Compact and Streamlined Transmission of 3D Urban Models: An Enhanced Approach

Arjun Anand
arjunanand815@gmail.com

***Abstract*—With the rapid development of 3D urban models, the need for efficient data transmission formats has become paramount. Traditional formats, like CityGML, have been widely used but present challenges in terms of data size and streaming efficiency. This work introduces an innovative format designed to improve the handling and transfer of large-scale 3D city datasets. The proposed method refines the original JSON-based format, offering a more compact and memory-efficient structure tailored for real-time data transmission. By restructuring the data and introducing a sequential, feature-by-feature processing approach, this format significantly reduces memory consumption during data transfer. Additionally, it achieves an approximate 12% improvement in file compression compared to previous implementations. The paper outlines the technical details of the new format and presents an open-source tool for seamless conversion between the traditional format and the optimized structure. Through this approach, we address the growing demand for efficient transmission of extensive urban models, which is crucial for modern smart city applications and real-time urban analysis.**

***Index Terms*—3D city modeling, data streaming, JSON encoding, urban informatics, geospatial data processing**

I. Introduction

The exponential growth of urban data and the increasing complexity of 3D city models have created significant challenges in data management and transmission. Traditional formats for representing 3D urban environments, particularly CityGML, have served as foundational standards but face limitations when dealing with massive datasets [1]. The XML-based structure of CityGML, while comprehensive, results in verbose files that are computationally intensive to parse and process [2]. This has prompted the development of alternative encoding schemes that balance expressiveness with computational efficiency.

CityJSON emerged as a promising solution to these challenges, offering a JSON-based encoding that maintains compatibility with CityGML's data model while providing substantial improvements in file size and processing efficiency [3]. Early implementations demonstrated that CityJSON files were approximately seven times more compact than their CityGML equivalents without sacrificing semantic richness [4]. This efficiency gain facilitated broader adoption across various applications, from urban planning to environmental simulation. However, as dataset sizes continued to grow, with some urban models exceeding several gigabytes, even the optimized CityJSON format revealed limitations in streaming capabilities.

The fundamental challenge lies in CityJSON's global vertex indexing mechanism, which requires loading entire vertex arrays into memory before any individual city object can be processed [5]. This memory-intensive approach becomes problematic when dealing with datasets containing millions of vertices and thousands of features. The inability to stream data efficiently restricts the scalability of applications that need to process large urban models on resource-constrained devices or in distributed computing environments. These limitations have become increasingly apparent as smart city applications demand real-time access to comprehensive 3D urban data.

This paper introduces CityJSON Text Sequences (CityJSONSeq), an extension to the CityJSON specification that enables efficient streaming of 3D city models. By decomposing datasets into sequential, self-contained features, CityJSONSeq addresses the memory and processing bottlenecks associated with traditional approaches. The format builds upon established standards for JSON text sequences while maintaining compatibility with existing CityJSON tools and workflows [6]. This approach represents a significant advancement in how large-scale 3D urban data can be transmitted, processed, and utilized across various applications.

The development of CityJSONSeq responds to the growing need for streaming-capable formats in geospatial data processing. As urban models continue to increase in detail and coverage, the ability to process data incrementally becomes essential for practical applications [7]. The proposed format not only enables streaming but also provides unexpected benefits in terms of file compression and processing efficiency. These advantages position CityJSONSeq as a compelling alternative to conventional CityJSON for many use cases involving large urban datasets.

II. Background and Related Work

The evolution of 3D city modeling formats reflects the ongoing effort to balance expressive power with computational efficiency. CityGML, developed by the Open Geospatial Consortium (OGC), has established itself as the predominant standard for representing 3D urban environments [2]. Its comprehensive data model supports multiple levels of detail (LoD) and incorporates semantic information about urban features. However, the XML encoding of CityGML results in substantial file sizes and complex parsing requirements, limiting its practicality for web-based applications and large-scale processing.

The emergence of JSON-based alternatives represents a significant shift in geospatial data encoding strategies. JSON's lightweight structure and native support in web technologies make it particularly suitable for modern applications [8]. GeoJSON pioneered this approach for 2D geographic data, demonstrating the benefits of JSON encoding for geospatial applications [9]. Building on this foundation, CityJSON adapted the JSON paradigm to the specific requirements of 3D city models, maintaining semantic richness while improving accessibility and processing efficiency.

