) introduced a framework for the 3D SDI of Iran on the basis of OGC standards. Table 2 compares the core components of different SDIs. These SDIs typically use relational database management systems (RDBMSs) to store 3D data, which are served through web 3D service (W3DS).

Table 2. Comparison of different 3D SDIs on core components

SDIs	*Data simplification*	*Data repository*	*Data service*
Heidelberg 3D SDI	3D extension on styled layer descriptor	PostgreSQL	WMS/W3DS
Netherlands 3D SDI	3D Standard NL, compatible with CityGML	RDBMS	Web Services
Iran 3D SDI	CityGML	RDBMS	WMS/W3DS

3D data management is at the heart of 3D SDIs. On the basis of Oracle Spatial, Stadler et al. (2009) proposed a mapping method for the CityGML Schema to RDBMS, which enabled parallel storage and querying of Berlin CityGML data. However, this approach caters primarily to the representation of simple geometries and struggles to store complex geometries. Subsequently, an open-source RDBMS tool known as 3DCityDB was developed, facilitating the import, management, and analysis of CityGML data (Yao et al., 2018). To manage the dynamic attributes encoded in ADE, a plugin was developed to increase the ability of 3DCityDB to handle CityGML data with ADEs (Chaturvedi et al., 2019). Karnatak and Kumar (2014) conducted tests on various spatial indices within RDBMS and reported that R-Tree and GiST often offer better performance for 3D spatial data queries.

Nonrelational databases (NoSQL) are typically built on distributed architectures, offering advantages such as high performance and elastic scalability. NoSQL have been applied in 3D spatial data management. For example, to address the challenges posed by large-scale 3D data, a strategy based on Hadoop was proposed to load 3D wavefront OBJ data efficiently (Luan et al., 2014). Mao et al. (2014) employed NoSQL to manage

increasingly complex 3D urban models, incorporating geographic indexing to increase query speeds.

Since diverse and complex 3D spatial data are difficult to manage, some efforts have focused on the management of specific attributes of 3D data. For example, Karim et al. (2022) introduced a scale-unique identifier to support cross-scale CityGML querying. Chadzynski et al. (2021) treated city objects within CityGML as distinct entities. These authors proposed ontological methods for urban geometric entities and utilized a graph database to manage 3D data. In addition, an event-driven spatiotemporal database was proposed to perform dynamic updates of 3D city models (Guo et al., 2016). In general, existing research has focused mainly on the management of CityGML data, hindering the ingestion of diverse 3D spatial data from various providers. Additionally, the complexity of 3D data structures requires component-level CRUD operations, which current studies fail to support. The large scale and volume of 3D data also demand streaming services, a factor not considered in existing management methods.

3. Improved conceptual model for component-level management of multisource 3D spatial data

3.1. Requirements from SDIs for 3D data

To enable the discovery and delivery of spatial data from repositories, the basic requirements of SDIs are as follows: 1) data management for CRUD operations of spatial data, 2) a data catalog for discovering and browsing spatial data, 3) a data service for allowing the delivery of the data, and 4) data processing for continuous data ingestion.

Integrating 3D data into SDIs introduces additional requirements as follows: 1) component-level management for hierarchically complex 3D spatial data, 2) streaming

services for large-scale 3D spatial data, and 3) semantic expression and extension of 3D spatial data for 3D geoanalytics.

Furthermore, to manage 3D spatial data at the component level, a DBMS-oriented information model that can easily inject various semantic 3D data is needed. With the vast volume of large-scale 3D models, streaming the managed data directly is required to minimize deserialization costs. To achieve these goals, this section introduces a concept model for multisource 3D information management at the component level. This model is mapped to a cloud-optimized encoding for streaming the managed data directly in Section 4.

3.2. Overview of 3DSIM CM

In 3DSIM CM, we adopt the principles of SceneGraph in computer graphics to logically organize the components of complex 3D spatial data using the bounding volume hierarchy (BVH). The BVH is represented by the associations and edges between 3D assets in Fig. 2, forming the foundation for CRUD operations on components. 3DSIM CM is designed according to the commonalities across diverse 3D formats, such as glTF, CityGML, netCDF[12], and 3D tiles, for better compatibility and scalability. We abstract 3D spatial data as a 3D asset described by *Abstract3DAsset*, which is inherited from geographic features (i.e., *AnyFeature*), as defined by ISO 19109. 3D assets not only encompass spatial geometries, textures, and materials but also include semantic attributes.

According to hierarchy complexity, we categorize 3D assets into two types: *3DScene* and *Abstract3DModel*. *3DScene*, such as a residential area, represents a complex 3D geographic scene consisting of multiple leaf nodes (*Abstract3DModel*) or group nodes (*3DScene*), whereas *Abstract3DModel*, such as a building model,

characterizes a 3DSI composed of one or more geometries and their textures and materials, usually stored in a single 3DSI instance file such as glTF.

We employ the geosolid coordinate reference system (CRS) EPSG[13] 4978 as a standardized spatial datum. To support 3D assets whose CRS (*localCRS*) is not EPSG 4978, a nonnull transformation matrix (*transform*) and a location of the reference point in the world coordinate system (*originLocation*) are designed to correctly position them, as shown in Section 3.3.

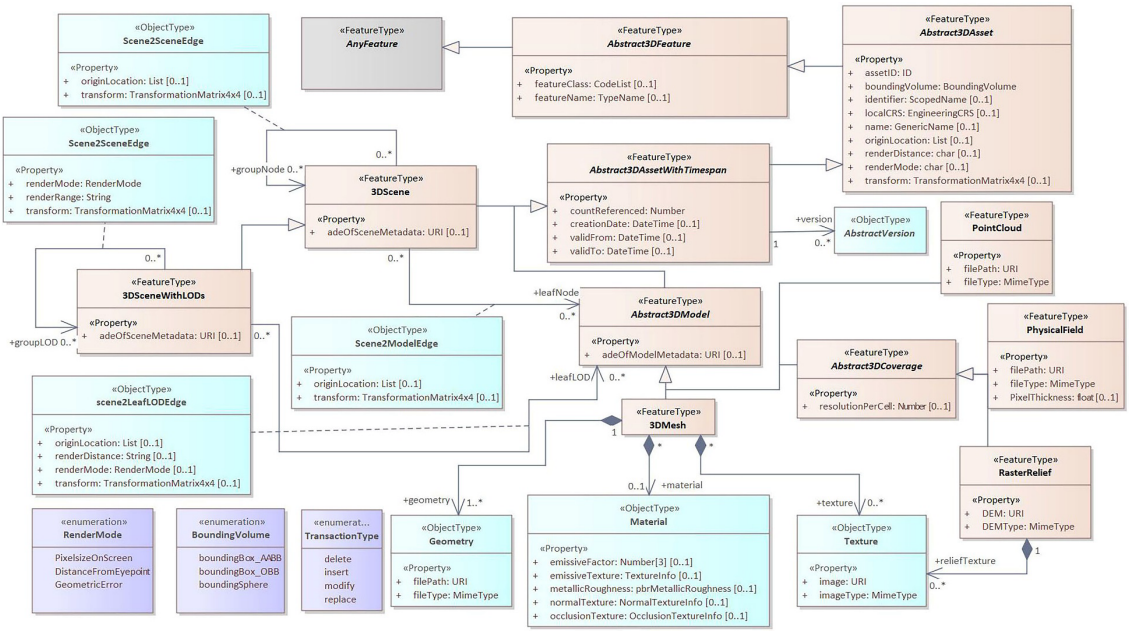


Figure 2. UML conceptual model for component-level management of multisource 3DSI.

3.3. Multigrained representation and composition of 3D assets on the basis of the bounding volume hierarchy

3DMesh, *PointCloud*, and *Abstract3DCoverage* are derived from *Abstract3DModel* to represent common 3D mesh models, point clouds, and 3D coverage data, respectively. For *Abstract3DCoverage*, we define two commonly used forms in GIS: terrain (*RasterRelief*) and the temporal 3D physical field (*PhysicalField*). *3DMesh* is primarily represented by geometry, texture, and material properties, which commonly follow the

glTF specification. For *RasterRelief* and *PhysicalField*, common file formats include GeoTIFF[14] and NetCDF.

We adopt BVH, a type of directed acyclic graph (DAG), as shown in Figs. 2 and 3, to represent the structure of *3DScene*. The nodes constituting *3DScene* are classified into the *leafNode* and the *groupNode*, both of which have the mandatory *boundingVolume* attribute for identifying the spatial extent of the node to support culling during collision detection.

The *3DScenesWithLODs* are derived from *3DScene* for multilevel representation of 3D assets. Specifically, its *groupLOD* and *leafLOD* are associated with other 3D assets through *LODScene2ModelEdge* and *LODScene2SceneEdge*. *renderMode* controls how the range values should be interpreted when active nodes are determined. *renderDistance* specifies the range that determines when the node is loaded or rendered. For example, if the *renderMode* is *DistanceFromEyepoint* and the *renderDistance* is [100, 500], the node is rendered when its center point distance from the viewpoint is between 100 and 500 units.

