S & M 2016

Algorithm for Removing Duplicate Vertices from 3D Spatial Objects to Improve Service Efficiency

Jihun Kang,[1*] Sewon Lee,[1] and Jihhyeok Park[2]

[1]Spatial Information Research Institute, Korea Land and Geospatial Informatix Corporation, Jeollabuk-do 55365, Korea
[2]Advance Technology Research Institute, LoDiCS, Seoul 04890, Korea

(Received June 19, 2019; accepted September 4, 2019)

Keywords: duplicate vertex, 3D spatial object, service efficiency, 3DS, SPOTool

The demand for highly detailed 3D spatial data is continuously increasing with the development of information technology. However, these 3D spatial data have a large capacity, which is a major cause of service degradation. To solve this problem, an algorithm for reducing the 3D data size by removing duplicate vertices, which contain information that is not used for 3D building object visualization, was proposed. Furthermore, prototype software was developed to verify this algorithm. The results of duplicate point elimination were similar to or slightly better than the features of professional 3D graphics software. The proposed technology is expected to be useful because it reduces the data capacity while maintaining the quality of 3D spatial information data.

1. Introduction

The 3D spatial information services market is exhibiting very rapid and continued growth around the world. Particularly in the case of a smart city, a representative 3D service field, MarketsandMarkets projected the 0.3 trillion dollar market in 2018 to grow to 0.7 trillion dollars by 2023,(1) and Frost & Sullivan projected it to grow into a market worth 2.0 trillion dollars by 2025.(2) In line with this trend, the Government of the Republic of Korea [Ministry of Land, Infrastructure and Transport (MOLIT)] has resolved to promote the creation of smart city industrial ecosystems; specifically, it has announced a master plan to operate a world-class national pilot system. To implement the "digital twin", which constitutes the core of a smart city, various research and development projects have been proposed to implement the range of technologies needed to build virtual 3D spatial information identical to a real-world city model.

In order to create a successful digital twin, realistic high-precision 3D data that can be used by linking information collected from various sensors are essential. Furthermore, information services should be continuously improved to make relevant data sets more applicable to diverse areas such as urban planning, visualization, spatial analysis, disaster management, films, and games; therefore, it is essential to improve the service efficiency. In this regard, the level

*Corresponding author: e-mail: kangdaejang@lx.or.kr
https://doi.org/10.18494/SAM.2019.2491

ISSN 0914-4935 © MYU K.K.
https://myukk.org/

of detail (LOD) is a widely used 3D data structure.(3) LOD is a technique for hierarchically structuring data so that only the necessary features of spatial objects are selectively displayed according to the distance on the map; this facilitates large-volume data services. However, these technical features alone cannot satisfy the continuously growing user demand for highly detailed 3D data and high-quality service.

As such, there is a need to develop technologies for improving the production efficiency of high-resolution 3D data. Existing spatial information technologies have focused on producing true-to-life 3D data, with the main research focus on automatic 3D modeling using high-resolution aerial photogrammetry or aerial light detection and ranging (LiDAR) data.(4,5) However, this approach is cost-intensive and time-consuming. To address this limitation, significant research efforts have been devoted to using existing data for efficient 3D data generation; for example, 3D building modeling methods using 2D scanned plans or aerial imagery products provided by Internet map services,(6,7) or images captured from unmanned aerial vehicles instead of expensive aerial photogrammetry or aerial LiDAR data.(8)

