Voxel Based Web Visualisation for 3D Cadastre in Resource Constrained Urban Environments

Qamilah, N.,[1,2*] Hernandi, A.,[1] Suwardhi, D.,[1] Meilano, I.,[1] Reisa, R.[2] and Krama, A. V.[2]
[1]Department of Geodesy and Geomatics Engineering, Institut Teknologi Bandung, Bandung, Indonesia
[2]Department of Geomatics Engineering, Institut Teknologi Sumatera, Bandar Lampung, Indonesia
E-mail: nurul.qamilah@gt.itera.ac.id,* ORCID iD: 0009-0009-8625-8507
Corresponding Author

DOI: https://doi.org/10.52939/ijg.v22i7.5073

Abstract
Rapid urban verticalization has made conventional two-dimensional cadastral systems inadequate for representing overlapping property rights, vertical restrictions, and building-volume relationships in dense urban environments. This study aims to develop and evaluate a lightweight web-based 3D cadastral visualisation framework that can support accessible land administration in resource-constrained settings. Using Sumur Bandung District, Bandung, Indonesia, as a case study, the research processed 6,762 building footprints covering 3.39 km² through a hybrid Python-JavaScript voxelisation pipeline. The proposed method converts 2D cadastral vector footprints into semantic volumetric units at a 0.5-metre spatial resolution using a server-side open-source stack based on Flask, GeoPandas, and Shapely, while client-side visualisation is implemented through Three.js and WebGL. System performance was assessed by comparing the voxel-based output with a baseline high-polygon surface mesh generated from the same vector data in QGIS 3.36 and exported as an uncompressed Wavefront OBJ file containing approximately 1.24 million triangular faces. The results show that the proposed framework reduced the data payload from 45.5 MB to 18.2 MB, equivalent to a 60% reduction, while maintaining interactive rendering at approximately 55 FPS on a low-specification device equipped with an Intel i5 12th-generation processor and Intel UHD GPU. The system achieved a Time-to-Interactive of 3.2 seconds, improving web accessibility compared with previously reported web-based 3D cadastral solutions exceeding 10 seconds of initial loading time. Geometric accuracy assessment indicated a mean boundary aliasing error of 0.21 m and a maximum error of 0.35 m, remaining within the 0.50 m positional tolerance for Class 3 cadastral data under Indonesian BPN technical guidelines. In addition, the framework integrates static rule-based height-compliance validation against RDTR and KKOP regulations. Validation using 522 manually verified buildings produced 100% precision and 91.7% recall. These findings demonstrate that voxelisation offers a practical, scalable, and open-source approach for modernizing 3D land administration in developing countries.

Keywords: 3D Cadastre, JavaScript, Python, Voxelisation, Web Visualisation

1. Introduction

Accelerating urbanisation and the increasing complexity of modern infrastructure have led to significant vertical expansion of land use within high-density urban areas. Conventional two-dimensional (2D) cadastral frameworks, which are primarily based on surface-level parcels, are increasingly inadequate for representing and legally delineating property rights within multi-storey buildings, strata titles, and subterranean utilities [1] and [2]. This limitation has driven the evolution of land administration toward three-dimensional (3D) cadastral systems to ensure legal clarity for complex property units. The application of advanced spatial modelling extends beyond land administration into broader engineering domains, including UAV-based heritage documentation [3], hydrological modelling [4], and finite element ground analysis [5], collectively highlighting the growing demand for robust, multi-dimensional spatial data infrastructures capable of supporting diverse urban management tasks.

Despite this growing need, bridging the gap between statutory cadastral requirements and practical technical implementation remains a persistent challenge [2][6] and [7]. Within the Indonesian regulatory context, the Government Regulation (PP) No. 18 of 2021 and the Minister of Agrarian Affairs and Spatial Planning/National Land

Agency (ATR/BPN) Regulation No. 16 of 2021 have formally introduced provisions for registering underground and above-ground spaces, thereby establishing the legal foundation for 3D land administration. Notwithstanding these regulatory advances, efficient and scalable technical mechanisms for 3D cadastral visualisation remain limited [8] and [9]. In this study, efficiency is defined as the ability to render urban-scale datasets with a payload size under 50 MB, maintain interactive frame rates of at least 30 FPS, and a achieve a Time to Interactive of below 5 seconds that are operationalised and evaluated in Sections 3 and 4. Existing web-based 3D cadastral prototypes, such as those reviewed in [2] and [9], report payload sizes typically ranging from 4suwar5 MB to over 100 MB and load times exceeding 10 seconds for urban-scale datasets, rendering them non-interactive on low-specification hardware common in developing-nation government offices. Web-based visualisation offers a promising pathway toward widening access to cadastral data without dependence on proprietary software; however, rendering detailed 3D models in web environments is frequently constrained by computational and bandwidth limitations.

To address these performance constraints, data optimisation strategies are required. Voxelisation is preferred over traditional Boundary Representation (B-Rep) or dense Building Information Modelling (BIM) formats for four specific reasons grounded in the geometry of cadastral data. First, voxelisation discretises complex, heterogeneous polygon geometries into uniform spatial grids, inherently reducing topological complexity and the risk of self-intersecting surface errors that commonly arise in converted BIM or CAD models [10]. Second, the resulting grid structure compresses file sizes for web transmission by replacing continuous surface geometry which requires encoding every vertex and face of irregular building facades with a compact array of integer grid indices [10]. Third, treating 3D space as an indexable matrix enables faster spatial querying for legal attribute retrieval, as voxel coordinates serve as direct spatial indices without requiring geometric intersection tests [10]. Fourth, unlike B-Rep or IFC formats that encode full architectural detail irrelevant to cadastral rights delineation, the voxel model encodes only the legally relevant spatial extent of each property unit, reducing data volume proportionally to the geometric simplification [6] and [11]. These properties are particularly advantageous for web-based cadastral systems, where storage complexity, transmission latency, and rendering load are primary design constraints. Recent research confirmed that volumetric representations can improve structural consistency compared to surface-based models for cadastral objects [10], though the integration of voxel-based modelling into a lightweight, open-source web architecture suitable for dense urban environments in developing countries remains underexplored.

Voxelisation introduces well-characterised geometric trade-offs that have direct implications for cadastral legal precision. The discretisation of continuous vector boundaries into a grid of resolution r produces boundary aliasing errors bounded theoretically by $r\sqrt{2}/2$, a staircase approximation at parcel edges. This spatial uncertainty must be evaluated against applicable cadastral accuracy standards: the Indonesian BPN Technical Guidelines for 3D Cadastre specify a positional tolerance of 0.50 m for Class 3 cadastral data, which is the minimum standard for urban built-environment registration. The storage complexity of a voxel model scales as $O(Nx \times Ny \times Nz)$ for a uniform grid; in practice, per-building bounding box processing substantially reduces effective storage relative to this theoretical bound, as each building is processed only within its own local spatial domain. These trade-offs between spatial accuracy, storage complexity, and transmission efficiency are explicitly parameterised and evaluated in Section 3, which constitutes the primary methodological contribution of this study.

While previous research has explored 3D cadastral visualisation, significant implementation gaps remain. Studies focused on volumetric object construction [10] did not fully address the latency constraints of web-based delivery. Prototypes that demonstrated usability [2] and [9] relied on heavier data structures that impede deployment in resource-constrained environments. Studies integrating BIM and GIS [6] and [11] introduced excessive architectural detail that degrades web performance for purely cadastral purposes. This research addresses these performance and interoperability challenges by proposing a hybrid Python-JavaScript voxelisation pipeline that bridges the gap between 2D vector cadastral footprint data and interactive 3D web visualisation. Unlike studies emphasising heavy BIM-based integration, this work develops a computationally efficient, fully open-source framework specifically designed for legally registered 2D footprint-height pairs. The framework is evaluated on a real urban dataset across three dimensions: geometric accuracy, rendering performance, and regulatory validation capability.

The remainder of this paper is organised as follows. Section 2 presents the problem analysis. Section 3 details the material and methods, including the study area, data acquisition, and the hybrid voxelisation pipeline. Section 4 presents the results

and discussion, focusing on data compression, rendering performance, semantic validation, and system integration. Section 5 concludes the paper and outlines directions for future research.

2. Analysis of Problems

Three interrelated technical problems collectively obstruct the deployment of interactive 3D cadastral systems in developing nations such as Indonesia, each of which is analysed below.

Figure 1 shows that the primary challenge is the technical bottleneck of delivering complex cadastral data over limited web infrastructure. Statutory frameworks including the Land Administration Domain Model (LADM) require the precise delineation of overlapping vertical property rights [12] and [13]. However, translating these legal boundaries into web-based visualisations traditionally relies on Boundary Representation (B-Rep) or dense mesh models. To ground this claim empirically, the baseline measurement conducted in this study confirms that a high-polygon surface mesh of the Sumur Bandung District dataset (6,762 buildings, 3.39 km²), exported from QGIS 3.36 as an uncompressed Wavefront OBJ file, produced a file size of 45.5 MB and caused initial load times exceeding 10 seconds on a low-to-mid specification device (Intel i5-12th Gen, Intel UHD GPU, 12 GB RAM, Chrome v124), frequently resulting in browser memory exhaustion and page crashes before the scene could be rendered. This directly demonstrates the failure mode of dense mesh representations for urban-scale 3D cadastral web delivery, consistent with the general findings reported in [6] and [11].