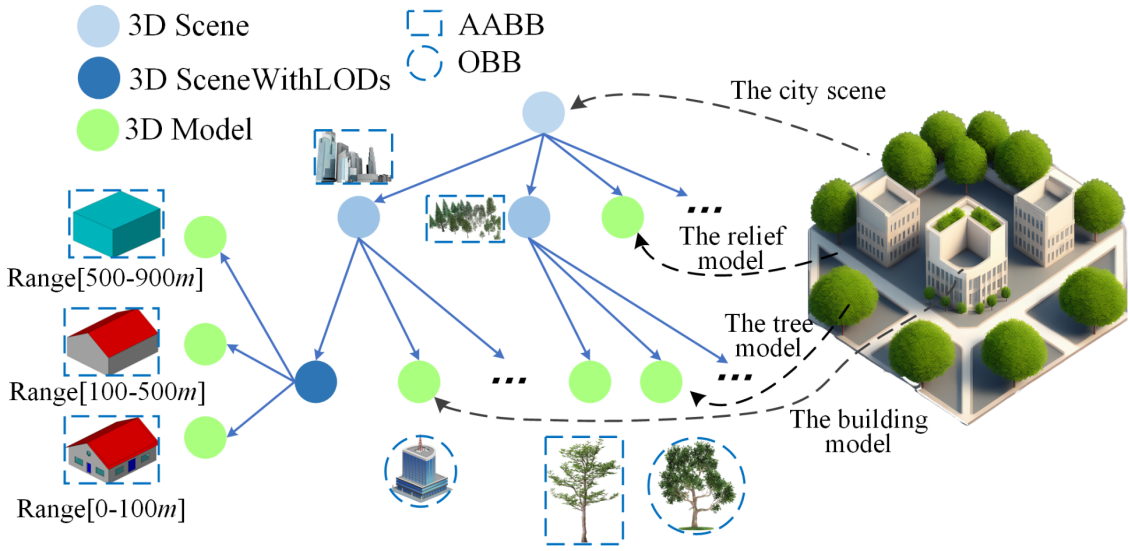


Figure 3. Bounding volumes, BVHs, 3D models, 3D scenes, and LODs.

We use node references and transformation to enable any 3D assets to be a *groupNode* or *leafNode* of a larger scene, allowing for flexible 3D spatial data

composition. An advantage is the reduction of storage waste in scenes that consist of numerous repetitive 3D models, such as trees in a city scene. The *transform* and *originLocation* within the *Scene2SceneEdge* and *Scene2ModelEdge* objects describe how a *groupNode* or *leafNode* is referenced accurately within a 3D scene. *transform* is an affine transformation matrix that acts upon the vertices and *boundingVolume* of the 3D asset. For a vertex v with homogeneous coordinates $p = (x, y, z, w)$, its transformed position is expressed via Eq. 1:

$$p' = (T_1 * p^T)^T, \tag{1}$$

where T_1 is an affine transformation matrix. When hierarchical transformations are involved, they transform sequentially in a bottom-up manner. For example, the process of transforming the model instance in Fig. 4 into a city scene instance is calculated via Eq. 2:

$$p' = (T_6 * (T_1 * p^T))^T. \tag{2}$$

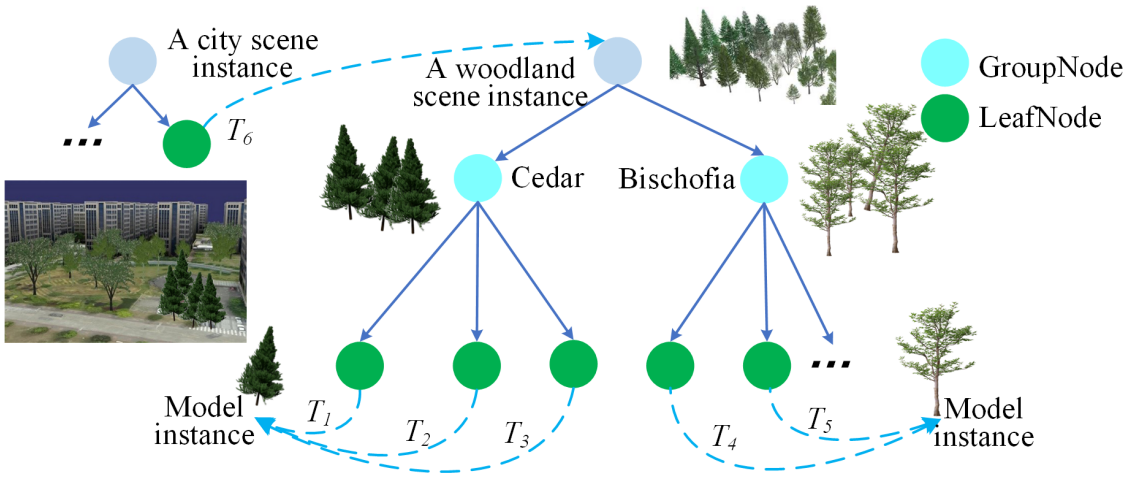


Figure 4. Node references and transformations.

The UML classes shown in Fig. 2 define the minimal attributes required for scene organization, visualization, etc. Additionally, ADEs are introduced to extend more metadata, which are required for unique application domains. As shown in Fig. 5, the extended metadata are described as key-value pairs. They can reside in structured

file types or databases optimized for key-value storage, which are then externally referenced through the *adeOfSceneMetadata* and *adeOfModelMetadata* attributes in the 3D assets. This solution allows unforeseen metadata to be flexibly defined while associations are maintained with 3D assets.

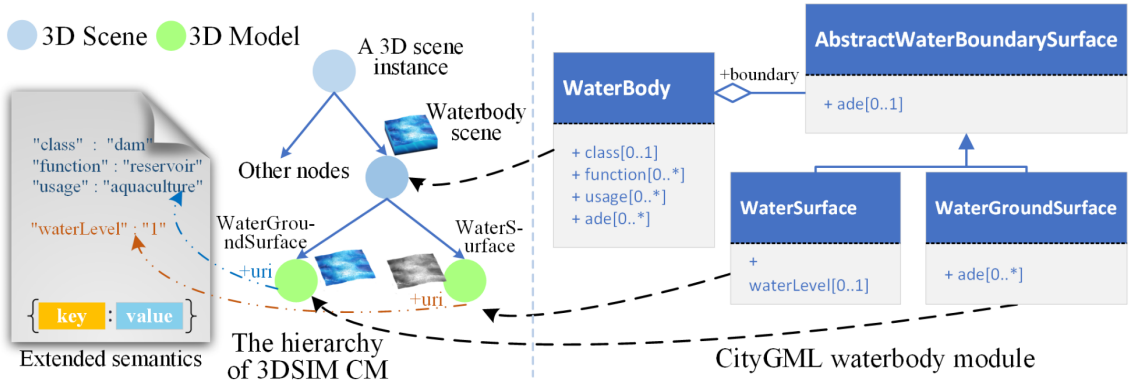


Figure 5. Attribute mapping of the waterbody module in CityGML to a 3D asset described by 3DSIM CM.

3.4. Fine-grained lifecycle management of 3D assets

We further specialize *Abstract3DAsset* into *Abstract3DAssetWithLifeSpan*. The *validFrom* and *validTo* attributes are used to represent the lifespan of the asset in the real world. This can be utilized to query the morphology and appearance of geographic scenes within specific time ranges. The *countReferenced* attribute is used to identify the survival status of the 3D asset in the DBMS. The initial value of *countReferenced* for each 3D asset is 1. The *countReferenced* is increased by 1 each time the 3D asset is referenced by another 3D scene and decreased by 1 whenever the asset needs to be removed from the DBMS. The asset can be completely deleted only when *countReferenced* = 0. This mechanism ensures consistency of the asset across all 3D scenes, preventing the scenario in which deleting a 3D asset would invalidate the 3D scene referencing it.

Furthermore, to support the semantic or geometric evolution of 3D assets, we designed a *versioning* module inherited from *AbstractVersion*. This module defines the concept of 3D assets having multiple version representations. As shown in Fig. 6, 3D assets can have multiple *versions* representing their state at different times or stages.

The *versionTransition* attribute of the *version* describes changes from a previous version (*reliantVersion*) to the current version, detailing the applied modifications. Each change is defined by a *transaction* with the type enumerated as *delete*, *insert*, *modify* or *replace*. Different versions of the same 3D asset share the same *identifier* value.

The *versioning* module enables tracking and management of variations in the geometric and semantic attributes of 3D assets longitudinally, encompassing alterations to its SceneGraph constituents (e.g., arboriculture events such as planting orexcavating vegetation), modifications to its geometric constitution (e.g., reconstructing a building), and transitions in the status of the object (e.g., modifying the *feature* or *timespan*).