It is also necessary to develop technologies for improving the 3D data service efficiency. To provide city-scale 3D data services, various technologies are required to process large-volume spatial data. To address this need, researchers have proposed various methods to reduce the data volume, such as setting the LOD-based data visualization level and reducing the numbers of edges and vertices that constitute the 3D objects.(9) For example, the Korean government launched the "V-World" service in 2013 and has since provided 3D geospatial imagery across the country (and beyond) and 3D spatial views of urban objects (buildings, streets, and subways) in Seoul and other metropolitan areas. It is a government-run spatial information open platform that provides maps and various types of public spatial data. Since the service was released in Korea, a number of follow-up studies have been conducted with a focus on "light-weight" or "simplification" for easier and more accurate 3D data services.(10,11) For example, Kim *et al.* (i) developed a method of minimizing building-related data and standardizing the 3D building data format to ensure data service efficiency and interoperability,(12) and (ii) conducted an evidence-based study to improve the V-World service quality by analyzing the update status and errors of 3D models and attribute data by verifying them using a publicly available cadastral registry.(13) Similar studies conducted in the construction sector have sought to improve the screen visualization efficiency of building information modeling (BIM) data.(14,15) This is associated with a light-weight algorithm considering BIM characteristics, which is indispensable for ensuring the smooth operation of large-scale BIM data on the geographic information system (GIS) platform.

Thus, with increasing demand for 3D data, various 3D-related technologies have been studied by different approaches. However, the majority of these studies have focused on developing techniques for producing and constructing highly detailed 3D data, and relatively little effort has been spent on improving the 3D data structure to provide better services to users. Generally, providing users with high-resolution 3D models is time-consuming and cost-intensive on account of their large data volume. Therefore, to provide general users with accurate data for a realistic 3D experience, it is important to not only produce highly detailed 3D data, but also improve the data retrieval speed by reducing the data volume.

The purpose of this study is to develop a data volume minimization algorithm that removes duplicate vertices that are unnecessary for visualizing a 3D spatial object, thereby improving the efficiency of 3D data services. Test data were generated using the aerial imagery-based 3D modeling software PLW Modelworks. An algorithm used to remove duplicate vertices from aerial-imagery-based highly detailed 3D objects was defined using these data, and a prototype was implemented to analyze its performance. Duplicate vertex removal performance was verified by comparing and analyzing similar functions using two specialized computer graphics software tools. The software developed in this study outperformed the other two software tools according to its duplicate vertex removal rate. Furthermore, we conclude that duplicate vertex removal is a key process following 3D building modeling.

2. Data Structure of 3D Spatial Object Data

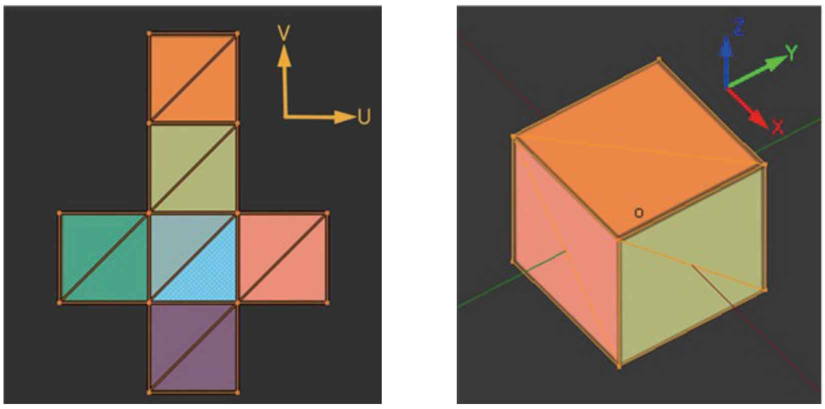
Aerial-imagery-based 3D object data are generally produced in the Autodesk 3DS format. The 3DS format is a binary file format that represents 3D geometric meshes made of triangles as the minimum unit. The component units of this 3DS format are a vertex, a face, a mesh, and a file, which are used on the basis of texture, as shown in Table 1. By design, each file of this 3DS format can have multiple meshes and textures; however, most 3D spatial data consist of one mesh and one texture.

In general, a 3D rendering system registers each object's vertex and texture coordinates separately and defines each list as an index. An index usually specifies the vertex and texture coordinates of an object with a float or double data type using the integer data type. An index thus defined can replace the repeatedly used corresponding vertex and texture coordinates, reducing the size allocated to the object data. Coordinate data are stored by using either two separate indices for vertex and texture coordinates or one index for the combined vertex and texture coordinates. A widely used format for the former is the OBJ format. The structure of the 3DS format data used in this study is the latter type of index storage.