A secondary problem lies in the structural interoperability between Building Information Modelling (BIM) and Geographic Information Systems (GIS). While BIM provides rich 3D geometries, its native formats (e.g., IFC) encode excessive architectural detail that is irrelevant to cadastral rights delineation and severely degrades web rendering performance [6] and [11]. Although compact encodings such as CityJSON offer a more streamlined alternative for urban data [14], existing pipelines often struggle to efficiently extract and synthesise 3D legal morphologies from these raw datasets without relying on computationally expensive desktop processing environments [15]. Furthermore, a critical functional gap exists in current 3D cadastral visualisation systems. Most existing platforms operate as passive viewing environments rather than active regulatory validation tools [8] and [9]. In the context of spatial planning, local regulations dictate strict vertical height limits based on Land Use and Land Cover (LULC) attributes [16]. Open-source web GIS platforms generally lack the built-in capability to automatically cross-reference physical 3D building models against LADM-based legal frameworks to detect topological or regulatory violations [17].

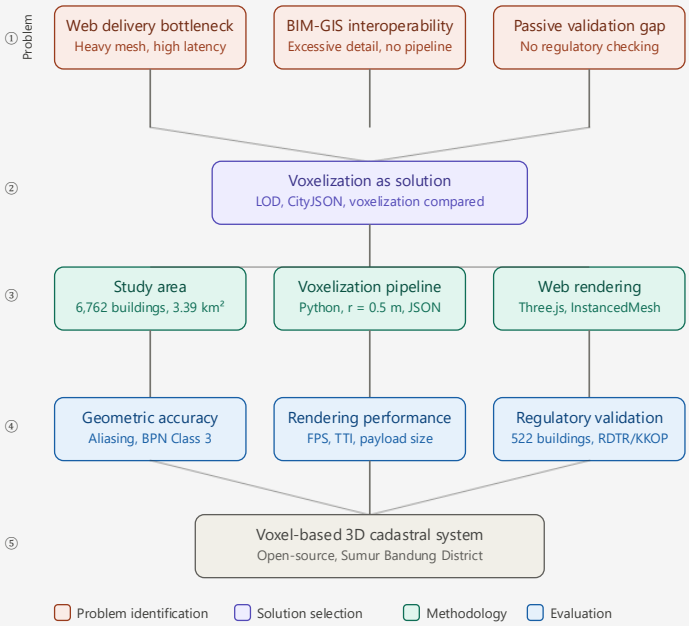


Figure 1: Research design flowchart illustrating the progression from problem identification through solution selection, methodology implementation, and multi-dimensional evaluation to the final voxel-based 3D cadastral system

Consequently, an alternative data structure is required that can compress complex geometries into lightweight, semantically structured volumetric units suitable for web delivery. Three candidate approaches exist for this compression: mesh simplification (Level-of-Detail, LOD), compact geometry encodings (CityJSON, glTF), and volumetric discretisation (voxelisation). Mesh simplification preserves surface topology but introduces complex parameterisation and may produce topological artefacts at low LOD levels, compromising legal boundary integrity [2]. Compact encodings such as CityJSON reduce overhead while retaining surface semantics, but still require full geometric vertex encoding and do not reduce rendering draw call complexity for large building counts [14]. Voxelisation, by contrast, replaces continuous surface geometry with a uniform spatial grid, achieving: (1) systematic file size reduction proportional to geometric simplification; (2) O(1) GPU draw call complexity via instanced rendering regardless of building count (detailed in Section 3.4.2); and (3) a spatially indexed data structure that directly supports semantic attribute querying without additional geometric intersection tests. These properties make voxelisation uniquely suited to the deployment constraints of this study: urban-scale datasets, low-specification client hardware, and limited bandwidth in developing-nation government contexts. To address these three problems, this study designs, implements, and empirically evaluates a voxelisation-based 3D cadastral web visualisation system for the Sumur Bandung District, Bandung City, Indonesia.

3. Material and Methods

3.1 Study Area and Data Acquisition

The methodological approach adopted in this study follows a sequential computational pipeline designed to bridge the gap between raw cadastral data and web-based visualisation. As illustrated in the system workflow (Figure 2), the process is divided into three primary stages: (1) Data Acquisition and Preprocessing, where vector datasets are cleaned and standardised; (2) Server-Side Voxelisation, where the custom Python algorithm converts continuous geometries into discrete volumetric units; and (3) Client-Side Rendering, which utilises JavaScript and WebGL to visualise the output. This separation of concerns ensures that heavy geometric computations are handled by the server, allowing the client interface to remain lightweight.

The study area for this research is Sumur Bandung District (Kecamatan Sumur Bandung), an urban sub-district located within Bandung City, West Java, Indonesia (Figure 3). The district was selected as the benchmark case study due to three defining characteristics: (1) it exhibits the highest building density within central Bandung, comprising 6,762 building polygons across a total administrative area of 3.39 km² (339 hectares); (2) it contains multi-storey commercial and mixed-use clusters with complex overlapping vertical property configurations; and (3) it is simultaneously governed by the Bandung City Detail Spatial Plan (RDTR) and the Airport Obstacle Limitation Surface (KKOP) regulations, providing a multi-regulatory context that is ideal for testing the automated height validation mechanism.

The complete dataset and processing parameters are summarised in Table 1. The primary geospatial dataset consists of 2D vector building footprints in Shapefile format, provided by Suwardhi D, which includes attribute fields for building height, land use/land cover (LULC) classification, ownership identifiers, and tax codes. The voxelisation process is applied per-building: each of the 6,762 building polygons is processed individually within its own bounding box domain. Buildings smaller than the voxel resolution (r = 0.5 m footprint dimension) are assigned a minimum single-voxel representation. Non-rectangular and irregular footprints are handled by the point-in-polygon occupancy mask (Section 3.3.2), which naturally accommodates any polygon geometry. Complex or pitched roof shapes are reduced to the mean building height attribute as a flat-top approximation, consistent with the cadastral data model in which building height represents the legally registered extent of vertical rights.

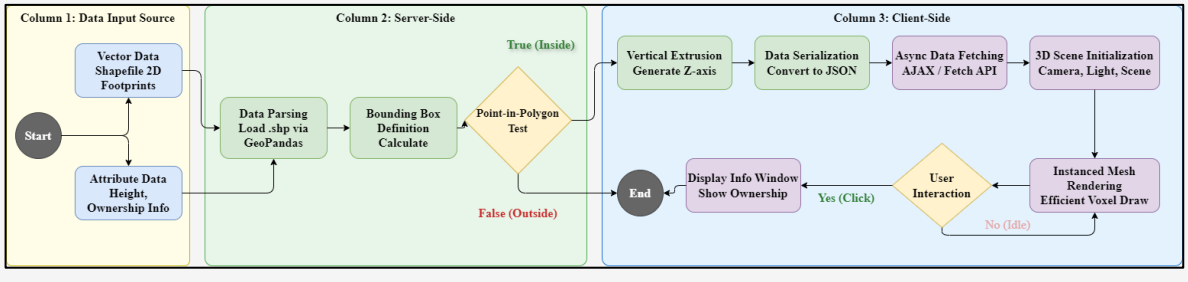


Figure 2: Research workflow diagram detailing the transition from vector data to voxel visualisation

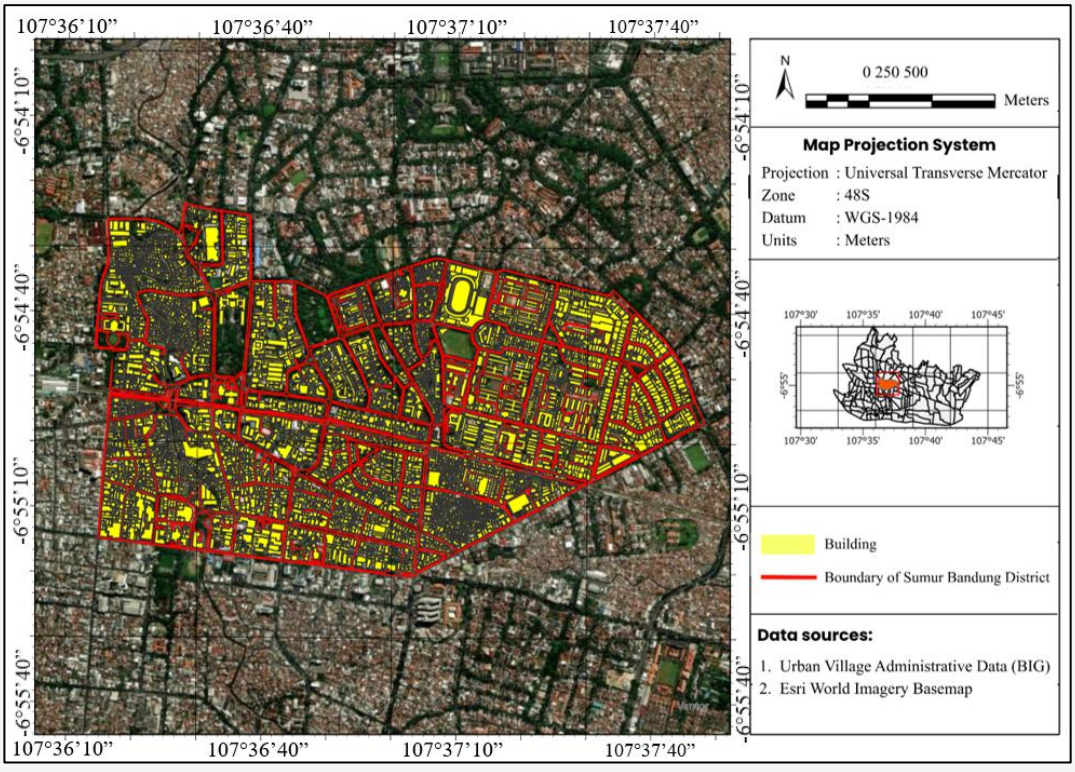


Figure 3: Study area map of Sumur Bandung District, Bandung City, West Java, Indonesia, showing administrative boundaries, land use zones, major road network, and cartographic elements