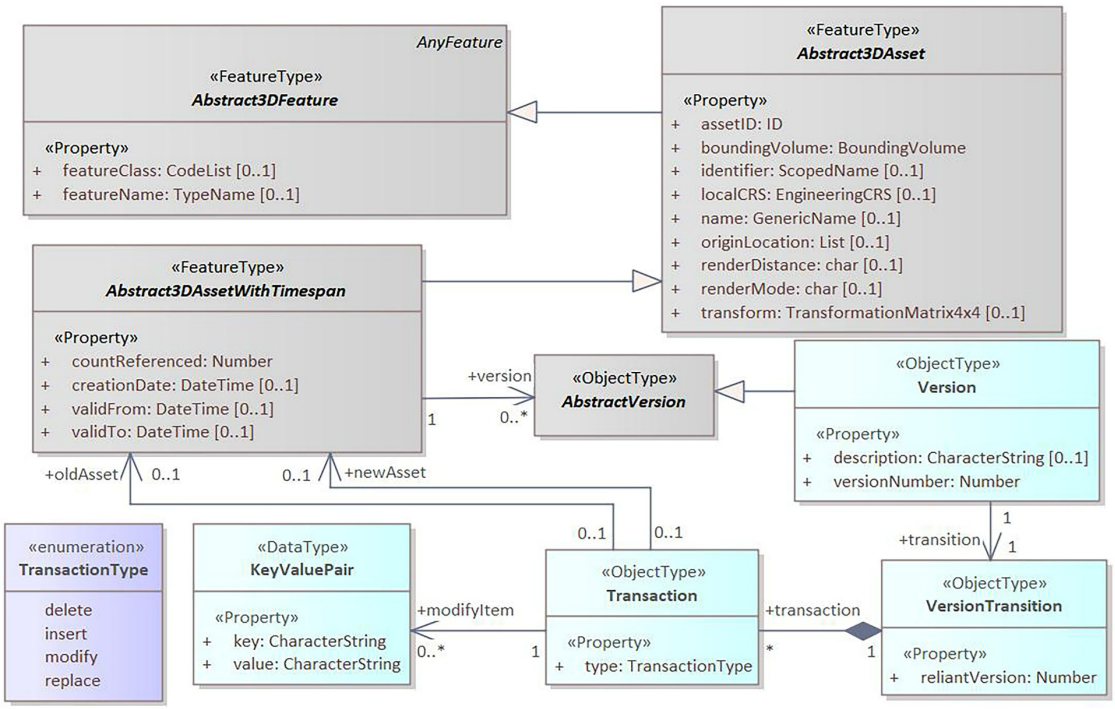


Figure 6. UML conceptual model for the versioning module of 3DSIM CM.

3.5. Extension of CRUD operations for 3DSIM CM

While CRUD operations are well known as basic operations for data management, complex 3D data structures introduce new requirements for CRUD operations.

The IDs of the desired 3D assets can be easily obtained through the *retrieve* operation. However, to obtain the complete *3DScene*, it is necessary to further identify the 3D assets that compose it to construct the BVH. This process can be implemented through a depth-first search approach, as shown in Algorithm 1.

The *create* and *delete* operations for a *3DScene* introduce the requirement of atomicity. This means that unless all nodes of a *3DScene* are fully created/deleted, no nodes are created/deleted. In addition, to avoid the deletion of assets used by other scenes, the *delete* for 3D asset a_i is defined via Eq. 3:

$$delete(a_i) = let\ (a_i.countReferenced -= 1). \tag{3}$$

Updating the attributes of asset a_i directly may trigger many propagations across 3D assets referencing it, potentially causing inconsistencies between datasets. To avoid propagation, all update operations are recorded within the *versioning* module. The *update* for a_i involves adding a version that includes all the *transactions* and *transitions* associated with the update.

Algorithm 1 ***Traverse a 3D scene.***

Input:
A 3D scene, s_i;
Output:
The 3D scenes that make up s_i, $\mathcal{S}_i$; the 3D models that make up s_i,$\mathcal{M}_i$;

$\mathcal{S}_i \leftarrow \emptyset$; $\mathcal{M}_i \leftarrow \emptyset$; Initialize a $stack$;
$stack.push(s_i)$;
while $stack$ is not empty **do**
  $s' = stack.top();\ stack.pop();$
  $\mathcal{S}_i = \mathcal{S}_i \cup s'$;
  for each leafNode m of s' **do**
    $\mathcal{M}_i = \mathcal{M}_i \cup m$;
  for each groupNode s of s' **do**
    $stack.push(s)$;
Return $\mathcal{S}_i$ and $\mathcal{M}_i$

4. Cloud-optimized encoding for 3DSIM CM

4.1. Cloud-optimized database schema for encoding 3DSIM CM

To facilitate efficient CRUD operations, we employ the constellation schema to map the concepts and structures of 3DSIM CM into a database-based and system-independent encoding structure suitable for cloud-based storage. As shown in Fig. 7, the attributes most frequently utilized for retrieving 3D assets are treated as dimensions. The other attributes, relationships and versions of 3D assets are documented as facts, which are typically implemented using the NoSQL database due to the large number of records. The geometry encoding is stored in cloud-based object storage services (OSSs) and associated with the facts through the *filePath* attribute.

The dimensions include *timeDimension*, *spatialDimension*, *featureDimension*, *productDimension*, and *viewpointDimension*. Each dimension documents a specific classification or hierarchy, enumerating the valid dimension values. For the spatial dimension, as users commonly utilize latitude and longitude for data retrieval, *spatialDimension* is encoded via the EPSG 4979 CRS. Facts consist of *3DModelAssetFact*, *3DSceneAsstFact*, *assetEdgeFact*, *versionTransistionFact*, and *transactionFact*. Each 3D asset is encoded as a *3DModelAssetFact* or *3DSceneAsstFact*. The *assetEdgeFact* documents the BVH structure of *3DScene*, with the *type* specifying whether it corresponds to *scene2SceneEdge*, *scene2LeafLODEdge*, or *scene2ModelEdge* of 3DSIM CM.

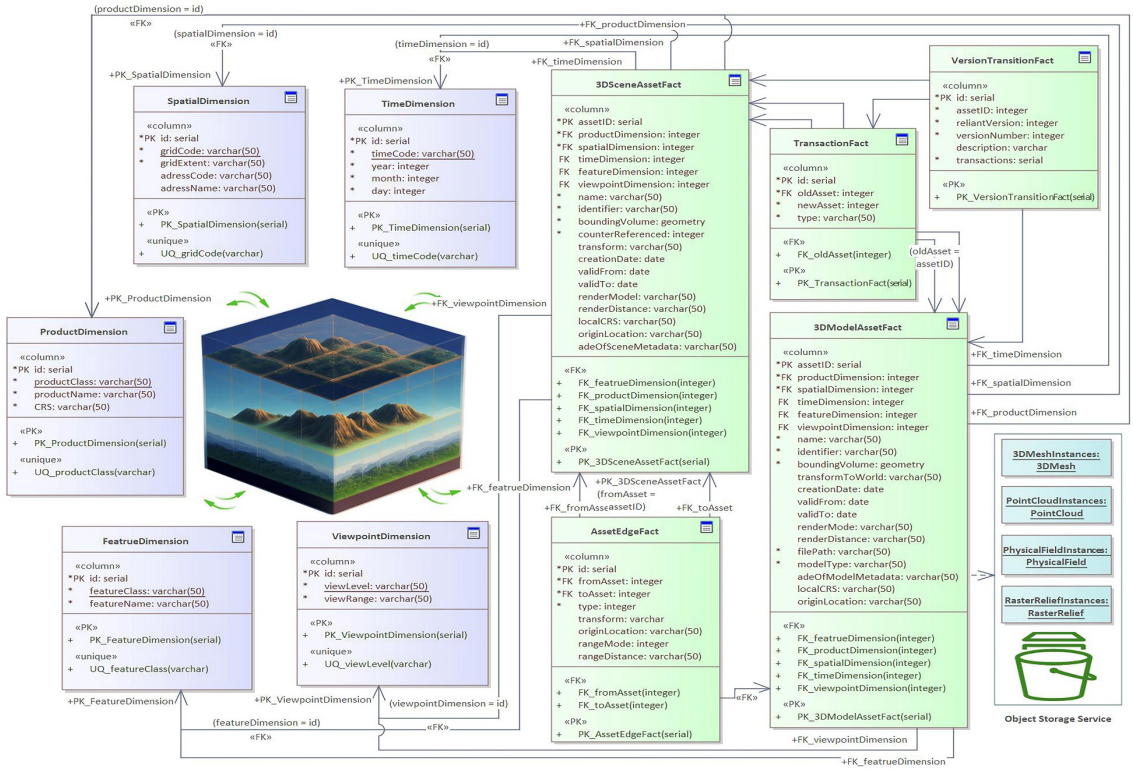


Figure 7. Cloud-optimized database schema for encoding 3DSIM CM.

4.2. Efficient 3D model streaming delivery using an object storage service

The geometry encoding of *3DModels* often comprises millions of primitives and vertices. Moreover, large-scale *3DScenes* often contain many *3DModels*, which are required for streaming delivery to clients. Storing vast geometric data directly in databases results in inefficient storage utilization and deteriorated performance and impedes direct streaming. On the other hand, an OSS facilitates the elastic storage of unstructured objects and offers HTTP(S) byte-stream access to any stored object.