For example, the 3D object data of a cube can be stored in 3DS format as follows: A cube is composed of vertex coordinates (X, Y, Z), which represent the eight vertices in a 3D space, and 14 texture coordinates (U, V), which correspond to the 14 corners matching their respective vertices when laid out flat in the 2D texture space from the 3D space of the cube (Fig. 1, Table 2). To store the data set in 3DS format using an index, 23 texture and 23 vertex coordinate indices are generated and stored (Table 3). The 12 triangle faces constituting the 3D object are stored on the basis of the coordinate indices (Table 4).

Table 1
3DS format component.

Component	Details
Vertex	The point that constitutes the model, expressed in 3D coordinates
Face	A triangle consisting of three vertices; the smallest unit of the model
Mesh	A group of multiple faces
File	A group of multiple meshes
Texture	The image corresponding to the position of the vertex (reference element in 3DS format)



(a) (b)

Fig. 1. (Color online) A cube laid out flat in (a) 2D texture space (U, V) and (b) 3D object space (X, Y, Z).

Table 2
Vertex and texture coordinates.

Vertex coordinates				Texture coordinates					
#	X	Y	Z	#	U	V	#	U	V
1	−1	−1	1	1	0.39	0.28	8	0.39	0.94
2	−1	−1	−1	2	0.61	0.28	9	0.39	0.06
3	1	−1	−1	3	0.61	0.50	10	0.61	0.06
4	1	−1	1	4	0.39	0.50	11	0.17	0.28
5	−1	1	1	5	0.39	0.72	12	0.17	0.50
6	1	1	1	6	0.61	0.72	13	0.83	0.28
7	1	1	−1	7	0.61	0.94	14	0.83	0.50
8	−1	1	−1						

Table 3
Vertex and texture coordinate indices of a 3D object.

#	X	Y	Z	U	V	#	X	Y	Z	U	V
1	−1	−1	1	0.39	0.28	13	−1	1	1	0.61	0.06
2	−1	−1	1	0.83	0.28	14	−1	1	1	0.61	0.28
3	−1	−1	1	0.39	0.06	15	−1	1	1	0.39	0.72
4	−1	−1	−1	0.39	0.28	16	1	1	1	0.61	0.50
5	−1	−1	−1	0.17	0.28	17	1	1	1	0.61	0.72
6	−1	−1	−1	0.61	0.28	18	1	1	−1	0.61	0.50
7	1	−1	−1	0.17	0.50	19	1	1	−1	0.39	0.50
8	1	−1	−1	0.61	0.50	20	1	1	−1	0.61	0.94
9	1	−1	−1	0.39	0.50	21	−1	1	−1	0.39	0.28
10	1	−1	1	0.83	0.50	22	−1	1	−1	0.61	0.28
11	1	−1	1	0.39	0.50	23	−1	1	−1	0.39	0.94
12	1	−1	1	0.39	0.72	—	—	—	—	—	—

Table 4
Vertex and texture coordinates of a polygon.

#	P1	P2	P3
1	1	6	8
2	1	8	11
3	15	17	20
4	15	20	23
5	3	13	22
6	3	22	4
7	5	21	19
8	5	19	7
9	9	18	17
10	9	17	12
11	14	2	10
12	14	10	16