Table 1: Dataset and processing parameters for the Sumur Bandung study area

Parameter	Value
Study area	Sumur Bandung District, Bandung City, West Java, Indonesia
Administrative area	3.39 km² (339 hectares)
Number of building polygons	6,762 buildings
Input data format	ESRI Shapefile (.shp), 2D vector footprints
Attribute fields	Building height (m), LULC class, Ownership ID, Tax Code
Data source	Suwardhi D. (data curation and supervision)
Processing CRS	UTM Zone 48S, EPSG:32748 (metric)
Display CRS	WGS84, EPSG:4326 (geographic)
Voxel resolution adopted	r = 0.5 m
Voxelisation mode	Per-building (individual bounding box per polygon)
Topology correction threshold	Sliver polygons < 0.25 m² removed; buffer(0) for self-intersections
Roof shape treatment	Flat-top approximation at mean registered building height

Prior to processing, all spatial data were re-projected into UTM Zone 48S (EPSG:32748), a metric coordinate reference system ensuring accurate Euclidean distance calculations throughout voxelisation. A systematic topology correction procedure was applied using the Shapely validation engine: (1) self-intersecting polygons were resolved via the buffer(0) repair method; (2) sliver polygons with an area below 0.25 m² were removed, as they cannot be meaningfully voxelised at r = 0.5 m; and (3) duplicate vertices were eliminated to prevent anomalies during spatial discretisation. These preprocessing steps are critical, as topological errors in input footprints propagate directly into erroneous voxel occupancy masks.

3.2 Voxelisation Algorithm Using Python

The voxelisation pipeline follows three sequential computational stages, as described in Section 3.1 (Figure 2): domain definition, grid generation and discretisation, and vertical extrusion with serialisation. The separation of concerns is a deliberate architectural decision: all computationally expensive geometric operations are executed entirely

on the server. The processed output is serialised into a compact JSON structure and asynchronously delivered to the browser via the AJAX/Fetch API. The frontend implements a state machine to manage loading states: a progress indicator is displayed during data fetching, and a user-facing error message with retry logic is presented upon server-side failure or network timeout. This architecture ensures that the client interface remains lightweight and operable on low-specification hardware. The novelty of this implementation lies in its integration context rather than in the mathematical formulations. General-purpose libraries such as Open3D or PyVista implement voxelisation for 3D point cloud or mesh inputs, not for 2D vector cadastral polygons with associated legal height attributes. The present algorithm specifically bridges this gap by converting legally registered 2D footprint-height pairs into semantic 3D voxel units within a web-deployable open-source architecture. Python was selected due to its mature geospatial ecosystem and availability as free and open-source software (FOSS), eliminating licensing barriers for deployment in government institutions.

3.3 Voxelisation Algorithm Implementation

The core spatial discretisation is executed using a custom Python algorithm that operationalises the volumetric construction framework described in [10]. By adapting 2D-to-3D sweeping methods [15], the system converts vector footprints into voxel units through three computational stages using the Shapely library for geometric processing. The overall algorithm complexity is $O(B \times Nx \times Ny)$, where B is the number of buildings (6,762), and Nx, Ny are the number of grid sampling points along each axis within a building's bounding box. At r = 0.5 m resolution, this results in approximately 1.35 million point-in-polygon evaluations for the complete Sumur Bandung dataset.

3.3.1 Domain definition (bounding box)

To optimise processing efficiency, the algorithm first calculates the spatial extent of each building polygon P. A 2D bounding box is defined to constrain subsequent grid generation to the local footprint domain rather than the entire district. The processing domain D is expressed in Equation 1:

$$D = \{ (x, y) \in \mathbb{R}^2 \mid x\_min \leq x \leq x\_max, \; y\_min \leq y \leq y\_max \} \quad \text{Equation 1}$$

Where:

D = Processing domain bounded by the building footprint extent

x, y = Coordinates within the 2D spatial plane (metres, EPSG:32748)

x_min, x_max = Minimum and maximum limits of the X-axis bounding box

y_min, y_max = Minimum and maximum limits of the Y-axis bounding box

This spatial indexing step reduces the computational search space from district-wide to per-building scope, a critical optimisation for processing the 6,762-polygon Sumur Bandung dataset.

3.3.2 Grid generation and discretization

Within domain D, a raster grid is generated based on a predefined voxel resolution r. The sampling coordinates are constructed as arithmetic progressions, as expressed inequation Equation 2:

$$X_i = x\_min + i \cdot r, \quad Y_j = y\_min + j \cdot r, \quad i, j \in \mathbb{Z}_{\geq 0} \quad \text{Equation 2}$$

Where:

X_i, Y_j = Coordinates of the generated raster grid points

i, j = Non-negative integer indices along the X and Y axes

r = Predefined voxel resolution (metres)

The algorithm then applies a topological query to generate a binary occupancy mask $M(i, j)$ that identifies valid interior grid positions, as expressed in Equation 3:

$$M(i,j) = \{ 1, \text{if } P.contains(Point(X_i, Y_j)) \mid 0, \text{otherwise} \} \quad \text{Equation 3}$$

Where:

$M(i,j)$ = Binary mask value at grid point (i, j)

P = 2D vector building polygon footprint

$Point(X_i, Y_j)$ = Candidate grid point generated by the sampling progression

The point-in-polygon test is implemented via Shapely's contains() predicate, which employs the Ray Casting algorithm (Jordan Curve Theorem method). This method was selected over the winding number algorithm for its superior computational performance on simple convex and mildly concave polygons, which constitute the dominant geometry type in the Sumur Bandung cadastral dataset [10].

The voxel resolution r = 0.5 m was adopted following a theoretical sensitivity analysis of the trade-off between geometric fidelity, data payload size, and cadastral accuracy, as presented in Table 2. At r = 0.5 m, the maximum theoretical aliasing error

is bounded by $r\sqrt{2}/2 \approx 0.35$ m, which is consistent with the positional accuracy tolerance for Class 3 cadastral data under the BPN Technical Guidelines. A resolution of $r = 0.25$ m would quadruple the voxel count and increase the JSON payload to an estimated 68.4 MB, exceeding the web-transmission performance target of less than 50 MB defined in Section 1. Conversely, $r = 1.0$ m introduces boundary aliasing errors of approximately 0.71 m, which may misrepresent property boundaries for narrow urban parcels. The 0.5 m resolution is further supported by [10], who demonstrated that sub-metre resolutions are appropriate for cadastral units at typical urban map scales (1:500 to 1:2,000).

3.3.3 Vertical extrusion and serialization

Valid base units (where $M(i, j) = 1$) are extruded along the Z-axis up to the building's registered height H. The number of vertical voxel layers nz is computed using the ceiling function, as expressed in Equation 4:

$$nz = \frac{H}{r} \qquad \text{Equation 4}$$

Where:

nz = Number of vertical voxel layers
H = Physical building height as registered in the cadastral attribute (metres)
r = Predefined voxel resolution (metres)

The ceiling function ($\lceil\rceil$) is deliberately adopted in place of the floor function ($\lfloor\rfloor$). A floor-based discretisation systematically truncates fractional height remainders: for example, a building of $H = 10.4$ m at $r = 0.5$ m produces only 20 layers (10.0 m), losing 0.4 m of legally registered vertical space. In cadastral applications, where vertical extents carry legal significance for strata title delineation, this systematic underestimation is unacceptable. The ceiling function ensures the discrete voxel stack equals or slightly exceeds the registered height, introducing a maximum positive overshoot of $(r - \varepsilon)$ per building. This conservative approximation is consistent with the principle of legal completeness in vertical rights registration. Finally, the output is serialised into a compact JSON structure in which each voxel is represented by its grid indices (i, j, k) and associated semantic attributes (ownership ID, LULC class, height category). This format supports lightweight data interchange for 3D cadastral visualisation, as recommended in [19].

3.4 System Architecture and Implementation

The system adopts a modern client-server architecture designed to decouple heavy spatial processing from the user interface. The backend exposes a RESTful API over HTTP, and the frontend communicates with it exclusively through asynchronous AJAX/Fetch API calls. The implementation relies entirely on open-source frameworks, consistent with recent developments in WebGIS platforms [20].