Therefore, unlike prior works, we store the geometry encoding of the *3DModel* in an OSS. As shown in Fig. 8, the geometry of the *3DModel* is stored as objects in the OSS when they are mapped to the database schema. By detaching the geometry encoding of the *3DModel* from the database, the database can focus solely on storing and managing the semantics and structure of 3D assets, without the overhead of deserializing large-volume geometries.

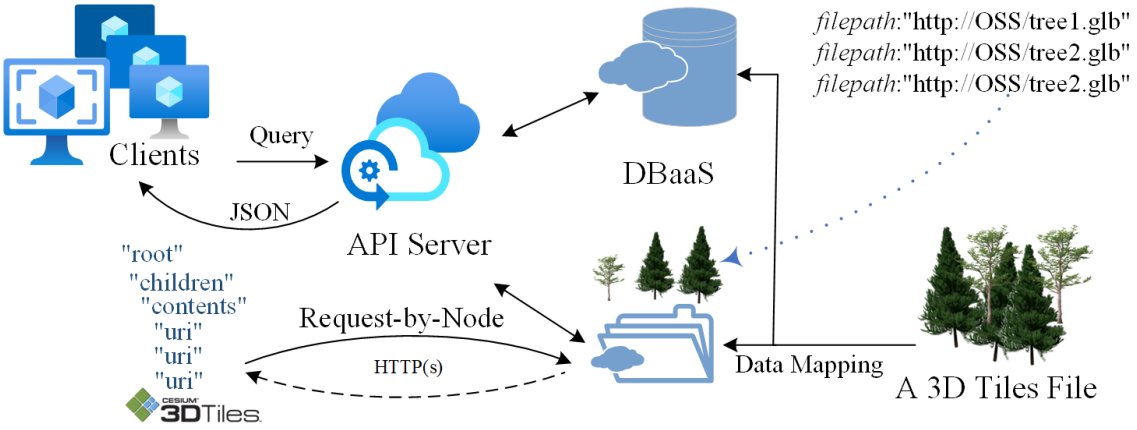


Figure 8. Schema mapping and streaming delivery of 3D assets in a cloud-based OSS.

Additionally, clients can directly access the required geometries of models via HTTP(s) to facilitate streaming without downloading the entire *3DScene*. Since clients can stream data on demand, they do not need local replicas, reducing overall data duplication. The efficient stream service workflow is shown in Fig. 9. In this process, the client initiates requests to the server via the 3D GeoVolumes for complex 3D scenes composed of various types of 3D assets. As the user subsequently shifts their current viewpoint within the client application, the client dynamically accesses the geometry encodings of *3DModel* from the OSS.

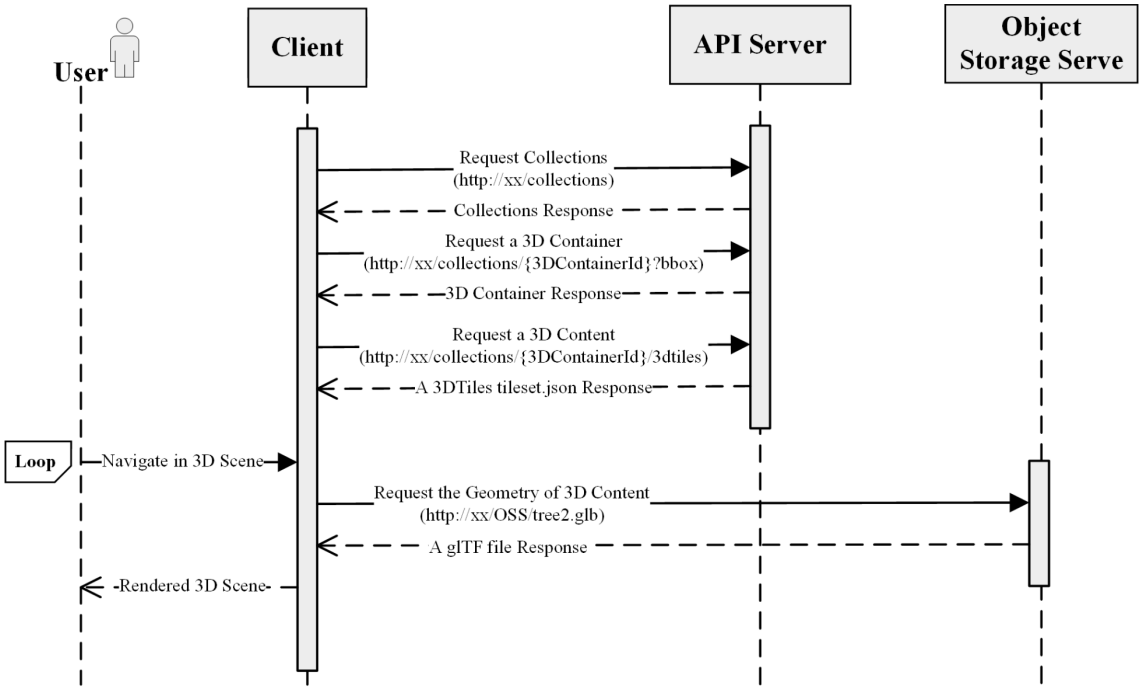


Figure 9. Sequence diagram of 3D asset cloud services in SDIs on the basis of 3D GeoVolumes and OSSs.

3D GeoVolumes integrates various specifications into an open standard-based solution, enabling applications to request diverse 3D data from different providers in an interoperable manner. Inherently, each 3D dataset is imbued with a bounding volume (referred to as the *extent* attribute) to form a geoVolume. Multiple geoVolumes can then be hierarchically combined to form a geoVolume with a relatively large geographical scope. Semantically, the geoVolume can be considered an abstract representation of the 3D asset defined by 3DSIM CM, because both impose bounding volume requirements. Consequently, the mapping of 3DSIM CM onto the service API of 3D GeoVolumes is straightforward. Table 3 shows the correspondence between the primary objects of 3DSIM CM and 3D GeoVolumes. For example, a 3D scene composed of diverse types can be mapped as a container resource within 3D GeoVolumes.

Table 3. Mapping between the primary objects of 3DSIM CM and 3D GeoVolumes

3D GeoVolume	*3DSIM CM*	*HTTP Path*	*Description*
geoVolume	3D asset	none	3D content with bounding volumes
collections	multi scenes	/collections	collection of containers
container	3D scene	/collections/{3DContainerId}	geoVolume structured hierarchically from multiple geoVolumes
dataset	multi 3D asset instances	/collections/{3DContainerId}/…/{datasetId}	collection of contents
content	3D asset instance	/collections/{3DContainerId}/…/{datasetId}/{contentId}	3D asset instance file such as 3DTiles and glTF

5. Implementation and evaluation

In this section, we present an implementation allowing CRUD, streaming optimization, and 3D geospatial analysis of various 3D assets on the web. We start with a presentation of the software environment. We continue with a proof-of-concept experiment, which demonstrates that our propositions provide efficient component-level retrieval, composition, access, and 3D geoanalytics cases.

5.1. Software implementation

The implementation for encoding 3DSIM CM uses RDBMS, NoSQL, and OSS. The dimension tables were instantiated in PostgreSQL, whereas the fact tables were stored in MongoDB. The geometry of 3D assets was stored in MinIO. The API server was implemented with Django and Python to handle requests from clients and organize retrieved information into the required formats. Client-side processing of 3D spatial data was implemented in CesiumJS and JavaScript.

5.2. Proof-of-concept experiments and evaluation

5.2.1. Component-level management and delivery for 3D assets within SDIs.

On the basis of 3DSIM CM, we developed an SDI that enables the management, access, and delivery of 3D spatial data in a coordinated and efficient way. In this system, various 3D spatial data can be easily ingested and managed at the component level, with semantics, LODs, ADE, and versioning properly organized to support client retrieval and analysis needs. Owing to the encoding being tailored for modern cloud storage, highly efficient component-level CRUD operations for 3D assets are implemented. As shown in Fig. 10, the client-side interface for semantic retrieval of 3D assets uses dimensions such as spatial extent, feature type, and product type (as depicted in Fig. 7) to retrieve the necessary 3D assets and access the component nodes comprising 3D scenes. The CRUD performance is evaluated in Section 5.2.3.