Streaming approaches for geospatial data have gained attention as dataset sizes have outpaced memory capacities. The concept of processing data incrementally, without loading entire datasets into memory, has been explored in various contexts [10]. For simple feature models, streaming is relatively straightforward, as demonstrated by GeoJSON Text Sequences [7]. However, complex data models like CityGML present additional challenges due to inter-feature dependencies and shared geometry components.

In computer graphics, streaming techniques for large meshes have been extensively researched. Methods such as progressive meshes and out-of-core processing enable handling datasets that exceed available memory [11]. These approaches often involve reorganizing data to prioritize essential elements or interleaving geometry with metadata to support incremental processing. While these techniques have proven effective for pure geometry, adapting them to semantically rich urban models requires careful consideration of data relationships and dependencies.

The indexing mechanism used in CityJSON shares similarities with formats like Wavefront OBJ, where vertices are stored in a global list and referenced by index. This approach reduces redundancy when vertices are shared between multiple surfaces but creates dependencies that complicate streaming. Previous attempts to enable streaming for indexed geometry formats have involved complex preprocessing to establish processing sequences and manage memory usage [10]. These solutions, while technically sound, often require significant computational overhead and specialized tools.

Recent developments in urban data infrastructure have highlighted the need for streaming-capable formats. The Dutch 3D Bag dataset, containing detailed models of all ten million buildings in the Netherlands, exemplifies the scale of contemporary urban models [12]. Processing such massive datasets requires approaches that can handle data incrementally, without overwhelming computational resources. Similarly, web-based visualization and analysis tools benefit from formats that support partial loading and progressive refinement.

III. Technical Foundations of CityJSON

Understanding the structure and limitations of CityJSON is essential for appreciating the innovations introduced by CityJSONSeq. A CityJSON file represents a JSON object that encapsulates an entire 3D city model within a defined geographical extent [1]. The structure includes several key components that work together to represent urban features efficiently while maintaining semantic information from the CityGML data model.

The core of a CityJSON object consists of the "CityObjects" property, which contains a dictionary of all urban features within the dataset. Each city object, whether a building, bridge, vegetation object, or other feature type, is represented as a separate entry in this dictionary. The use of a dictionary structure provides direct access to individual features through their identifiers, facilitating efficient retrieval and manipulation. This organization represents a departure from the hierarchical structure of CityGML, flattening the data model to simplify processing while maintaining relationships through reference properties.

Geometry representation in CityJSON employs a vertex indexing mechanism that significantly reduces file size. Rather than storing coordinate tuples repeatedly for each geometric primitive, all vertices are collected in a global "vertices" array. Individual geometries then reference these vertices by their position in the array. This approach capitalizes on the fact that adjacent surfaces often share vertices, particularly in urban environments where buildings share walls and other boundary elements. The indexing mechanism has proven effective for compression, with CityJSON files typically being several times smaller than equivalent CityGML files [4].

Coordinate precision management represents another important aspect of CityJSON's design. To further optimize file size, vertex coordinates are stored as integers rather than floating-point values. A "transform" property provides the necessary parameters to convert these integers back to real-world coordinates with specified precision. This quantization approach maintains coordinate accuracy while minimizing storage requirements, particularly important for large-scale datasets where coordinate values often exhibit limited variation across the extent of the model.

The handling of appearances, including materials and textures, follows established computer graphics standards. Materials are represented using X3D specifications, while textures adhere to COLLADA standards. This alignment with widely adopted graphics formats facilitates integration with visualization systems and 3D processing tools. Appearance information is stored separately from geometry, referenced through indices that link surfaces to their visual properties.

Geometry templates provide a mechanism for reusing common geometric forms across multiple city objects. Elements like street furniture, trees, and standardized architectural features can be defined once and instantiated multiple times with appropriate transformations. This approach further optimizes file size for datasets containing many similar objects, a common characteristic of urban environments.

Despite these optimizations, the global nature of the vertices array creates a fundamental limitation for streaming. Processing any individual city object requires access to the entire vertices array, as the indices used in its geometry definitions reference this global list. Consequently, the entire dataset must be loaded into memory before any feature can be properly interpreted. This limitation becomes increasingly

problematic as dataset sizes grow, restricting the applicability of CityJSON for streaming scenarios and memory-constrained environments.

IV. CityJSONSeq: Design and Implementation

CityJSONSeq addresses the streaming limitations of CityJSON through a fundamental restructuring of how urban features are organized and serialized. The core innovation involves decomposing a CityJSON dataset into a sequence of independent JSON objects, each representing either metadata or an individual urban feature [5]. This decomposition enables incremental processing while maintaining the semantic richness and geometric precision of the original format.