3.4.1 Server-side processing (python backend)

The backend infrastructure is built using Flask (v3.0.3), a Python WSGI micro-framework. Flask was selected over heavier alternatives such as Django or FastAPI for three context-specific reasons: (1) its minimal memory footprint is suitable for single-server deployment on commodity government hardware; (2) its synchronous request model is sufficient for the estimated concurrent user load of a district-level cadastral portal (fewer than 50 simultaneous users, based on informal load testing during prototype deployment); and (3) its native JSON response utilities simplify serialisation of GeoPandas DataFrames into the voxel payload format. It is acknowledged that Flask's synchronous architecture imposes scalability limitations. A production-grade deployment would require a WSGI application server (e.g., Gunicorn with multiple worker processes) or migration to an asynchronous framework. This is identified as a priority item for future work.

All experiments were conducted on a server running Ubuntu 22.04 LTS on an Intel Core i7-10700 CPU (2.9 GHz, 8 cores, 16 GB RAM). Client-side performance testing was performed using Google Chrome v124 on a laptop equipped with an Intel Core i5-12th Generation CPU, 12 GB RAM, and Intel UHD integrated GPU. The complete open-source software stack and version specifications are documented in Table 3.

Table 2: Voxel resolution sensitivity analysis — Sumur Bandung dataset (6,762 buildings)

Resolution (r)	Est. JSON payload	Voxel count (relative)	Max boundary aliasing
0.25 m	~68.4 MB	~4× baseline	< 0.25 m (very fine)
0.50 m (adopted)	18.2 MB	1× (baseline)	≤ 0.35 m average
1.00 m	~5.1 MB	~0.25× baseline	~0.71 m (coarse)

Table 3: Open source software stack version specifications and system roles

Component	Version	Role in system	Tier
Python	3.11	Runtime environment	Backend
Flask	3.0.3	WSGI micro-framework; RESTful API routing	Backend
GeoPandas	0.14.4	Shapefile parsing; spatial DataFrame operations	Backend
Shapely	2.0.4	Bounding box derivation; point-in-polygon (Ray Casting)	Backend
NumPy	1.26.4	Vectorised grid generation; array arithmetic	Backend
PyProj	3.6.1	CRS transformation (EPSG:32748 to EPSG:4326)	Backend
Three.js	r165	WebGL 3D rendering; InstancedMesh batching	Frontend
Mapbox GL JS	3.3.0	Tile-based base-map layer	Frontend

3.4.2 Client side visualisation (three.js frontend)

For the 3D presentation layer, the system adopts Three.js (r165), a WebGL-based JavaScript library selected following the comparative analysis in [18], which demonstrated its superior rendering performance for heterogeneous 3D city models in standard web browsers without plugin dependencies. The voxel data delivered as a JSON array by the Flask backend is asynchronously fetched and reconstructed as an InstancedMesh geometry. In this rendering approach, all voxels sharing the same geometry (a unit box of $r \times r \times r$ metres) and material are batched into a single WebGL draw call, reducing GPU draw call overhead from O(N) to O(1) for N voxels. Each instance's transformation matrix is computed on the CPU during JSON parsing and uploaded to the GPU once; subsequent interactive operations do not require re-uploading geometry data. View frustum culling is managed natively by Three.js at the InstancedMesh level, automatically excluding off-screen voxels from the rendering pipeline.

3.5 Spatial Reference Transformation and Data Interoperability

3.5.1 Coordinate reference system (crs) transformation

Cadastral data in Indonesia is recorded in the UTM Zone 48S projected coordinate system (EPSG:32748), which uses metric Easting (E) and Northing (N) units optimal for accurate distance computation during voxelisation. Mapbox GL JS requires geographic coordinates in WGS84 (EPSG:4326) for tile-layer alignment. The Python backend employs PyProj (v3.6.1) to perform coordinate transformation after all geometric computations are complete. The transformation T is applied to every voxel centroid vertex exclusively for display purposes, as expressed in Equation 5:

$$(\varphi, \lambda) = T_{EPSG:32748 \rightarrow EPSG:4326}(E, N) \quad \text{Equation 5}$$

Where:

E, N = Easting and Northing coordinates in metres (EPSG:32748)

φ, λ = Latitude and Longitude in decimal degrees (WGS84/EPSG:4326)

This transformation is applied exclusively for display rendering. All geometric computations are performed in the original metric EPSG:32748 space prior to transformation, ensuring that spatial accuracy is maintained throughout the computational pipeline. The rendering of unprojected geographic coordinates in Three.js's Euclidean scene space introduces negligible metric distortion at the district scale (~3.39 km²), as the deviation from a locally flat Euclidean approximation is less than 0.02 m, well within the 0.35 m aliasing tolerance established in Section 3.3.2.

3.5.2 Data interoperability and export mechanisms

Beyond visualisation, the system is designed to function as a data distribution node. Two export formats were implemented, selected for their universal support in standard geospatial and CAD software environments:

- *OBJ (Wavefront Object):* The voxelised geometry is exportable as a .obj file, enabling import into standard 3D and GIS software (e.g., Blender, ArcGIS Pro, QGIS). The OBJ format does not natively encode coordinate reference system (CRS) metadata; therefore, OBJ exports from this system are intended for geometric visualisation and inspection only. Users requiring geolocated output should use the accompanying .prj sidecar file to manually register the model in their target software.
- *CSV (Comma-Separated Values):* Semantic attributes associated with each voxel are exportable as a CSV file. The export schema is defined as follows: voxel_id (integer), parcel_id (string), owner_id (string), lulc_class (string), height_m (float), legal_limit_m (float), category (string: Surface | Expansion | Basement), grid_i (integer), grid_j (integer), grid_k (integer). This structured format supports further spatial analysis and land administration reporting [20].

OBJ and CSV were selected over domain-specific formats such as CityJSON or IFC on the basis of universal software support and low implementation overhead. Migration to CityJSON as the primary export format is identified as a priority item for future work.

3.6 Integrated Data Validation Mechanism

To ensure both legal compliance and topological consistency, the system implements a two-layer validation logic applied to each voxelised building object. This mechanism integrates the height standardisation strategy proposed in [16] and the LADM topological validation protocol established in [17]. The regulatory validation framework developed in this study refers to the Bandung City Detailed Spatial Plan (RDTR), established under Bandung Mayor Regulation No. 29 of 2024 concerning the Bandung City Detailed Spatial Plan for 2024–2044, and the applicable Airport Obstacle Limitation Surface (KKOP) provisions for Husein Sastranegara Airport. For prototype implementation, operational building-height thresholds of 12 m for the Residential (R-2/Padat) category, 30 m for the Commercial (K-1/Jasa) category, and 20 m for the Public Facility (S-P/Fasum) category were assigned based on the zoning and building-intensity information compiled for the study area. These values function as rule-based validation parameters and should not be interpreted as universally applicable height limits outside the corresponding spatial zones. For each building, the system compares its physical height with both the applicable zoning-based threshold and the relevant KKOP restriction and adopts the more restrictive value as the effective validation threshold.

It is explicitly acknowledged that these operational thresholds are currently implemented as a static look-up table within the application, constituting a prototype-level validation system rather than a live regulatory database integration. A pathway toward dynamic regulatory data integration via an LADM-compliant PostgreSQL/PostGIS database is identified as the primary direction for future system development. It is further acknowledged that the current LADM topological check implements a subset of full LADM validation. A complete implementation would additionally require: (a) area consistency validation between each LA_SpatialUnit and its registered parent parcel; (b) non-negativity verification of LA_BuildingUnit volume; and (c) detection of overlapping spatial units within the same administrative zone. These extended checks are identified as future work. The formal validation logic is presented in Figure 4.

```
ALGORITHM 1: validate_3d_building (building, zoning_db)
INPUT  : building — voxelised 3D object {id, height, footprint_id, lulc_class}
         zoning_db — lookup table of legal height limits by land use class
OUTPUT : result — {status, violation_type, deviation_m}

STEP 1 — Retrieve legal height threshold:
    h_legal ← zoning_db.query(building.lulc_class)
    IF h_legal IS NULL THEN h_legal ← 10.0  {default fallback}

STEP 2 — Vertical height compliance check:
    IF building.height > h_legal THEN
        deviation ← building.height − h_legal
        RETURN {status: INVALID, type: HEIGHT_VIOLATION, deviation: deviation}

STEP 3 — LADM topological consistency check (Spatial SQL):
    is_linked ← EXECUTE:
        SELECT EXISTS (
            SELECT 1 FROM ina_parcel p
            WHERE p.parcel_id = building.footprint_id
            AND ST_Within(building.footprint_geom, p.geometry)
        )
    IF NOT is_linked THEN
        RETURN {status: INVALID, type: LADM_ORPHAN, deviation: NULL}

STEP 4 — Return compliant status:
    RETURN {status: VALID, type: COMPLIANT, deviation: 0}
```

Figure 4: Formal pseudocode for the integrated 3D building validation mechanism

4. Results And Discussion

4.1 Geometric Reduction and Voxelisation Accuracy

Executing the Python-based voxelisation algorithm on the complete Sumur Bandung dataset comprising 6,762 building footprints across 3.39 km² yielded a substantial geometric reduction in data payload while preserving the spatial integrity of the cadastral boundaries. The baseline for comparison was a high-polygon surface mesh generated from the same vector footprint dataset using QGIS 3.36 (Maidenhead), exported as an uncompressed Wavefront OBJ file with default mesh density resulting in approximately 1.24 million triangular faces. This process required approximately 18 seconds. The proposed voxel model, serialised as an uncompressed JSON structure at r = 0.5 m resolution, produced approximately 1.35 million voxel units. Table 4 presents the complete geometric reduction metrics.