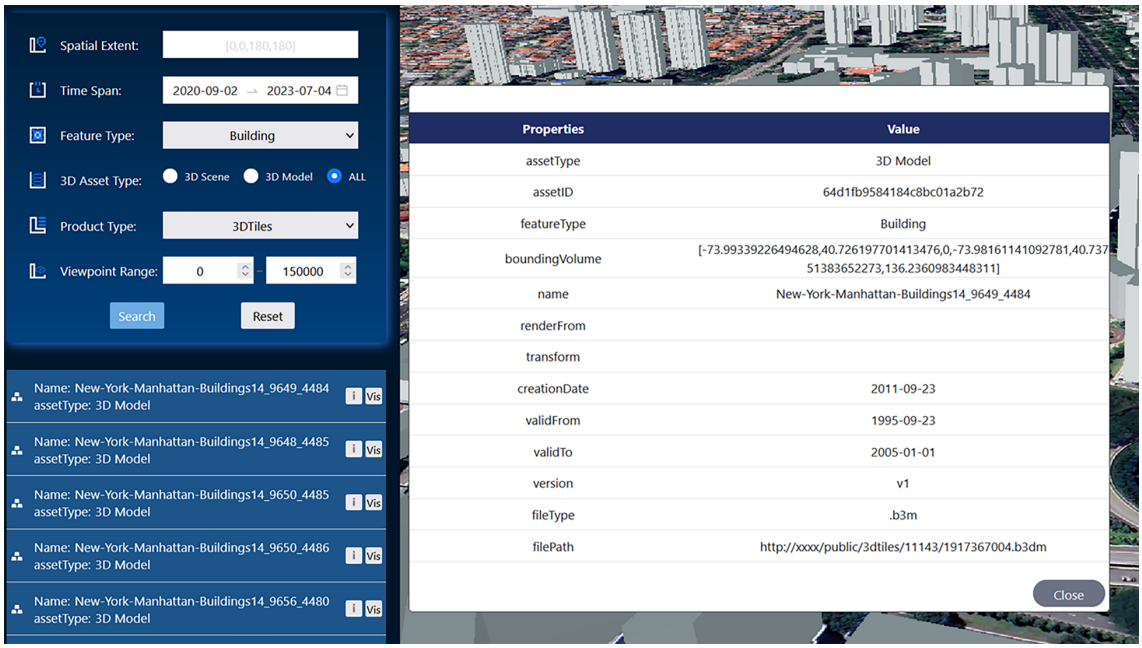


Figure 10. Client-side graphical user interface for retrieving 3D assets from SDIs.

An advantage of the proposition is that the geometries of 3D assets managed within the SDI can be streamed on demand from OSS to clients via HTTP(s). After the desired assets are retrieved from Fig. 10, users can obtain the semantics of 3D assets delivered in JSON and directly request the geometry of each node from the OSS via the *filePath* attribution. Additionally, we implemented interoperable standards such as glTF, 3D Tiles, and I3S to deliver the managed 3D assets.


```
{"collections": [{
    "id": "Singapore",
    "title": "Singapore Thread for OGC FMSDI",
    "description": "All Supported 3D Containers for Singapore Thread",
    "collectiontype": "3d-container",
    "extent": {
        "spatial": {
            "bbox": [ ···
            ],
            "crs": "http://www.opengis.net/def/crs/EPSG/9.9.1/4979"
        }
    },
    "links": [ ···
    }],
    "content": [{
        "title": "3D Tree Mesh",
        "rel": "original",
        "href": "http://xxx/GeoVolumes/collections/Singapore/tree.glb",
        "type": "application/gltf"
    }, {
        "title": "Store Field",
        "rel": "original",
        "href": "http://xxx/GeoVolumes/collections/Singapore/p-sg-42.nc",
        "type": "application/netcdf"
    }, {
        "title": "3D Relief",
        "rel": "original",
        "href": "http://xxx/GeoVolumes/collections/Singapore/sg-relief.tif",
        "type": "application/geotiff"
    }],
    "children": [{
        "id": "blk",
        "title": "Singapore Buildings",
        "collectiontype": "3d-container",
        "extent": {···
        },
        "links": [ ···
        }],
        "content": [{
            "title": "Photo-realistic 3D Buildings",
            "rel": "original",
            "href": "http://xxx/GeoVolumes/collections/Singapore/blk/3dtiles",
            "type": "application/gltf"
        }, {
            "title": "Store Field",
            "rel": "original",
            "href": "http://xxx/GeoVolumes/collections/Singapore/blk/citygml",
            "type": "application/xml-citygml"
        }],
        "children": []
    }]
}]}
```


Figure 11. JSON encoding of the service API using 3D geovolumes.

Another advantage is its support for component-level retrieval and assembly, which enables the retrieval of subparts of complex 3D scene assets and then combines

these subparts with other 3D assets to create a new 3D scene for delivery to the client in 3D GeoVolumes. The JSON encoding of its service API, as shown in Fig. 11, reveals how different types of 3D instances are mapped as contents within the 3D GeoVolumes. Fig. 12 presents a rendered 3D scene comprising a diverse assembly of various types of retrieved 3D assets. This scene is delivered to the client through 3D GeoVolumes and encompasses 3D storms, buildings in 3D Tiles, buildings in CityGML format, and relief data.

Figure 12. Rendered 3D scene comprising diverse 3D assets via 3D GeoVolumes

5.2.2. 3D geospatial analysis cases supported by 3DSIM CM

One primary motivation for building an SDI for 3D is to support the multifaceted needs of 3D geoanalytics. We present several case studies to demonstrate how the proposed method can be used in 3D geospatial analysis. Fig. 13 shows an online analytical processing (OLAP) case using multisource 2D and 3D spatial data in a geo-computing platform that integrates the proposed method for managing and delivering 3D spatial data. This case simulates a flood disaster caused by prolonged rainfall. The simulation calculates the flood inundation areas and depths under specified rainfall amounts on the

basis of vector data, reliefs, and 3D building meshes. The vector data are managed by other modules of the platform. The reliefs and 3D building meshes are managed and delivered in 3DSIM CM and provide the necessary elevation and permeability information for flood simulation.

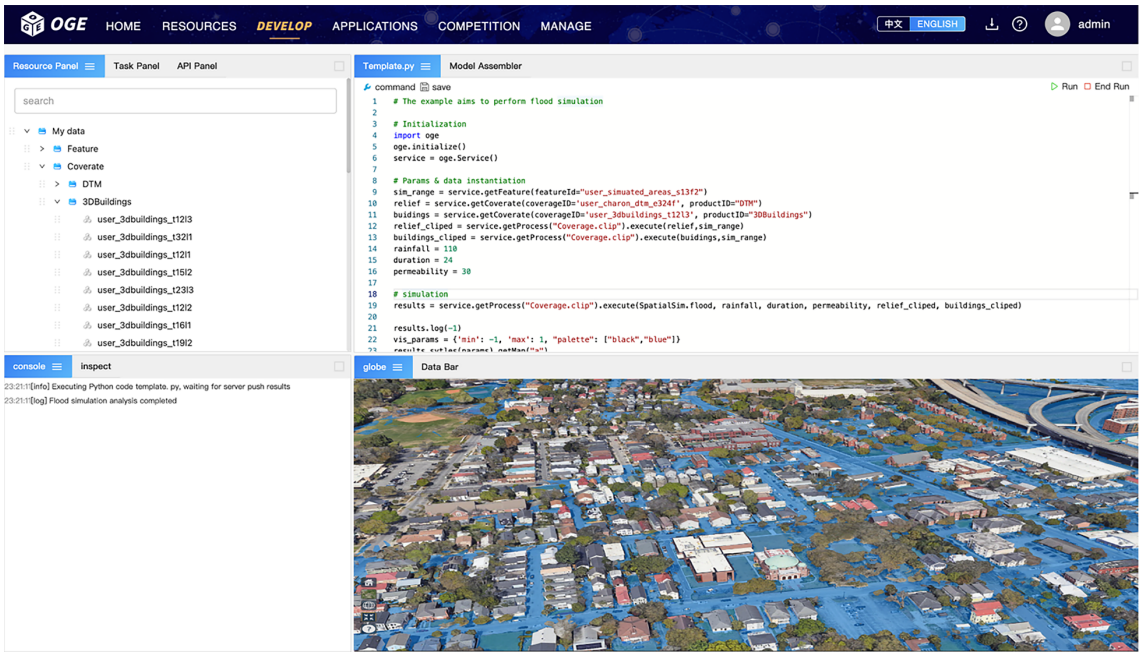
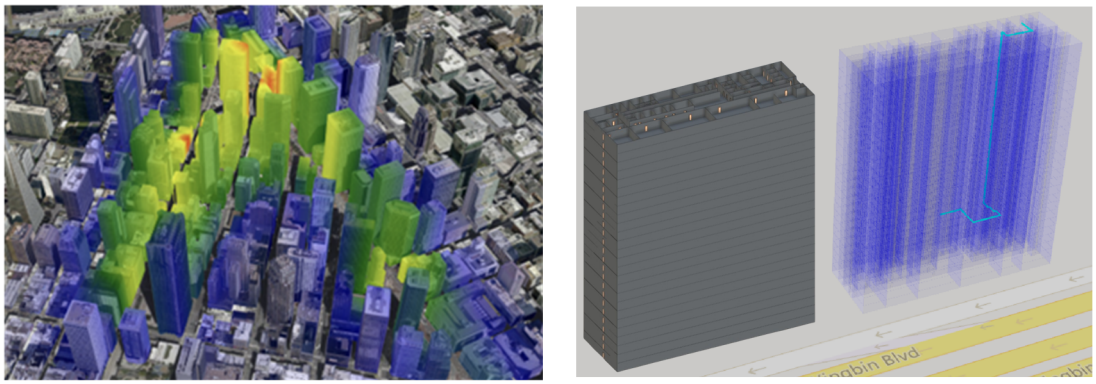


Figure 13. Online analytical processing for flood simulation in a geo-computing platform on the basis of the proposed method.