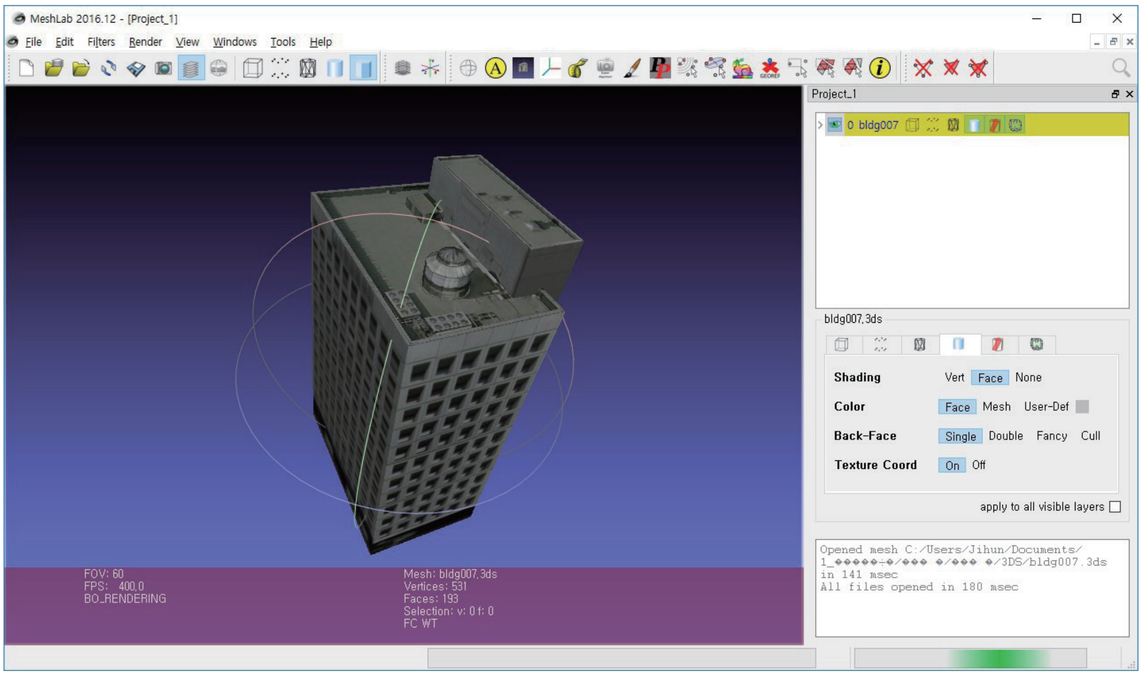
3. Methods to Remove Duplicate Vertices

In principle, removing duplicate vertices means removing the vertex coordinates with exactly the same coordinate values. Functionally, however, it involves removing the texture coordinates with exactly the same coordinate values. In the removal method where two separate indices are used for the vertex and texture coordinates, as explained in the previous section, the duplicate vertices should be removed in their respective coordinates. In the removal method where one index is used for the combined vertex and texture coordinates, the duplicate points exist in both vertex and texture coordinates. It is often the case that 3D object optimization functions, including the removal of duplicate vertices, are embedded in specialized computer graphics software; for example, open source software such as Blender, which can construct and edit 3D objects, and MeshLab, which provides diverse methods and algorithms for analyzing and transforming 3D objects.

3.1 Review of existing techniques

MeshLab is a 3D mesh processing software system developed by the Institute of Information Science and Technology of the Italian National Research Council. It is provided under the General Public License (GPL), available for various operating systems such as Windows, Mac, and Linux, and is broadly used in academic and research work (Fig. 2). Blender is also a general public license (GNU) GPL open source 3D creation software tool developed by the Blender Foundation and supported by the Dutch government. It provides business solutions that comprehensively support the 3D creation process encompassing modeling, rigging, animation, and simulation, and is widely used in commercial fields (Fig. 3).

Both tools have diverse in-built functions for 3D object manipulation, some of which can remove duplicate vertices. MeshLab provides the "Cleaning and Repairing" function under the menu "Filters", with which duplicate vertices and texture coordinates can be eliminated by invoking the filter "Remove Duplicated Vertices." Blender provides the function of removing



```
FOV: 60
FPS:   400.0
BO_RENDERING

Mesh: bldg007.3ds
Vertices: 531
Faces: 193
Selection: v: 0 f: 0
FC WT
```

Shading: Vert [✓ Face] None
Color: [✓ Face] Mesh User-Def
Back-Face: [✓ Single] Double Fancy Cull
Texture Coord: [✓ On] Off
apply to all visible layers ☐

```
Opened mesh C:/Users/Jihun/Documents/
1_◆◆◆◆◆÷◆/◆◆◆ ◆/◆◆◆ ◆/3DS/bldg007.3ds
in 141 msec
All files opened in 180 msec
```

Fig. 2. (Color online) Basic screen layout of MeshLab.