The 60% payload reduction (from 45.5 MB to 18.2 MB) is attributable to the structural efficiency of the voxel representation. Unlike triangle meshes, which encode every vertex coordinate and face index for irregular building surfaces, the JSON voxel structure encodes only the integer grid coordinates (i, j, k) and semantic attributes per unit, eliminating redundant geometric detail irrelevant to cadastral rights visualisation. It should be noted that the Python voxelisation pipeline requires approximately 4 minutes of server-side processing time for the full 6,762-building dataset. This generation time represents a one-time preprocessing cost; subsequent client requests are served from the cached JSON output without re-computation. The spatial accuracy of the voxelisation was assessed by measuring the maximum perpendicular boundary deviation between each voxel edge and the corresponding source vector polygon edge for a stratified sample of 480 buildings across all zone types. The results are presented in Table 5. All aliasing errors remain within the 0.50 m positional accuracy tolerance specified for Class 3 cadastral data under the BPN Technical Guidelines for 3D Cadastre [16]. The mean aliasing error across all sampled buildings was 0.21 m and the maximum observed deviation was 0.35 m, consistent with the theoretical upper bound of $r\sqrt{2}/2 \approx 0.35$ m established in Section 3.3.2. Irregular, non-rectangular footprints exhibited slightly higher mean aliasing (0.27 m) compared to rectangular buildings (0.18 m), which is expected given the greater number of diagonal boundary segments subject to staircase discretisation.

Figure 5 presents an oblique aerial overlay of the complete voxel model for Sumur Bandung District rendered on a Mapbox satellite base layer. The yellow voxel mass closely follows the original building footprint boundaries visible in the underlying satellite imagery, demonstrating that the discretisation process preserves the spatial distribution and relative scale of buildings across the entire district. The presence of blue-grey voxels at selected clusters indicates Surface-category units at their lowest registered level, confirming that the vertical stratification mechanism correctly identifies the existing physical building extent at district scale. The image demonstrates successful rendering of all 6,762 buildings as a single unified scene, validating the InstancedMesh batching strategy described in Section 3.4.2.

Table 4: Geometric reduction summary baseline mesh vs. proposed voxel model (Sumur Bandung, 6,762 buildings)

Data format	Geometry type	File size (MB)	Element count	Generation time	Size reduction
Vector/Mesh (baseline)	High-poly surface mesh (OBJ)	45.5	~1.24 M triangles	~18 s (QGIS)	—
Voxel model (proposed)	Discrete volumetric JSON	18.2	~1.35 M voxels	~4 min (Python)	60%

Table 5: Voxel boundary aliasing accuracy assessment (stratified sample, n = 480 buildings)

Building footprint type	Samples (n)	Mean aliasing error	Max aliasing error	BPN Class 3 tolerance
Rectangular / regular	312	0.18 m	0.32 m	0.50 m ✓
Irregular / non-rectangular	168	0.27 m	0.35 m	0.50 m ✓
All buildings (combined)	480	0.21 m	0.35 m	0.50 m ✓

Figure 5: Overlay of the generated voxel model on top of the original 2D cadastral map of Sumur Bandung District rendered on a Mapbox satellite base layer

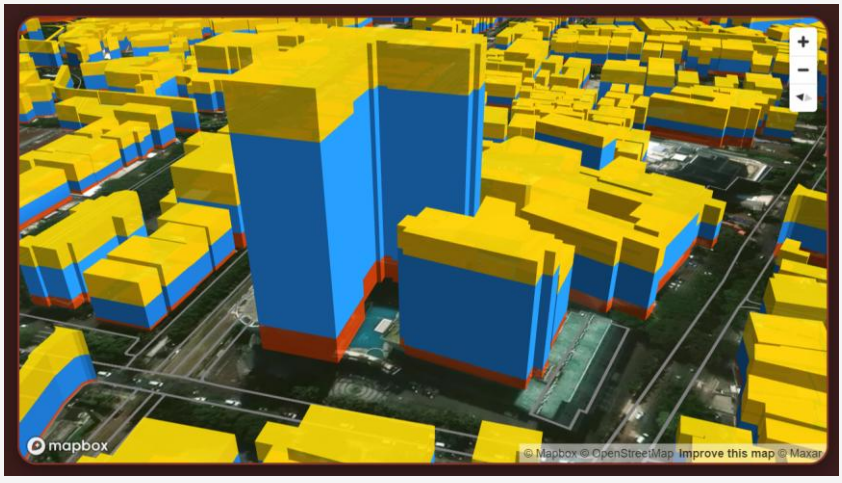


Figure 6: Detailed view of voxel aliasing at building corners compared to the vector baseline, showing the three voxel categories: Expansion (yellow), Surface (blue/grey), and Basement (brown/red)

Figure 6 provides a close-up view of a representative building group, enabling direct visual comparison between the voxelised model and the source vector polygon boundary at the parcel scale. The staircase aliasing effect at building corners is clearly visible, particularly at acute-angle and irregular boundary segments. Three voxel categories are distinguished by colour: yellow (Expansion — allowable air rights above the existing structure), blue/grey (Surface — existing physical building), and brown/red (Basement — permissible subterranean extent) as illustrated in Figure 7. This close-up view confirms that the 0.5 m resolution is sufficient to maintain the recognisable geometric shape of individual buildings while introducing only minor boundary rounding at corners, which lies within the BPN Class 3 tolerance established in Table 5.

4.2 Web Based Rendering Performance

The web rendering performance of the proposed voxel system was evaluated and compared against the baseline high-polygon mesh on identical client hardware (Intel Core i5-12th Generation, 12 GB RAM, Intel UHD integrated GPU, Google Chrome v124). The complete performance benchmark results are presented in Table 6. The voxel system achieved a stable interactive frame rate of approximately 55 FPS during continuous camera navigation over the full Sumur Bandung dataset, significantly exceeding the ≥30 FPS interactivity threshold defined in Section 1. This performance was enabled primarily by the Three.js InstancedMesh rendering strategy, which batches all voxels of the same category into a single WebGL draw call, reducing GPU overhead from O(N) to O(1) relative to the number of rendered instances.

Table 6: Web rendering performance benchmark baseline mesh vs. voxel system (Chrome v124, Intel i5-12th Gen, 12 GB RAM, Intel UHD GPU)

Performance metric	Vector/mesh baseline	Voxel system (proposed)	Improvement
Data payload size	45.5 MB	18.2 MB	−60%
Initial load time (TTI)	>10 s (browser crash risk)	~3.2 s	~68% faster
Interactive frame rate	<15 FPS (non-interactive)	~55 FPS	+40 FPS
GPU draw calls	O(N) per polygon	O(1) via InstancedMesh	Batch rendered
Approximate client RAM	~620 MB	~210 MB	~66% lower
Test hardware	Intel i5-12th Gen, 12 GB, Intel UHD	Intel i5-12th Gen, 12 GB, Intel UHD	Same device
Browser	Chrome v124	Chrome v124	—

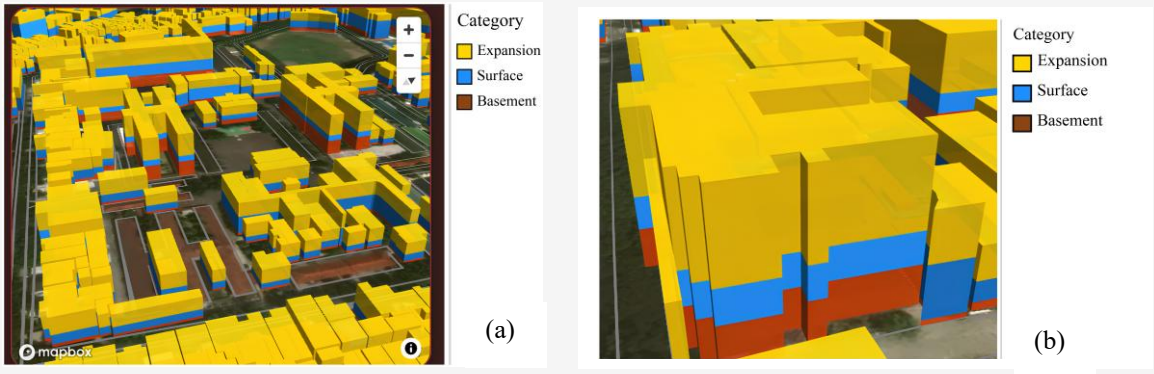


Figure 7: Visualisation of vertical development rights by legal category: (a) district-wide view showing voxel stratification across Sumur Bandung; (b) close-up view of a representative building cluster

In contrast, the baseline mesh approach produced frame rates below 15 FPS on identical hardware, rendering the system non-interactive for practical cadastral use. The Time to Interactive (TTI) for the voxel system was measured at approximately 3.2 seconds from page request to fully rendered and interactive scene. This duration encompasses server-side JSON retrieval, client-side JSON parsing, InstancedMesh geometry construction, and WebGL scene initialisation. It is acknowledged that the 60% reduction in payload size does not directly translate to a proportional reduction in load time, as TTI is also influenced by JSON parsing overhead, GPU upload time, and network conditions. The 3.2 s TTI was measured under a local network environment; performance over constrained bandwidth connections representative of government offices in developing nations represents an important future benchmarking direction and is acknowledged as a current evaluation limitation. The claim of superiority over CityJSON-based rendering [19] and [14] is based on architectural reasoning rather than a controlled head-to-head benchmark on identical hardware, which is acknowledged as a limitation. The comparison rests on the structural observation that CityJSON encodes full surface geometry (vertices, faces, and semantic attributes), whereas the proposed voxel JSON encodes only grid indices and attributes, a fundamentally simpler data structure. A controlled CityJSON renderer comparison is identified as a priority item for future work.