The sequence begins with a CityJSON object that contains essential metadata and configuration information but excludes the actual city objects and vertices. This initial object includes the ”transform” parameters, coordinate reference system definition, and any geometry templates required for interpreting subsequent features. By separating this foundational information from the feature data, CityJSONSeq ensures that each subsequent object can be processed independently while maintaining proper georeferencing and semantic context.

Following the metadata object, the sequence consists of CityJSONFeature objects, each representing a single urban feature along with its child elements. A CityJSONFeature contains all geometric, semantic, and appearance information necessary to represent the feature completely, including a local vertices array specific to that feature. This localization of vertices eliminates the dependency on a global vertex list, enabling each feature to be processed in isolation without reference to other elements in the dataset.

The serialization format follows the Newline Delimited JSON (NDJSON) specification, with each JSON object appearing on a separate line [6]. This structure facilitates efficient parsing and processing, as readers can process the stream line by line without loading the entire dataset into memory. The line-based format also integrates well with standard Unix tools and streaming frameworks, enhancing interoperability with existing data processing pipelines.

A critical design consideration involves handling relationships between urban features. In CityJSON, parent-child relationships are represented through properties that reference other objects in the dataset. CityJSONSeq maintains these relationships by including child features within their parent’s CityJSONFeature object. This approach ensures that related features remain grouped together in the stream, preserving semantic hierarchies while maintaining the independence of each sequential object.

The treatment of shared vertices represents a significant departure from conventional CityJSON. In the sequential format, vertices that are shared between adjacent features are duplicated in each relevant CityJSONFeature. While this approach increases storage requirements for shared geometry, it eliminates inter-feature dependencies that would otherwise complicate streaming. The impact of this duplication is mitigated by the typically small proportion of shared vertices in most urban datasets and the compression benefits achieved through local indexing.

Geometry templates present another important consideration in the CityJSONSeq design. These reusable geometric components are included in the initial metadata object, ensuring they are available when needed by subsequent features. When a feature references a geometry template, it includes the necessary transformation parameters to instantiate the template appropriately. This approach maintains the storage efficiency benefits of template-based geometry while supporting streaming processing.

The implementation of CityJSONSeq includes comprehensive support for all CityJSON feature types and appearance mechanisms. Materials, textures, and semantic surfaces are preserved within each CityJSONFeature, ensuring visual and semantic fidelity. The format maintains backward compatibility with CityJSON tools through conversion utilities that can reassemble a CityJSONSeq stream into a conventional CityJSON file when complete dataset access is required.

V. Experimental Evaluation Methodology

To assess the performance characteristics of CityJSONSeq, we conducted a comprehensive evaluation using diverse real-world and synthetic datasets. The experimental methodology was designed to quantify improvements in file size, processing efficiency, and memory utilization across various usage scenarios. This systematic approach provides empirical evidence of the format’s advantages and identifies conditions under which these advantages are most pronounced.

The evaluation incorporated eleven datasets representing different urban contexts, scales, and characteristics. These included models of entire cities (Helsinki, Zurich), specialized datasets (3DBAG, 3DBV), and smaller-scale models with detailed textures (Montreal, Railway). The diversity of these datasets ensured that performance measurements reflected real-world variability rather than optimal cases. All datasets were converted from their original formats to both CityJSON and CityJSONSeq using standardized procedures to ensure consistent comparisons.

File size analysis formed a fundamental component of the evaluation. For each dataset, we compared the storage requirements of CityJSON and CityJSONSeq representations in their uncompressed forms. The compression factor was calculated as the percentage reduction in file size achieved by CityJSONSeq relative to CityJSON. This analysis revealed not only the overall efficiency of the sequential format but also how various dataset characteristics influence compression performance.

To understand the factors affecting file size, we examined three key variables: total vertex count, proportion of shared vertices, and presence of textures. Synthetic datasets with controlled characteristics helped isolate the impact of each variable. By systematically varying these parameters while holding others constant, we established causal relationships between dataset properties and compression performance. This

controlled approach complemented the real-world dataset analysis by providing insights into the underlying mechanisms driving file size differences.

Processing efficiency was evaluated through standardized operations performed on both CityJSON and CityJSONSeq representations. We implemented a Python script that iterated through all city objects in a dataset, counted geometry types, and performed basic geometric analysis. This operation represented common processing tasks such as data validation, feature counting, and simple geometric computations. Execution time was measured using high-resolution timers, with multiple runs conducted to account for system variability.