Fig. 14 shows 3D geospatial analyses that require extended semantics. Fig. 14a displays a noise map simulated using 3DSIM CM and its encoding. This map was computed on the basis of the emission locations of the noise sources and the 3D environment in which the noise propagates. The noise distribution on the building surfaces was calculated using a ray-tracing algorithm, and the required reflection and absorption coefficients of the buildings were recorded in the *adeOfModelMetadata* attribute. Fig. 14b depicts a case of indoor route planning within a 3D building. Using the Astar algorithm, this case implemented the shortest route planning across multiple stories in a 3D building managed in 3DSIM CM. The building was originally in IFC format. As shown in Fig. 15, the IFC data are mapped to 3DSIM CM in an SDI, where

classes such as *IfcWall* and *IfcStair* are converted to leaf or group nodes. Geometry is extracted as glTF, and complex semantics are extracted as key-value pairs in *adeOfModelMetadata* or *adeOfSceneMetadata*. To conduct cross-level shortest path analysis, first, we create floor-level paths on the basis of the geometric positions of corridors. Second, an indoor multistory topology is constructed on the basis of the geometric positions of the stairs and elevators.



(a) (b)

Figure 14. 3D geospatial analyses that require extended semantics: (a) noise propagation simulation and (b) rendered BIM building and results for indoor route planning.

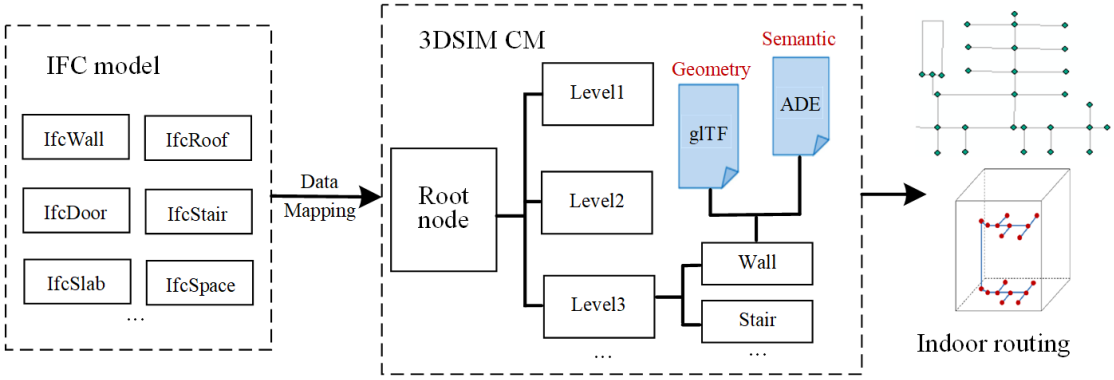


Figure 15. Shortest route planning across multiple stories in a 3D building managed in 3DSIM CM.

5.2.3. Performance evaluation

This section presents experiments to demonstrate management performance. For the sake of measurement, performance evaluation experiments were conducted exclusively on a single Linux server with Xeon CPU E5-2690 V4 and 225 TB Raid drivers.

Table 4. Databases with various data volumes used for testing.

Database ID	*Number of 3D Model Facts*	*Number of 3D Scene Facts*	*Number of Scene Edge Facts*	*Total Number of Facts*	*Data Volume of Geometry*
DB1	23 *K*	27 *K*	50 *K*	100 *K*	4.5 *GB*
DB2	231 *K*	270 *K*	500 *K*	1 *M*	45 *GB*
DB3	23.1 *M*	27 *M*	50 *M*	100 *M*	4.5 *TB*
DB4	0.29 *B*	0.25 *B*	0.55 *B*	1.1 *B*	49.5 *TB*

Table 4 presents the data records of four different databases used in the tests, ranging from 100 thousand (*K*) to one billion (*B*) total facts. For CRUD operations on *3DModels*, typically, a single query is involved; thus, it usually takes milliseconds. However, for CRUD operations on *3DScenes*, the time depends on the number of nodes involved. The more nodes there are, the longer a query takes.

In Fig. 16, we compare the proposed method with 3DCityDB (Yao et al., 2018) in terms of retrieval, creation, and update efficiency of 3D scenes in databases of various sizes. 3DCityDB has been in productive and commercial use for more than 14 years worldwide. The geometric volumes of the 3D scenes used are 0.4 *GB*, 0.8 *GB*, and 1.5 *GB*.

As shown in Fig. 16a, with increasing database volume, the retrieval and creation efficiency of 3DCityDB significantly decreases. In contrast, our method can retrieve and create the same 3D scenes in just a few seconds, as shown in Fig. 16b, regardless of the database volume ranging from 4.5 *GB* to 45 *TB* and the number of facts from 100 *K* to 1.1 *B*. This finding indicates that the database size has no substantial impact on the efficiency. However, the complexity of the scene significantly affects efficiency, with longer times required as the number of child nodes increases. As outlined in Table 4 and Fig. 16, the efficiency of 3D assets from the database is

independent of the volume of geometry stored in OSS, and these assets can be directly delivered to the web side as needed. The challenge posed by the massive data volumes encountered in 3D SDI construction is effectively addressed.

As shown in Fig. 16c, the update time depends on the number of modifications involved. For example, updating the geometry of a node may change the bounding volumes of multiple parent nodes. 3CityDB accelerates updates to semantic attributes, but updates to geometry are relatively slower. Our method shows nearly identical update times for geometry and semantic attributes, and 4.41 seconds is required for 2,000 updates.

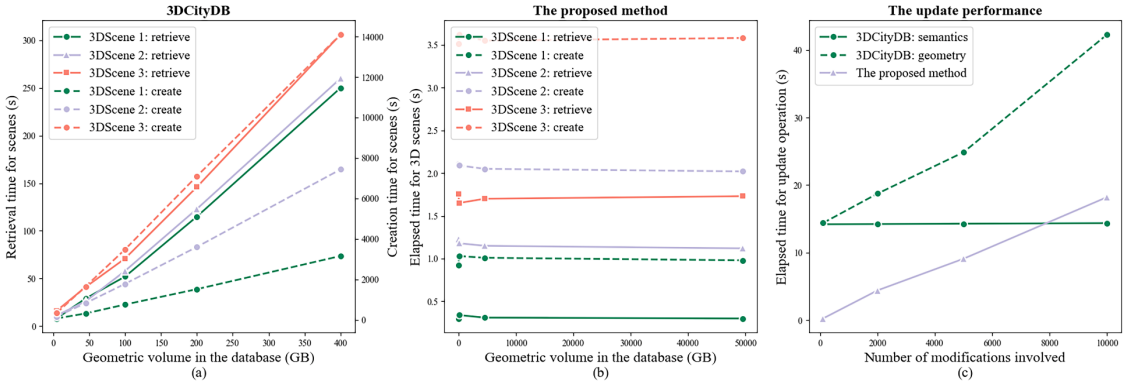


Figure 16. Performance comparison between 3DCityDB and the proposed method: Retrieval and creation times for different scenes using 3DCityDB (a) and the proposed method (b); (c) updated performance comparison for DB4.

6. Discussion

An increasing amount of geographic information is being captured in 3D. However, the description, management, and sharing of existing 3D spatial data are still performed in ad hoc ways and remain uncoordinated. The management, service, access, visualization, analysis cases, and performance evaluation of the proposed method are explored. The following observations are worthy of further discussion:

6.1. Method applicability

One key challenge is to improve the generalizability of 3DSIM CM by applying it to various 3D cases. In this work, we collect multiple 3D data and investigate their management and services within the SDI.

The following lessons on how to convert and inject existing 3D spatial data can be learned: 1) the 3D asset type to which the 3D spatial data asset belongs should be identified; 2) nonstream optimized geometric encodings, such as the GML encoding of CityGML, need to be converted to stream-optimized formats, such as glTF; and 3) the hierarchical semantics of 3D scenes should be added as nested key-value pairs to the *adeOfSceneMetadata* or *adeOfModelMetadata* of the corresponding assets.

With respect to the data partition, some 3D spatial data may be organized into hierarchical or modular structures. We consider the hierarchy to be built during data ingestion for the following reasons: 1) Hierarchical or modular structures can easily be mapped to the SceneGraph used in this paper, 2) data repartitioning is challenging because of the required topology reconstruction, and 3) decoupling geometry from the database helps avoid high deserialization costs.

Low-frequency variations can be recorded in *versioning*, whereas high-frequency changes, such as sensor data, can be integrated with our approach at the client to meet specific needs for time series analysis. However, for animations such as skeletal or keyframes, their implementation often relies heavily on the scene parameters rendered at the client, making predefining a universally applicable animation for all scenarios difficult. Therefore, the current version does not support animation management.