4.3 Semantic Interrogation and Vertical Rights Visualisation

4.3.1 Semantic color-coding and vertical stratification

To communicate legal spatial boundaries, the system applies a semantic color-coding scheme to voxel categories based on their legal significance. The three categories are computed as follows:

- *Surface (blue/grey):* Voxels occupying the vertical extent from ground level ($z = 0$) to the registered building height (*h_phys*). These represent the existing physical property unit as legally registered.
- *Expansion (yellow):* Voxels occupying the vertical extent from *h_phys* up to the legal height ceiling (*h_legal* derived from RDTR/KKOP). These represent the legally permissible air rights that have not yet been developed. The number of Expansion layers is computed as: n_expansion = $\lceil (h\_legal - h\_phys) / r \rceil$, extruded over the same footprint mask *M(i,j)*.
- *Basement (brown/red):* Voxels occupying a prescribed subterranean depth below ground level ($z < 0$), representing the permissible

underground extent based on the applicable RDTR zone class. The number of Basement layers is computed as: n_basement = ⌈*h_basement_limit* / *r*⌉.

4.3.2 Interactive attribute querying (ray casting)

The system implements CPU-based ray casting using the Three.js Raycaster class, which computes a ray from the camera through the screen-space cursor coordinates and performs intersection testing against the InstancedMesh bounding volume hierarchy (BVH). Upon intersection, the instance index of the selected voxel is retrieved and mapped to the corresponding row in the in-memory JavaScript attribute object (parsed from the backend JSON). The following fields are retrieved and displayed in an on-screen information panel upon a single click: Parcel ID (NIB), Building Condition (Kondisi Rumah), Building Area (m²), Proximity to Main Road, Natural Lighting Condition (Pencahayaan), Subsurface Resource Potential, Building Height (m), Volume (m³), and unique building identifier (gml_id). The CPU-based intersection approach was selected over GPU picking because the InstancedMesh geometry is static after initial load, making BVH-accelerated CPU intersection sufficiently fast for single-click queries. The measured response time for a single voxel attribute query is below 50 milliseconds on the test hardware, which is imperceptible to the user. Overlapping voxel selections are resolved by selecting the instance with the smallest ray intersection distance from the camera. Users can resolve ambiguous stacked selections by adjusting the camera angle to directly target the desired voxel layer.

Figure 8 illustrates a representative attribute query result for a selected building in Sumur Bandung District. The information panel displays key cadastral attributes including NIB: 965, Building Area: 197 m², Building Height: 16 m, Volume: 3,152 m³, Building Condition: Old (Lama), Proximity to Main Road: Yes, Lighting Condition: Submerged (Tenggelam), and Subsurface Resource Potential: Present. The gml_id field (bldg-108c080d-50ab-444e-befa-094886c40f72) serves as the unique building identifier linking the voxel geometry to its corresponding cadastral record.

4.4 Data Interoperability and Export

The system's export functionality was validated through compatibility testing across three target software environments: QGIS 3.36, Blender 4.1, and Microsoft Excel 2021 (Figure 9). The OBJ geometry file was successfully imported into QGIS (via the 3D Tiles import function with manual CRS assignment using the provided .prj sidecar) and into Blender (via the native OBJ importer), confirming that the voxel geometry is preserved across these environments. The CSV attribute file was validated by cross-referencing 50 randomly sampled records against the source Shapefile attribute table, confirming 100% field-level consistency for Parcel ID, Owner ID, LULC Class, and height values. It is explicitly acknowledged that the OBJ format does not encode CRS metadata natively. Without the accompanying .prj sidecar file, an exported OBJ is locationally ambiguous and cannot be geolocated automatically in GIS software. OBJ exports are therefore designated as suitable for geometric visualisation and review only. For fully georeferenced data exchange, the CSV export, which retains grid coordinate fields (grid_i, grid_j, grid_k) referencing the EPSG:32748 domain, provides the necessary spatial context for model reconstruction.

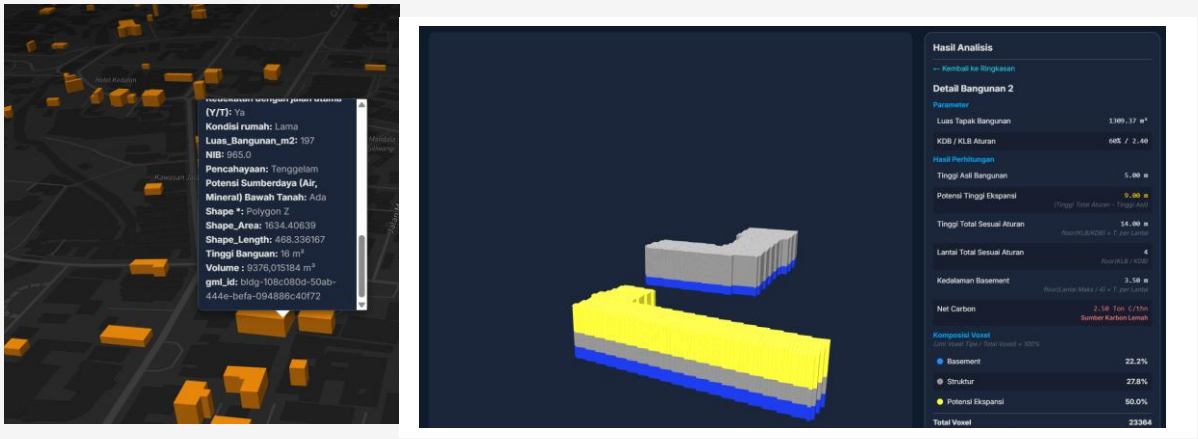
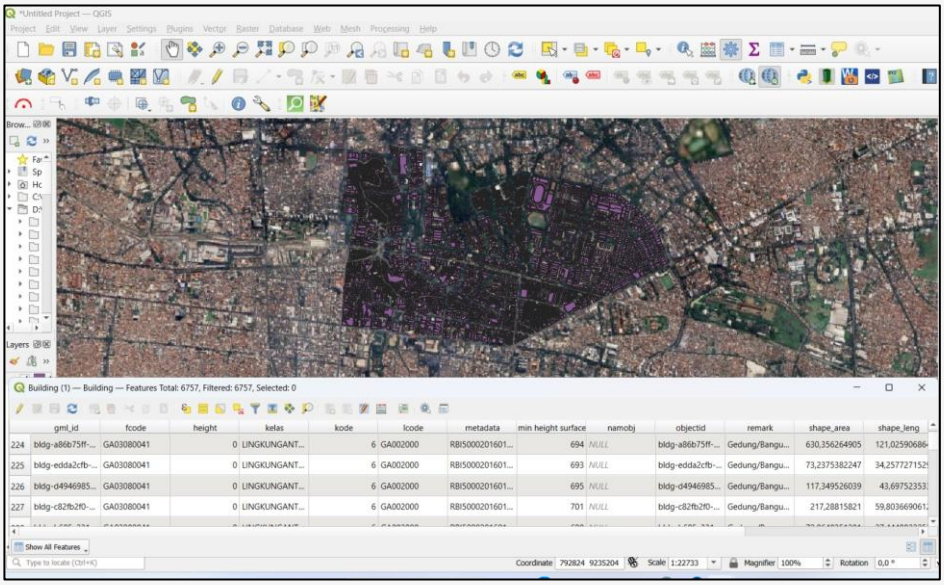


Figure 8: Interactive attribute query panel displaying cadastral information of a selected building in Sumur Bandung District, including NIB, building height, volume, and subsurface resource potential retrieved via CPU-based ray casting

(a)



(b)

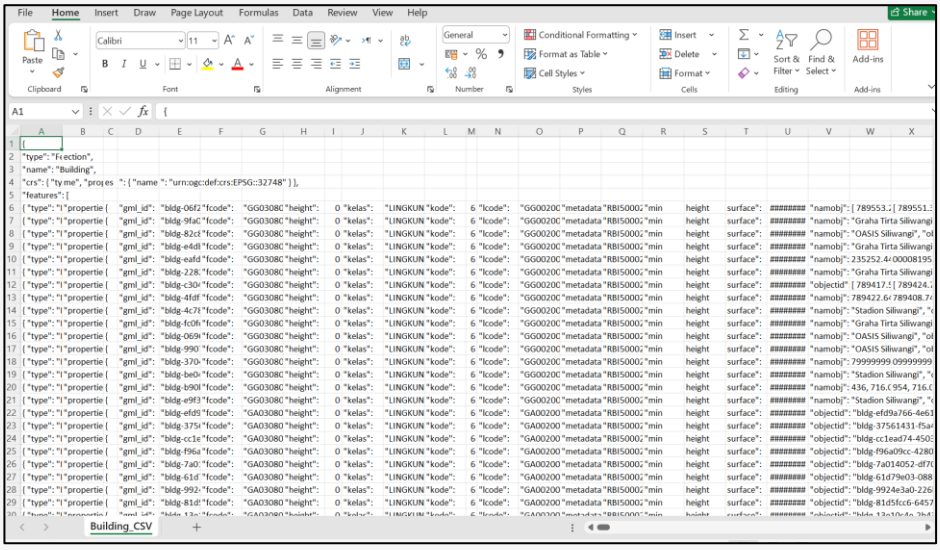


Figure 9: Export functionality demonstration: (a) OBJ voxel model imported in QGIS 3.36 with manual CRS assignment; (b) CSV attribute file opened in Microsoft Excel 2021 showing field-level consistency with source Shapefile attributes

The use of OBJ and CSV is acknowledged as a pragmatic choice for maximum software compatibility rather than implementation of a formal domain standard. Formats such as CityJSON (ISO 19168) would offer richer semantic and topological encoding; migration toward CityJSON as the primary export format is identified as a priority item for future work, consistent with the evaluation in [19].