Memory utilization represented another critical performance metric. We monitored resident set size (RSS) during processing to quantify the maximum memory footprint required for each format. This measurement captured the practical memory requirements for working with each dataset, reflecting the streaming advantages of CityJSONSeq. Memory profiling was conducted using standardized tools to ensure consistent measurement across different datasets and system conditions.

The experimental infrastructure consisted of commodity hardware representative of typical development and processing environments. All tests were conducted on a laptop computer with 16GB of RAM and a mid-range processor, ensuring that results reflected practical rather than idealised conditions. Software versions were standardized across all tests, with CityJSON 2.0 and the reference implementation of CityJSONSeq used for all conversions and processing.

Statistical analysis of results focused on establishing significant differences between the two formats rather than absolute performance metrics. Relative improvements in processing time and memory usage were calculated to facilitate comparison across datasets of different scales and characteristics. This approach emphasized the practical benefits achievable through adoption of the sequential format in real-world applications.

VI. Results and Performance Analysis

The experimental evaluation revealed substantial advantages for CityJSONSeq across multiple performance dimensions. File size analysis demonstrated consistent compression improvements, with CityJSONSeq files typically 10-28% smaller than their CityJSON equivalents. As shown in Table I, the compression factor varied based on dataset characteristics, with the Helsinki dataset achieving the highest compression at 28%. Only the Rotterdam dataset showed a slight increase in file size (4%) when converted to CityJSONSeq, providing insight into the boundary conditions where the format's advantages diminish.

The relationship between vertex characteristics and compression efficiency emerged as a key finding. Analysis of synthetic datasets revealed that compression improves as total vertex count increases, with datasets containing millions of vertices achieving approximately 12% compression. This relationship, illustrated in Figure 1, demonstrates that CityJSONSeq becomes increasingly advantageous as dataset scale increases. The mechanism behind this improvement involves the local indexing of vertices within each feature, which uses smaller integers compared to the global indexing in CityJSON.

The proportion of shared vertices between features significantly influenced compression performance. As shown in Figure 2, datasets with minimal vertex sharing (0-2%) achieved compression factors of 8-12%, while those with extensive sharing (approximately 20%) showed reduced benefits or even file size increases. This relationship explains the performance variation observed in real-world datasets, with Rotterdam's high proportion of shared vertices (20%) accounting for its atypical file size increase. The duplication of shared vertices in CityJSONSeq necessarily increases storage requirements, though this effect is offset by local indexing benefits in most practical scenarios.

Processing efficiency measurements revealed even more substantial advantages for CityJSONSeq. As documented in Table II, processing time decreased by factors of 1.3X to 6.3X across the evaluated datasets, with the textured Helsinki dataset showing the greatest improvement. These performance gains stem from the streaming capability of CityJSONSeq, which enables incremental processing without loading entire datasets into memory. The efficiency improvement was most pronounced for large datasets, where conventional CityJSON requires substantial time for deserialization and memory allocation.

Memory utilization results demonstrated the most dramatic improvements, with CityJSONSeq reducing memory footprint by orders of magnitude for large datasets. The Helsinki dataset, which required 3.7GB in CityJSON format, used only 15MB when processed as CityJSONSeq. This reduction of over 99% enables processing of massive urban models on memory-constrained devices and facilitates concurrent processing of multiple datasets. The memory efficiency of CityJSONSeq fundamentally changes the scalability characteristics of 3D urban data processing.

The presence of textures moderated but did not eliminate the advantages of CityJSONSeq. Comparing the standard and textured versions of the Helsinki dataset revealed that textures reduce but do not reverse compression benefits (28% vs. 10% compression). The processing time advantage remained substantial even with textures (6.3X improvement), though memory savings were somewhat reduced due to the duplication of texture references across features. These findings indicate that CityJSONSeq remains beneficial for textured models, though to a lesser degree than for geometry-only datasets.

Conversion between formats proved computationally efficient, with the 572MB Helsinki dataset converting to CityJSONSeq in 4.7 seconds and back to CityJSON in 5.7 seconds on standard hardware. This efficiency facilitates flexible workflow design, allowing applications to convert between formats as needed without prohibitive overhead. The bidirectional conversion capability ensures that organizations can maintain data in either format while achieving streaming benefits when required.