6.2. Data consistency

One key challenge in implementing 3DSIM CM is maintaining consistency among the complex *3DScenes*. This involves avoiding dangling nodes, incorrect referencing, and incomplete data during CRUD operations. The proposed method can help ensure data consistency in the following ways: 1) the *countReferenced* feature prevents incorrect referencing caused by the deletion of used assets while allowing periodic deletion of dangling assets with *countReferenced* = 0, 2) recording all update operations in *versioning* mitigates inconsistencies resulting from propagations across 3D assets, and 3) to maintain data integrity, the implementation of ingestion and deletion of *3DScenes* must be atomic.

6.2. 3D SDIs and geospatial digital twins

A 3D SDI can be regarded as an architecture with capabilities because the various 3D spatial data are interoperable, accessible, and discoverable in a distributed infrastructure. Although this paper provides a solution for integrating the management and streaming service of 3D data in the cloud, additional components are needed to form a complete 3D SDI. The first is the enhanced data ingestion component, which seamlessly facilitates the conversion between various 3D data formats and 3DSIM CM. The second is a catalog component based on OGC Catalog Services or STAC, which enables the discovery of data within repositories. The third is a portal component, which queries, displays, and analyses 3D spatial data.

Geospatial digital twins (GDTs) combine the advantages of virtual geographic environments (Lin et al., 2013) and digital twins to reflect real-world geographic conditions in real time through digital models (Zhang et al., 2024). The construction of GDTs involves large-scale and multisource 3D spatial data (Zhu, 2024). This work provides efficient 3D spatial data management and services for GDTs. Admittedly, the

proposed method is not designed to handle real-time data generated from IoT devices, which are crucial for analysis in GDTs. However, such data can be delivered with solutions such as OGC SensorThings and integrated with our approach in GDTs for time series analysis.

7. Conclusion

Managing and streaming 3D spatial data is important for understanding and analyzing geospatial changes. In this paper, we present design rationales and a unified conceptual model suitable for component-level management of diverse 3D spatial data. The model is subsequently mapped to a cloud-optimized encoding schema to effectively manage and deliver massive 3D spatial data within SDIs. This work provides a scientific exploration that integrates management and services to enable direct streaming of managed 3D spatial data without the need for redundant replication and conversion. The proposed approach is implemented and evaluated across services, accessibility, analysis cases, visualization, and efficiency.

This work helps develop a valuable reference for the management and delivery of 3D SDIs. Future work will consider moving features of 3D spatial data, such as sensor data and skeletal animations. The on-the-fly and batch computing methods for the vast 3D datasets managed within a 3D SDI are also helpful. In the coming months, we will develop a 3D SDI based on the proposed model and encoding solution, which offers online access and computation services for a substantial and varied collection of 3D spatial data and associated operators.

Disclosure statement

No potential conflicts of interest were reported by the author(s).

Acknowledgment

The work was supported by the National Natural Science Foundation of China under Grant 42425108; Chongqing Technology Innovation and Application Development Project under Grant CSTB2022TIAD-DEX0013; Fundamental Research Funds for the Central Universities under Grant 2042022dx0001. This research is part of the project Large-scale 3D Geospatial Data for Urban Analytics, which is supported by the National University of Singapore under the Start Up Grant. The authors acknowledge financial support from China Scholarship Council.

Notes on contributors

Dayu Yu is a lecturer at Nanjing Normal University, and a member of OpenGMS Research Group. He holds a PhD in Cartography and Geographic Information Engineering from Wuhan University. His research focuses on 3D GIS, high-performance geo-computing, and geographic information services.

Peng Yue is a Dean's Chair professor at Wuhan University. He serves as the director at Hubei Province Engineering Center for Intelligent Geoprocessing and the director at the Institute of Geospatial Information and Location Based Services.

Binwen Wu is a Master student majoring in Cartography and GIS from Wuhan University. His research focuses on high-performance geocomputing.

Filip Biljecki is an Assistant Professor at the National University of Singapore and the principal investigator of the NUS Urban Analytics Lab. He holds an MSc in Geomatics and a PhD in 3D GIS from the Delft University of Technology in the Netherlands.

Min Chen is a Professor with the School of Geography, Nanjing Normal University, and a member of OpenGMS Research Group. His research interests include virtual geographic environment, geographic modeling and simulation, and geographic information system.

Luancheng Lu is an undergraduate student majoring in Photogrammetry and Remote Sensing from Wuhan University.

ORCID

Dayu Yu: https://orcid.org/0000-0003-1720-8302

Peng Yue: http://orcid.org/0000-0003-3006-4542

Filip Biljecki: http://orcid.org/0000-0002-6229-7749

Min Chen: https://orcid.org/0000-0001-8922-8789

Data and codes availability statement

The codes and data that support the findings of the present study are available on DOI:

10.6084/m9.figshare.27643416

Endnote

[1] https://docs.ogc.org/cs/22-025r4/22-025r4.html
[2] https://www.ogc.org/standard/i3s/
[3] https://www.ogc.org/standard/citygml/
[4] https://www.ogc.org/standard/cdb/
[5] https://www.khronos.org/gltf/
[6] https://www.w3.org/XML/
[7] https://www.cityjson.org/
[8] https://ogcapi.ogc.org/geovolumes/
[9] https://technical.buildingsmart.org/standards/ifc/
[10] https://www.ogc.org/standard/infragml/
[11] https://www.ogc.org/standard/indoorgml/
[12] https://www.ogc.org/standard/netcdf/
[13] https://epsg.io/
[14] https://www.ogc.org/standard/geotiff/

References

Alizadehashrafi, B. (2019). Introducing a Customized Framework for 3D Spatial Data Infrastructure of Iran Based on OGC Standards. *Earth Observation and Geomatics Engineering, 3(1), 92-101*.

Basanow, J., Neis, P., Neubauer, S., Schilling, A., Zipf, A. (2008). Towards 3D Spatial Data Infrastructures (3D-SDI) based on open standards - experiences, results and future issues//*Advances in 3D Geoinformation Systems*. Berlin: Springer, 2008: 65-86

Biljecki, F., Ledoux, H., & Stoter, J. (2016). An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, *59*, 25–37. https://doi.org/10.1016/J.COMPENVURBSYS.2016.04.005

Biljecki, F., Lim, J., Crawford, J., Moraru, D., Tauscher, H., Konde, A., Adouane, K., Lawrence, S., Janssen, P., & Stouffs, R. (2021). Extending CityGML for IFC-sourced 3D city models. *Automation in Construction*, *121*, 103440. https://doi.org/10.1016/J.AUTCON.2020.103440

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., & Çöltekin, A. (2015). Applications of 3D City Models: State of the Art Review. *ISPRS International Journal of Geo-Information*, *4*(4), 2842–2889. https://doi.org/10.3390/IJGI4042842

Boeters, R., Arroyo Ohori, K., Biljecki, F., & Zlatanova, S. (2015). Automatically enhancing CityGML LOD2 models with a corresponding indoor geometry. *International Journal of Geographical Information Science*, *29*(12), 2248–2268. https://doi.org/10.1080/13658816.2015.1072201

Budhathoki, N. R., Bruce, B., & Nedovic-Budic, Z. (2008). Reconceptualizing the role of the user of spatial data infrastructure. *GeoJournal*, *72*(3–4), 149–160. https://doi.org/10.1007/S10708-008-9189-X

Chadzynski, A., Krdzavac, N., Farazi, F., Lim, M. Q., Li, S., Grisiute, A., Herthogs, P., von Richthofen, A., Cairns, S., & Kraft, M. (2021). Semantic 3D City Database — An enabler for a dynamic geospatial knowledge graph. *Energy and AI*, *6*, 100106. https://doi.org/10.1016/J.EGYAI.2021.100106

Chaturvedi, K., Yao, Z., & Kolbe, T. H. (2019). Integrated Management and Visualization of Static and Dynamic Properties of Semantic 3D City Models. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII-4-W17*(4/W17), 7–14. https://doi.org/10.5194/ISPRS-ARCHIVES-XLII-4-W17-7-2019

Gao, F., Yue, P., Cao, Z., Zhao, S., Shangguan, B., Jiang, L., Hu, L., Fang, Z., & Liang, Z. (2022). A multi-source spatio-temporal data cube for large-scale geospatial analysis. *International Journal of Geographical Information Science*, *36*(9), 1853–1884. https://doi.org/10.1080/13658816.2022.2087222

Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., & Moore, R. (2017). Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment*, *202*, 18–27. https://doi.org/10.1016/J.RSE.2017.06.031

Gröger, G., & Plümer, L. (2012). CityGML – Interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, *71*, 12–33. https://doi.org/10.1016/J.ISPRSJPRS.2012.04.004

Guo, H., Li, X., Wang, W., Lv, Z., Wu, C., & Xu, W. (2016). An event-driven dynamic updating method for 3D geo-databases. *Geo-Spatial Information Science*, *19*(2), 140–147. https://doi.org/10.1080/10095020.2016.1182808

Hendriks, P. H. J., Dessers, E., & van Hootegem, G. (2012). Reconsidering the definition of a spatial data infrastructure. *International Journal of Geographical Information Science*, *26*(8), 1479–1494. https://doi.org/10.1080/13658816.2011.639301

Izdebski, W., Zwirowicz-Rutkowska, A., & Nowak da Costa, J. (2021). Open data in spatial data infrastructure: the practices and experiences of Poland. *International Journal of Digital Earth*, *14*(11), 1547–1560. https://doi.org/10.1080/17538947.2021.1952323

Jaillot, V., Servigne, S., & Gesquière, G. (2020). Delivering time-evolving 3D city models for web visualization. *International Journal of Geographical Information Science*, *34*(10), 2030–2052. https://doi.org/10.1080/13658816.2020.1749637

Jaillot, V., Rigolle, V., Servigne, S., Samuel, J., & Gesquière, G. (2021). Integrating multimedia documents and time‐evolving 3D city models for web visualization and navigation. *Transactions in GIS, 25(3)*, 1419-1438.