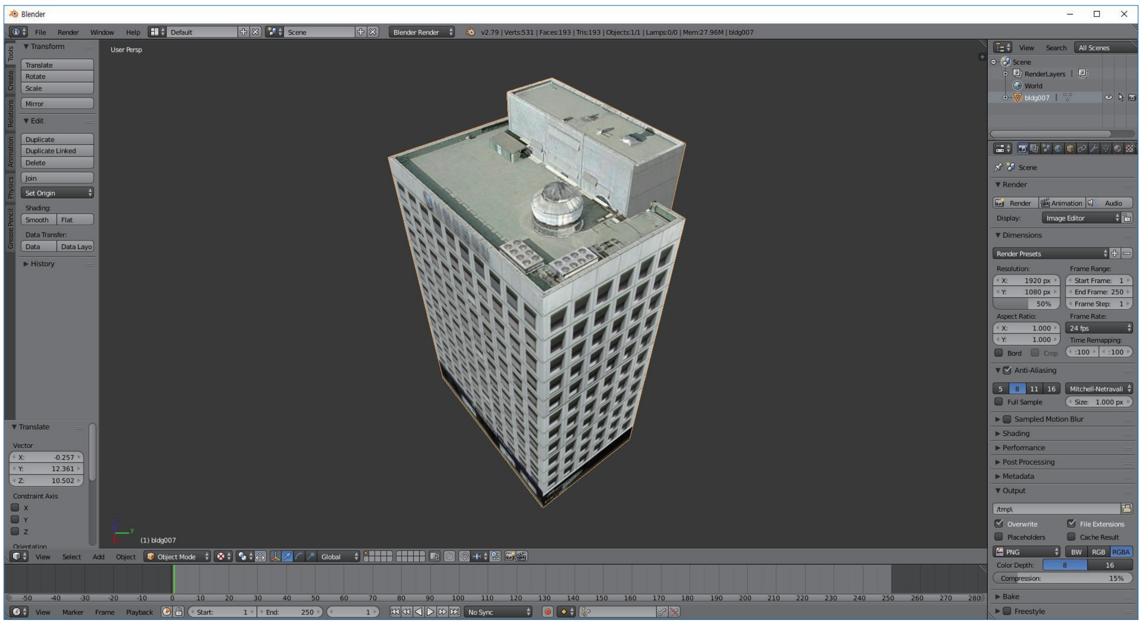


Fig. 3. (Color online) Basic screen layout of Blender.

duplicate texture coordinates under the menu "Remove Doubles UV" and removing duplicate texture and vertex coordinates under the menu "Remove Doubles". Note that invoking "Remove Doubles" automatically activates the function "Remove Doubles UV".

Although these software systems are verified techniques, the related functions are complicated to use and cannot be employed separately for redundant vertex removal because they are built-in functions. More importantly, it is impossible to process a data set as a whole, which makes it infeasible to use existing software tools for processing data in the field of spatial information, in which large-volume spatial object data sets are created and updated. To address this problem and ensure the efficient processing of 3D object data created in 3DS format, we define a duplicate vertex removal algorithm and develop the related software.

3.2 Development of a method to remove duplicate vertices of multi-structured data

To use the diverse duplicate vertex removal functions of the software tools presented above, it is necessary to convert the 3D data of different formats into the format required by each software system. The algorithm developed in this study is optimized for the 3DS format in which 3D objects are created and stored. Its main focus is detecting the duplicate vertices sharing the same coordinate combinations occurring in the 3DS format structure and reconstructing the faces linked to the corresponding vertices. The details of the proposed method are as follows.