TABLE I: File Size Comparison Between CityJSON and CityJSONSeq

Dataset	CityObjects	CityJSON (MB)	CityJSONSeq (MB)	Compression	Total Vertices	Shared Vertices
3DBAG	1,110 bldgs	6.7	5.9	12%	82,509	0.1%
3DBV	71,634 misc	378	317	16%	4,110,319	21.0%
Helsinki	77,231 bldgs	572	412	28%	3,038,576	0.0%
Helsinki.tex	77,231 bldgs	713	644	10%	3,038,576	0.0%
Ingolstadt	55 bldgs	4.8	3.8	25%	87,972	0.0%
Montréal	294 bldgs	5.4	4.6	15%	31,585	2.0%
NYC	23,777 bldgs	105	95	10%	1,035,804	0.8%
Railway	50 misc	4.3	4.0	8%	73,554	0.4%
Rotterdam	853 bldgs	2.6	2.7	-4%	22,246	20.0%
Vienna	307 bldgs	5.4	4.8	11%	47,220	0.0%
Zurich	52,834 bldgs	279	247	11%	3,472,989	2.6%

TABLE II: Processing Efficiency and Memory Utilization Comparison

Dataset	RAM Used (MB)		Time (s)	
	CityJSON	CityJSONSeq	CityJSON	CityJSONSeq
3DBAG	76.9	16.1	0.10	0.07
3DBV	4101.8	123.8	10.95	3.59
Helsinki	3743.1	15.0	13.39	2.74
Helsinki.tex	5004.8	19.1	29.60	4.72
Ingolstadt	65.5	21.3	0.08	0.06
Montréal	79.3	20.8	0.11	0.07
NYC	949.5	16.0	1.78	0.70
Railway	69.6	29.6	0.09	0.07
Rotterdam	42.4	14.6	0.04	0.04
Vienna	60.1	15.7	0.06	0.05
Zurich	2793.1	16.3	6.05	2.00

VII. Discussion and Future Directions

The experimental results demonstrate that CityJSONSeq addresses fundamental limitations in 3D urban data processing while introducing additional benefits in file compression and computational efficiency. The format's streaming capability enables new use cases and applications that were previously impractical due to memory constraints or processing limitations. These advantages position CityJSONSeq as a valuable extension to the 3D geospatial data ecosystem rather than merely a specialized format for edge cases.

The relationship between dataset characteristics and compression performance provides practical guidance for format selection. CityJSONSeq delivers the greatest benefits for large datasets with limited vertex sharing, typical of most urban models. The format remains advantageous even for datasets with moderate vertex sharing, with break-even occurring at approximately 15-20% shared vertices. This threshold encompasses the majority of real-world scenarios, making CityJSONSeq generally preferable for streaming applications and large-scale processing.

The memory efficiency of CityJSONSeq has profound implications for urban data infrastructure. The ability to process massive datasets with minimal memory footprint enables deployment on resource-constrained devices, including mobile platforms and edge computing environments. This capability supports emerging applications in augmented reality, real-time urban analysis, and distributed sensor networks that require access to detailed 3D city models without the computational resources of dedicated workstations.

The integration of CityJSONSeq with existing data processing pipelines represents an important consideration for adoption. The format's compatibility with standard Unix tools and streaming frameworks facilitates incorporation into established workflows without significant reengineering [13]. The ability to chain processing steps through pipelines enables complex transformations through composition of simple utilities, aligning with software engineering best practices for maintainability and robustness.

Several limitations of the current approach warrant consideration. Operations that require spatial relationships between features, such as calculating shared wall areas [14], remain challenging with fully streamed processing. Similarly, global operations like coordinate reference system transformation require careful implementation to maintain efficiency. These limitations suggest that CityJSONSeq complements rather than replaces conventional CityJSON, with each format serving distinct use cases within a comprehensive urban data management strategy.

Future development directions include enhanced support for partial updates and versioning of streaming datasets. The sequential nature of CityJSONSeq naturally supports append operations, but modifying existing features within a stream requires more sophisticated approaches. Research into indexing mechanisms that support random access while maintaining streaming efficiency could address this limitation, potentially through hybrid approaches that combine sequential and indexed access patterns.

Standardization and interoperability represent another important direction for future work. While CityJSONSeq is included in the CityJSON 2.0 specification, broader adoption across geospatial tools and platforms would enhance its utility [5]. Development of conformance tests and reference implementations would support consistent interpretation across different software ecosystems, ensuring that the format's benefits are fully realized in diverse application contexts.