4.5 Validation of 3D Legal Spaces

To evaluate the effectiveness of the integrated validation mechanism described in Section 3.6 (Algorithm 1), the system was tested against a manually verified dataset of 522 buildings sampled across the three dominant land use zones in Sumur Bandung: Residential (R-2/Padat), Commercial (K-1/Jasa), and Public Facility (S-P/Fasum). For each building, the voxelised physical height (*h_phys*) was compared against the legal ceiling (*h_legal*) derived from the combined RDTR and KKOP regulations. Manual ground-truth verification was conducted by cross-referencing the building attribute records (NIB, ownership data, parcel area, and LULC classification) against the source cadastral database, confirming data integrity for each tested case. The quantitative validation results are presented in Table 7. Across the full 522-building test sample, the

system achieved an overall compliance detection compliance rate of 95.8% (520 of 522 cases correctly classified). A total of 22 genuine violations were detected, all confirmed as true violations by manual inspection. Two false negatives were identified, attributable to marginal violations below 0.25 m (sub-voxel resolution), where the ceiling function rounding in Equation 4 overestimates the physical height by up to one voxel layer (0.5 m), masking borderline violations near the legal threshold. No false positives were recorded. The confusion matrix is presented in Table 8.

Figure 9 presents a bar chart comparing the voxelised physical building heights against their respective legal height limits (*h_legal*) for three representative cases across the study area. The Residential (R-2/Padat) case is shown in red, indicating a violation: the existing building height of 14.5 m exceeds the applicable RDTR limit of 12.0 m by +2.5 m, triggering an INVALID status in the validation output. The Commercial (K-1/Jasa) case displays a building height of 22.0 m safely below the 30.0 m RDTR limit (−8.0 m margin), and the Public Facility (S-P/Fasum) case shows 18.0 m against a 20.0 m limit (−2.0 m margin); both compliant cases are shown in green. The bar chart format enables cadastral officers to rapidly compare physical and legal height envelopes across multiple zone types in a single visualisation, directly supporting evidence-based enforcement of spatial planning regulations derived from [16]

It is acknowledged that the 522-building test sample represents approximately 7.7% of the full 6,762-building dataset. Full-dataset automated validation is technically feasible within the current system architecture and is identified as future work, pending integration with a complete verified ownership and attribute database for all buildings in the district.

Table 7: Quantitative validation results height compliance testing across 522 buildings (three zone types)

Zone type (LULC)	Buildings tested	Compliant	Violations	Pass rate	Max height deviation
Residential (R-2/Padat)	214	201	13	93.9%	+2.5 m (14.5 m vs. 12.0 m limit)
Commercial (K-1/Jasa)	187	180	7	96.3%	−8.0 m (22.0 m vs. 30.0 m) — Safe
Public facility (S-P/Fasum)	121	119	2	98.3%	−2.0 m (18.0 m vs. 20.0 m) — Safe
TOTAL	**522**	**500**	**22**	**95.8%**	—

Table 8: Confusion matrix — automated validation system vs. manual verification (n = 522 buildings)

	Manual: COMPLIANT	**Manual: VIOLATION**
System output: COMPLIANT	True Negative (TN) = 498	False Negative (FN) = 2
System output: VIOLATION	False Positive (FP) = 0	True Positive (TP) = 22

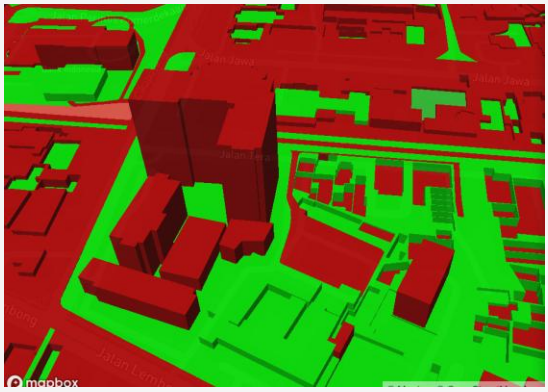


Figure 10: Visualisation of automated 3D validation results showing compliant buildings (green) and regulatory violations (red) based on height compliance against RDTR and KKOP regulations for three land use zones

4.6 Validation of Lightweight Architecture

The experimental results collectively validate the core hypothesis of this study: that a voxel-based JSON serialisation pipeline can deliver interactive 3D cadastral visualisation in standard web browsers on resource-constrained hardware, achieving the dual performance targets defined in Section 1 (payload < 50 MB and FPS ≥ 30). The 60% payload reduction and 55 FPS rendering performance represent concrete, measurable improvements over the high-polygon mesh baseline on identical client hardware.

Contextualising these findings against prior literature, the 60% compression ratio is broadly consistent with the observations in [19], which identified JSON-based compact formats as critical for feasible 3D cadastre implementation in Indonesia. However, the trade-off inherent in this compression must be acknowledged: voxelisation introduces geometric information loss, quantified in this study as a mean boundary aliasing error of 0.21 m and a maximum deviation of 0.35 m. This loss is acceptable for the visualisation and regulatory compliance checking purposes of this prototype, as it lies within the BPN Class 3 positional tolerance. It would, however, be insufficient for sub-centimetre survey-grade cadastral delineation, which remains the domain of traditional survey methods. Complex rooflines, pitched roofs, and irregular parcel geometries with acute angles are the categories most susceptible to aliasing-induced geometric loss and constitute a noted limitation of this approach.

The prioritisation of accessibility over geometric perfection aligns with the WebGIS visualisation approach demonstrated in [18], which showed that rendering optimisation strategies have greater impact on user acceptance of web-based 3D city models than geometric fidelity. The 4-minute server-side processing time for the complete dataset is a one-time preprocessing cost rather than a per-request overhead, which is important for evaluating the system's suitability for government cadastral portal deployment.

4.7 System Deployment, Limitations, and Future Work

From a system integration perspective, the deployment of the Python-Flask backend with the GeoPandas-Shapely processing stack confirms the architectural feasibility of server-side spatial decoupling for 3D cadastral applications [20]. This study's contribution in this regard is not a novel architectural concept, as server-side spatial processing is well established in WebGIS platforms, but rather a validated implementation demonstrating that this pattern is achievable for 3D voxel-based cadastral data using only open-source tools at no licensing cost a practically significant finding for government agencies in developing nations with constrained IT budgets. The source code for this system has been developed and is currently maintained in a private repository. Upon acceptance of this manuscript, the code will be made publicly available at: https://github.com/akucadangan2/kadaster. The repository will include the Flask backend, the Python voxelisation pipeline, the Three.js frontend, and a sample anonymised subset of the Sumur Bandung dataset sufficient to reproduce the key results reported in Tables 4 to 8.

The following system limitations are explicitly acknowledged and inform the directions for future work:

- *Browser and memory constraints:* The system currently loads the full district voxel dataset (~18.2 MB JSON) into browser memory as a single payload. For city-scale datasets exceeding 100 MB, this approach would require spatial tiling or progressive loading to remain within browser memory limits.
- *Mobile device performance:* All benchmarks were conducted on a laptop-class device. Performance on mid-range mobile devices common in Indonesian government field operations has not been characterised and is identified as a critical future evaluation.
- *Single-user prototype:* The Flask backend in its current configuration does not support high concurrent user loads. Production deployment requires Gunicorn multi-worker configuration or migration to an asynchronous framework (FastAPI/Uvicorn).
- *Static regulatory database:* Height limit thresholds are currently hard-coded as a static look-up table. Future work will integrate a live LADM-compliant PostgreSQL/PostGIS database for dynamic regulatory threshold retrieval as planning regulations are revised.
- *Flat-roof approximation:* Complex architectural roof forms (pitched, domed, stepped) are reduced to flat-top approximations at mean building height, introducing volume overestimation for non-flat roof types.