Karim, H., Rahman, A. A., Azri, S., & Halim, Z. (2022). The development of multi-scale data management for citygml-based 3d buildings. *Geomatics and Environmental Engineering*, *16*(1), 71–94. https://doi.org/10.7494/GEOM.2022.16.1.71

Karnatak, H. C., & Kumar, V. (2014). Performance study of various spatial indexes on 3D geo-data in Geo-RDBMS. *Geocarto International*, *30*(6), 633–649. https://doi.org/10.1080/10106049.2014.952354

Kutzner, T., Chaturvedi, K., & Kolbe, T. H. (2020). CityGML 3.0: New Functions Open Up New Applications. *PFG - Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, *88*(1), 43–61. https://doi.org/10.1007/S41064-020-00095-Z/FIGURES/19

Labetski, A., Vitalis, S., Biljecki, F., Arroyo Ohori, K., & Stoter, J. (2023). 3D building metrics for urban morphology. *International Journal of Geographical Information Science*, *37*(1), 36–67. https://doi.org/10.1080/13658816.2022.2103818

Lai, X., Liu, J., Jiang, L., Wang, L., Zhao, H., Liu, S., Qi, X., & Jia, J. (2022). Stratified Transformer for 3D Point Cloud Segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 8490–8499. https://doi.org/10.1109/CVPR52688.2022.00831

Liang, J., Shen, S., Gong, J., Liu, J., & Zhang, J. (2017). Embedding user-generated content into oblique airborne photogrammetry-based 3D city model. *International Journal of Geographical Information Science*, 31(1), 1-16.

Lin, H., Chen, M., Lu, G., Zhu, Q., Gong, J., You, X., Wen, Y., Xu, B., & Hu, M.,Virtual Geographic Environments (VGEs): A New Generation of Geographic Analysis Tool, *Earth-Science Reviews*, 126, 74-84.

Liu, J., Liu, Y., & Li, H. (2016). Application of BIM and GIS Based Data Integration in Water Conservancy and Hydropower Engineering. *Journal of Engineering Management*, 30(4): 95–99. doi:10.13991/j.cnki.jem.2016.04.018.

Ledoux, H., Arroyo Ohori, K., Kumar, K., Dukai, B., Labetski, A., & Vitalis, S. (2019). CityJSON: a compact and easy-to-use encoding of the CityGML data model. *Open Geospatial Data, Software and Standards*, *4*(1). https://doi.org/10.1186/S40965-019-0064-0

Lewis, A., Oliver, S., Lymburner, L., Evans, B., Wyborn, L., Mueller, N., Raevksi, G., Hooke, J., Woodcock, R., Sixsmith, J., Wu, W., Tan, P., Li, F., Killough, B., Minchin, S., Roberts, D., Ayers, D., Bala, B., Dwyer, J., … Wang, L. W. (2017). The Australian Geoscience Data Cube — Foundations and lessons learned. *Remote Sensing of Environment*, *202*, 276–292. https://doi.org/10.1016/J.RSE.2017.03.015

Luan, H., Fan, Y., Zhou, M., & Wang, X. (2014). Towards effective 3D model management on hadoop. In *Advances in Computer Science and its Applications*, *279*: 131–139. https://doi.org/10.1007/978-3-642-41674-3_20/COVER

Mao, B., Harrie, L., Cao, J., Wu, Z., & Shen, J. (2014). Nosql based 3D city model management system. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *40*(4), 169–173. https://doi.org/10.5194/ISPRSARCHIVES-XL-4-169-2014

Owers, C. J., Lucas, R. M., Clewley, D., Tissott, B., Chua, S. M. T., Hunt, G., Mueller, N., Planque, C., Punalekar, S. M., Bunting, P., Tan, P., & Metternicht, G. (2022). Operational continental-scale land cover mapping of Australia using the Open Data Cube. *International Journal of Digital Earth*, *15*(1), 1715–1737. https://doi.org/10.1080/17538947.2022.2130461

Saeedi, S., Liang, S., Graham, D., Lokuta, M. F., & Mostafavi, M. A. (2017). Overview of the OGC CDB Standard for 3D Synthetic Environment Modeling and Simulation. *ISPRS International Journal of Geo-Information*, *6*(10), 306. https://doi.org/10.3390/IJGI6100306

Schilling, A., Bolling, J., & Nagel, C. (2016). Using glTF for streaming CityGML 3D city models//*In Proceedings of the 21st International Conference on Web3D Technology*, 109-116.

Liang, J., Shen, S., Gong, J., Liu, J., & Zhang, J. (2017). Embedding user-generated content into oblique airborne photogrammetry-based 3D city model. International Journal of Geographical Information Science, 31(1), 1-16.

Stadler, A., Nagel, C., König, G., & Kolbe, T. H. (2009). Making interoperability persistent: A 3D geo database based on CityGML. In *3D Geo-Information Sciences*, Kluwer Academic Publishers: 175–192. https://doi.org/10.1007/978-3-540-87395-2_11/COVER

Stoter, J., Ohori, G. A. K. A., Dukai, B., Labetski, A., Kavisha, K., Vitalis, S., & Ledoux, H. (2020). State of the art in 3D city modelling: Six challenges facing 3D data as a platform. *GIM International: The Worldwide Magazine for Geomatics*, *34*.

Stoter, J. Vosselman, G., Goos, E. J., Zlatanova, S., Verbree, E., Klodster, R., Reuvers, M. (2011). Towards a National 3D Spatial Data Infrastructure: Case of The Netherlands. *Photogrammetrie Fernerkundung Geoinformation (PFG)*, 6, 405-420.

Wang, X., & Xie, M. (2022). Integration of 3DGIS and BIM and its application in visual detection of concealed facilities. *Geo-Spatial Information Science*, *27*(1), 132–141. https://doi.org/10.1080/10095020.2022.2054732.

Yao, Z., Nagel, C., Kunde, F., Hudra, G., Willkomm, P., Donaubauer, A., Adolphi, T., & Kolbe, T. H. (2018). 3DCityDB - a 3D geodatabase solution for the management,

analysis, and visualization of semantic 3D city models based on CityGML. *Open Geospatial Data, Software and Standards 2018 3:1*, *3*(1), 1–26. https://doi.org/10.1186/S40965-018-0046-7

Yu, D., Tang, L., Ye, F., & Chen, C. (2021). A virtual geographic environment for dynamic simulation and analysis of tailings dam failure. *International Journal of Digital Earth*, *14*(9), 1194–1212. https://doi.org/10.1080/17538947.2021.1945151

Yu, D., Yue, P., Ye, F., Tapete, D., & Liang, Z. (2023). Bidirectionally greedy framework for unsupervised 3D building extraction from airborne-based 3D meshes. *Automation in Construction*, *152*, 104917. https://doi.org/10.1016/J.AUTCON.2023.104917

Yuan, W., Gu, X., Dai, Z., Zhu, S., & Tan, P. (2022). Neural Window Fully-connected CRFs for Monocular Depth Estimation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 3906–3915. https://doi.org/10.1109/CVPR52688.2022.00389

Yue, P., Guo, X., Zhang, M., Jiang, L., & Zhai, X. (2016). Linked Data and SDI: The case on Web geoprocessing workflows. *ISPRS Journal of Photogrammetry and Remote Sensing*, *114*, 245–257. https://doi.org/10.1016/J.ISPRSJPRS.2015.11.009

Yue, P., Zhang, C., Zhang, M., Zhai, X., & Jiang, L. (2015). An SDI Approach for Big Data Analytics: The Case on Sensor Web Event Detection and Geoprocessing Workflow. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *8*(10), 4720–4728. https://doi.org/10.1109/JSTARS.2015.2494610

Zhang, J., Zhu, J., Zhou, Y., Zhu, Q., Wu, J., Guo, Y., … Zhang, H. (2024). Exploring geospatial digital twins: a novel panorama-based method with enhanced representation of virtual geographic scenes in Virtual Reality (VR). *International Journal of Geographical Information Science*, 1–24. https://doi.org/10.1080/13658816.2024.2386064

Zhu, J., Zhang, J., Zhu, Q., Zuo, L., Liang, C., Chen, X., & Xie, Y. (2024). Virtual geographical scene twin modeling: a combined data-driven and knowledge-driven method with bridge construction as a case study. *International Journal of Digital Earth*, *17*(1), 1–23. https://doi.org/10.1080/17538947.2024.2356126