First, a data structure, in which the vertex and texture coordinates (X, Y, Z+U, V) can be stored as one record, is constructed, and coordinate and mapping lists are compiled. The coordinate list manages the record index and the mapping list is used to map the index in the original vertex list and the index in the new vertex list. All the vertices of the object are then sequentially inspected, thereby checking all vertex and texture coordinates, so that the coordinate list can be completed without duplicate coordinate values. That is, each vertex is checked against the list and discarded or added to the list depending on whether or not it shares the same coordinate values with any of the vertex and texture coordinates in the list. The mapping list is updated accordingly, i.e., it is updated using information about the indices related to the vertices discarded from the original list and added to the new list.

The coordinate list free of duplicate vertices and the mapping list are completed along with the completion of the sequential inspection of all vertices. The process of duplicate vertex removal is completed by inspecting each face and updating the indices of the related vertices and creating new original data using the completed mapping list. Figure 4 shows the pseudocode presented in this study for duplicate vertex removal. This algorithm is for the removal of duplicate points in 3DS format data characterized by using the same index for vertex and texture coordinates. If it is to be applied to other 3D formats that use indexes separately, the vertex and texture coordinate lists must be removed separately.

To determine whether two records share the same vertex and texture coordinates, individual sets of coordinate values (X, Y, Z+U, V) are checked, whereby checking for the same coordinate values corresponds to checking for the same floating points. In general, if the absolute value of the difference between two floating point numbers is lower than the reference value (e.g., epsilon: the smallest number represented by the related data type), then these two floating point numbers are considered to be equal. A negligible difference in value can be used as the reference value, i.e., if the unit of vertex coordinates of the 3D spatial object data is meters, a value of 0.001 m (1 mm) can be used as the reference value. Accordingly, the number of vertices can be gradually reduced by expanding the number of duplicate vertices to be reduced on the basis of the set reference value.

```
SET empty vertex list to UniqVerts
FOR each Vert in entire vertex list
    IF Uniqverts not contains Vert THEN
        ADD Vert to UniqVerts
    END IF
END LOOP
FOR each Face in entire face list
    FOR each Vert in face
        EDIT index of Vert to new index in UniqVerts
    END LOOP
END LOOP
SET UniqVerts to entire vertex list
```

Fig. 4. Pseudocode for removing duplicate vertices.

Similarly, comparing the texture coordinates is the same as comparing the floating point numbers, whereby the texture coordinates are inverse-normalized to the pixel coordinates depending on the texture size. That is, texture coordinates are values obtained by normalizing the pixel coordinates in the texture space to values between 0 and 1. Accordingly, the number of vertices can be gradually reduced by applying the value of 0.01 (= 0.01 pixels) as the reference value.

In this study, the floating point number was compared on the basis of the minimum value of the data type provided by the software language, which was set as the default value, with the option for specifying different values as the need arises. When determining whether a vertex shares the same coordinate values with a vertex in the list, the number of operations is proportional to the number of vertices in the list. The number of operations can be substantially reduced by using an appropriate space index. In this study, the number of operations was reduced by using the distance of each vertex from the center of the 3D spatial object data as the index of each vertex.

4. Comparison of Duplicate Vertex Removal Outcomes

The duplicate vertex removal performance of the proposed algorithm was tested with Spatial Object Tool (SPOTool), an in-house prototype software tool that employs C++ as the development language. SPOTool was configured to be implemented in the console environment without GUI. For other functions related to duplicate vertex removal, namely, 3DS file input and output and spatial index, lib3ds and gdal open source libraries were used.

As the test data, 3DS-format building objects modeled with PLW Modelworks 3D object modeling software were used. The test objects were 12 buildings located in the Yeouido area of Seoul. Various building types, from simple bungalows to high-rise structures, were selected to diversify the number of vertices (Table 5). The total number of vertices of the 12 data sets amounted to 16789 and the total file size was 444.81 kB.

To verify the performance of the proposed algorithm, MeshLab and Blender were used for comparison testing. The test was conducted in the following order: load the test data set into the software, remove duplicate vertices, save the data set in 3DS format, reopen the stored file, and count the number of remaining vertices. Table 6 presents the results of the duplicate vertex removal testing on 12 data sets comparing the performance levels of three tools (SPOTool, MeshLab, and Blender).