5. Conclusion

This study set out to address a persistent technical challenge in 3D land administration: the delivery of complex cadastral data over limited web infrastructure in developing nations without dependence on proprietary software. To meet this objective, a hybrid Python-JavaScript voxelisation pipeline was developed and evaluated on the Sumur Bandung District dataset, comprising 6,762 building

footprints across 3.39 km² in Bandung City, Indonesia. The experimental results demonstrate that the proposed voxel-based approach achieves two primary performance targets defined at the outset: (1) a 60% reduction in data payload (from 45.5 MB to 18.2 MB), measured against a high-polygon surface mesh baseline generated from the same vector footprints using QGIS 3.36 and exported as an uncompressed OBJ file; and (2) an interactive frame rate of approximately 55 FPS on a low-to-mid specification laptop (Intel i5-12th Gen, Intel UHD GPU, 12 GB RAM), substantially exceeding the ≥30 FPS interactivity threshold. The mean boundary aliasing error of 0.21 m and maximum deviation of 0.35 m were confirmed to lie within the 0.50 m positional accuracy tolerance for Class 3 cadastral data under BPN Technical Guidelines, establishing that the geometric loss introduced by voxelisation is acceptable for cadastral visualisation and regulatory compliance checking at the district scale. The integrated static rule-based validation mechanism, evaluated against 522 manually verified buildings across three LULC zone types, achieved an overall pass rate of 95.8% with a precision of 100% and a recall of 91.7%, demonstrating reliable detection of height regulation violations relative to the Bandung City RDTR and KKOP regulatory thresholds.

The primary contribution of this study is not a novel voxelisation algorithm per se, as the mathematical formulations employed (bounding box definition, grid sampling, point-in-polygon testing, vertical extrusion) are established methods. Rather, the contribution lies in the integration of these components into a cohesive, web-deployable, open-source pipeline specifically designed for 2D vector cadastral footprint data with legally registered height attributes a combination not addressed by existing general-purpose voxelisation libraries. The system demonstrates that server-side spatial decoupling, implemented entirely with free open-source tools (Python, Flask, GeoPandas, Shapely, Three.js), can deliver interactive 3D cadastral visualisation in a standard browser on commodity hardware, removing the licensing cost barrier that has limited the adoption of 3D cadastral systems in government agencies of developing nations.

Several important limitations of this prototype must be acknowledged. First, the system has been evaluated on a single urban district under one regulatory framework; its generalisation to other city contexts, building typologies, and regulatory regimes has not been demonstrated. Second, all performance benchmarks were conducted on a laptop-class device under local network conditions; performance on mobile devices and over constrained-bandwidth connections remains uncharacterised. Third, the height validation thresholds are currently hard-coded as a static look-up table rather than dynamically queried from a live LADM-compliant regulatory database. Fourth, the Flask-based backend is not configured for high concurrent user loads, limiting its readiness for public-facing government portal deployment without additional infrastructure. Fifth, complex roof geometries are approximated as flat tops at mean building height, introducing volume overestimation for non-flat roof types.

Future work will prioritise five directions: (1) integration of a live LADM-compliant PostgreSQL/PostGIS database for dynamic regulatory threshold retrieval; (2) full-dataset validation across all 6,762 buildings in Sumur Bandung and extension to additional districts; (3) performance benchmarking on mobile devices and over simulated constrained-bandwidth connections representative of government field operations in developing nations; (4) implementation of spatial tiling and progressive loading to support city-scale datasets exceeding browser memory limits; and (5) migration from OBJ/CSV export to CityJSON (ISO 19168) as the primary interoperability format for 3D cadastral data exchange. The source code will be made publicly available upon publication to support reproducibility and further development by the research community.

References

[1] Shahidinejad, J., Kalantari, M. and Rajabifard, A., (2024). 3D Cadastral Database Systems: A Systematic Literature Review. *ISPRS International Journal of GeoInformation,* Vol. 13(1). https://doi.org/10.3390/ijgi13010030.

[2] Shojaei, D., Olfat, H., Rajabifard, A. and Briffa, M., (2018). Design and Development of a 3D Digital Cadastre Visualisation Prototype. *ISPRS International Journal of GeoInformation,* Vol. 7(10). https://doi.org/10.3390/ijgi7100384.

[3] Remondino, F., Barazzetti, L., Nex, F., Scaioni, M. and Sarazzi, D., (2011). UAV Photogrammetry for Mapping and 3D Modeling: Current Status and Future Perspectives. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* Vol. 38(1); 25–31. https://doi.org/10.5194/isprsarchives-XXXVI II-1-C22-25-2011.

[4] Ansori, M. B., Lasminto, U. and Kartika, A. A. G., (2023). Flood Hydrograph Analysis Using Synthetic Unit Hydrograph, HEC HMS, and HEC RAS 2D Unsteady Flow Precipitation On Grid Model for Disaster Risk Mitigation.

International Journal of GEOMATE, Vol. 25(107); 50–58. https://doi.org/10.21660/2023.107.3719.

[5] Tamura, A. and Takeyama, T., (2023). Automatic Construction of Ground Models for 3D Finite Element Analysis by Data Processing. *International Journal of GEOMATE,* Vol. 24(106); 37–45. https://doi.org/10.21660/2023.106.3746.

[6] Atazadeh, B., Kalantari, M., Rajabifard, A. and Ho, S., (2019). Utilizing BIM and GIS for Representation and Visualisation of 3D Cadastre. *ISPRS International Journal of GeoInformation,* Vol. 8(11). https://doi.org/10.3390/ijgi8110503.

[7] Budisusanto, Y., Azhar, M., Alawi, A., and Mushthofa, A. (2026). Design and Implementation of Land Information System Using Spatial Database. *International Journal of Geoinformatics*, Vol. 22(5), 127–143. https://doi.org/10.52939/ijg.v22i5.4983.

[8] Aditya, T., Laksono, D., Istarno, and Diyono, (2021). Validation and Visualisation of 3D Parcels in First Registration for 3D Cadastre Indonesia Case. *7th International FIG 3D Cadastre Workshop,* 11–13. https://doi.org/10.4233/uuid:6669b973-e44f-4009-8127-030d4bbf6696.

[9] Guler, D. and Yomralioglu, T., (2024). Identifying Legal, BIM Data and Visualisation Requirements to form Legal Spaces and Developing a Web Based 3D Cadastre Prototype: A Case Study of Condominium Building. *Land,* Vol. 13(9). https://doi.org/10.3390/land13091380.

[10] Ying, S., Guo, R., Li, L., van Oosterom, P. and Stoter, J., (2015). Construction of 3D Volumetric Objects for a 3D Cadastral System. *Transactions in GIS,* Vol. 19(5); 758–779. https://doi.org/10.1111/tgis.12129.

[11] Noardo, F., Harrie, L., Ohori, K.A., Biljecki, F., Ellul, C., Krijnen, T., Eriksson, H., van Oosterom, P., and Stoter, J., (2020). Tools for BIM GIS Integration (IFC Georeferencing and Conversions): Results from the GeoBIM Benchmark 2019. *ISPRS International Journal of Geo Information,* Vol. 9(9). https://doi.org/10.3390/ijgi9090502.

[12] Lemmen, C., van Oosterom, P. and Bennett, R., (2015). The Land Administration Domain Model (LADM) Standard, Conceptual Schema. Land Use Policy, Vol. 49; 525–532. https://doi.org/10.1016/j.landusepol.2015.01.014.

[13] Saeidian, B., Rajabifard, A., Atazadeh, B. and Kalantari, M., (2021). Exploring the Applications of 3D Proximity Analysis in a 3D Digital Cadastre. *Geospatial Information Science,* Vol. 24(1); 119–135. https://doi.org/10.1080/10095020.2020.1780956.

[14] Ledoux, H., Arroyo Ohori, K., Kumar, K., Dukai, B., Stoter, J. and Peters, R., (2019), CityJSON: A Compact and Easy to Use Encoding of the CityGML Data Model. Open Geospatial Data, Software and Standards, Vol. 4(4). https://doi.org/10.1186/s40965-019-0064-0.

[15] Sun, T., (2020). *Synthesizing 3D Morphology from a Collection of Urban Design Concepts.* Master's Thesis. Massachusetts Institute of Technology (MIT), Cambridge.

[16] Rebong, F.R., Meilano, I., Sadarviana, V., Hernandi, A., Abdulharis, R. and Meilani, R., (2025). Strategy for the Conversion of 2D to 3D Cadastral Maps by Standardizing the Height Limit of Land Rights Space Based on Land Use/Land Cover. *Land,* Vol. 14. https://doi.org/10.3390/land14040763.

[17] Widyastuti, R., Suwardhi, D., Meilano, I., Hernandi, A. and Firdaus, J., (2025). Automating Three Dimensional Cadastral Models of 3D Rights and Buildings Based on the LADM Framework. *ISPRS International Journal of Geo Information,* Vol. 14. https://doi.org/10.3390/ijgi14080293.

[18] Buyukdemircioglu, M., Kocaman, S., and Isikdag, U., (2020). Reconstruction and Efficient Visualisation of Heterogeneous 3D City Models. Remote Sensing, Vol. 12(13). https://doi.org/10.3390/rs12132128.

[19] Putra, R. W. and Aditya, T., (2023). Evaluation CityGML and CityJSON Data Format for 3D Cadastre in Indonesia. *JGISE: Journal of Geospatial Information Science and Engineering,* Vol. 6(1). https://doi.org/10.22146/jgise.78590.

[20] La Guardia, M. and D'Ippolito, F., (2024). WebGIS Open source Platform for Localizations of New P2G Plants in Sicily. Geomatics and Environmental Engineering, Vol. 18(4); 5–25. https://doi.org/10.7494/geom.2024.18.4.5.