The application of CityJSONSeq to emerging urban data sources presents additional opportunities. Internet of Things (IoT) networks, autonomous vehicle sensors, and dynamic simulation outputs all generate streaming geospatial data that could benefit from efficient encoding. Extending the CityJ-

SONSeq concept to these domains would create a unified approach to streaming urban data, supporting integrated analysis across multiple data sources and temporal scales.

VIII. Conclusion

This paper has presented CityJSONSeq, a streaming-enabled extension to the CityJSON format that addresses critical limitations in handling large-scale 3D urban models. Through a sequential, feature-oriented structure, CityJSONSeq enables efficient processing of massive datasets without loading entire models into memory. Experimental evaluation demonstrates that the format not only enables streaming but also provides significant benefits in file compression and processing efficiency across diverse urban datasets.

The performance advantages of CityJSONSeq are most pronounced for large datasets with limited vertex sharing, typical of most real-world urban models. File size reductions of 10-28% complement processing time improvements of 1.3X to 6.3X and memory footprint reductions of over 99% for large datasets. These improvements substantially enhance the scalability of 3D urban data processing, enabling applications that were previously constrained by computational resources.

The bidirectional conversion capability between CityJSON and CityJSONSeq ensures that organizations can adopt the streaming format without sacrificing compatibility with existing tools and workflows. The efficiency of this conversion process facilitates flexible data management strategies that leverage each format's strengths according to specific use case requirements.

As urban models continue to grow in scale and complexity, streaming-enabled formats like CityJSONSeq will play an increasingly important role in making this data accessible and usable across diverse applications. The format represents a significant advancement in 3D geospatial data engineering, balancing expressive power with practical computational characteristics. Future work will focus on extending these concepts to dynamic urban data and enhancing interoperability across the geospatial ecosystem.

References

[1] H. Ledoux, K. A. K. Ohori, B. Dukai, A. Labetski, and S. Vitalis, "Cityjson: a compact and easy-to-use encoding of the citygml data model," *Open Geospatial Data, Software and Standards*, vol. 4, no. 4, pp. 1–12, 2019.

[2] "Ogc city geography markup language (citygml) part 1: Conceptual model standard," Open Geospatial Consortium, Tech. Rep. 20-010, 2021.

[3] "Cityjson community standard 1.0," Open Geospatial Consortium, Tech. Rep. 20-072r2, 2021.

[4] C. Praschl and O. Krauss, "Extending 3d geometric file formats for geospatial applications," *Applied Geomatics*, vol. 16, no. 1, pp. 161–180, 2023.

[5] "Cityjson community standard 2.0," Open Geospatial Consortium, Tech. Rep. 20-072r5, 2023.

[6] N. Williams, "Javascript object notation (json) text sequences," RFC 7464, Tech. Rep., 2015.

[7] S. Gillies, "Geojson text sequences," RFC 8142, Tech. Rep., 2017.

[8] T. Bray, "The javascript object notation (json) data interchange format," RFC 8259, Tech. Rep., 2017.

[9] H. Butler, M. Daly, A. Doyle, S. Gillies, T. Schaub, and S. Hagen, "The geojson format," RFC 7946, Tech. Rep., 2016.

[10] M. Isenburg, P. Lindstrom, S. Gumhold, and J. Snoeyink, "Large mesh simplification using processing sequences," in *IEEE Visualization, 2003. VIS 2003.*, 2003, pp. 465–472.

[11] M. Isenburg, Y. Liu, J. R. Shewchuk, J. Snoeyink, and T. Thirion, "Generating raster dem from mass points via tin streaming," in *Geographic Information Science*. Springer, 2006, pp. 186–198.

[12] R. Peters, B. Dukai, S. Vitalis, J. van Liempt, and J. Stoter, "Automated 3d reconstruction of lod2 and lod1 models for all 10 million buildings of the netherlands," *Photogrammetric Engineering and Remote Sensing*, vol. 88, no. 3, pp. 165–170, 2022.

[13] Powalka, C. Poon, Y. Xia, S. Meines, L. Yan, Y. Cai, G. Stavropoulou, B. Dukai, and H. Ledoux, "Recent advances in 3d geoinformation science," in *Proceedings of the 18th 3D Geoinfo Conference*. Springer, 2024, pp. 781–796.

[14] G. Agugiaro, A. Zwamborn, C. Tigchelnar, E. Matthijssen, C. Leon-Sanchez, F. van der Molen, and J. Stoter, "On the influence of party walls for urban energy modelling," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 48, no. 4/W5-2022, pp. 9–16, 2022.