The performance comparison of the three software tools yielded similar overall vertex removal rates (Table 7); however, SPOTool based on the algorithm proposed in this study showed a slightly higher average removal rate. MeshLab and Blender employed their respective additional processes for optimization, yielding different removal outcomes, with more reductions than simply the removal of duplicate vertices. SPOTool also added a process to enhance the removal rate. Specifically, some of the 3D spatial object data contain records of texture coordinates lying outside the texture space containing faces without texture allocation. SPOTool was configured to process such data as duplicate vertices after providing them with random values, further reducing the number of vertices. Furthermore, after the first run of

Table 5
Test data.

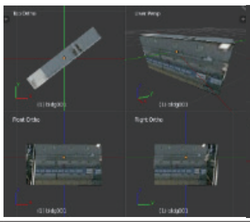
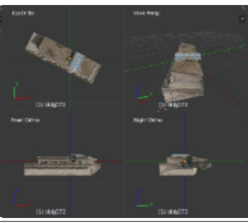
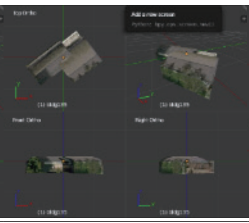
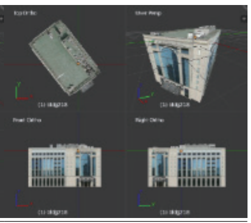
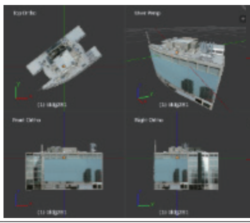
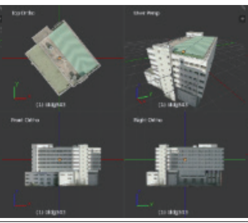
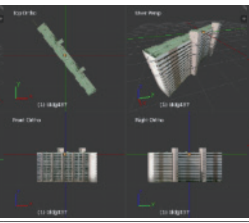
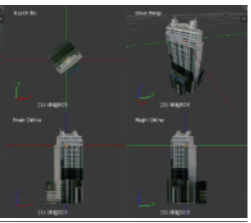
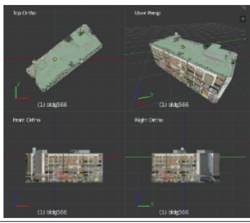
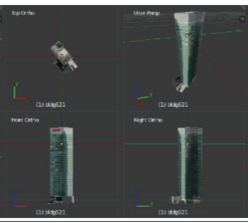
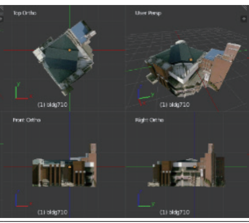
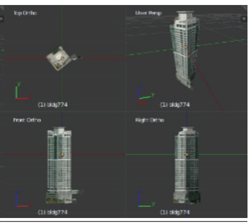
File name	bldg001	bldg072	bldg135	bldg218
No. of vertices	24	270	24	1816
File size (kB)	1.02	7.57	1.02	47.5
Image				
File name	bldg281	bldg343	bldg437	bldg503
No. of vertices	4722	1451	1888	1809
File size (kB)	126	37.3	50.1	48
Image				
File name	bldg566	bldg621	bldg710	bldg774
No. of vertices	394	2032	1149	1210
File size (kB)	10.5	53.6	30.4	31.8
Image				

Table 6
Number of vertices remaining after removal of duplicate vertices.

	Original data	SPOTool	MeshLab	Blender
bldg001	24	14	15	14
bldg072	270	202	207	210
bldg153	24	14	14	14
bldg218	1816	1148	1148	1148
bldg281	4722	3519	3520	3519
bldg343	1451	778	778	774
bldg437	1888	1354	1355	1355
bldg503	1809	1191	1193	1193
bldg566	394	195	196	195
bldg621	2032	1379	1380	1379
bldg710	1149	787	791	791
bldg774	1210	772	775	772

Table 7
Comparison of vertex removal efficiency for different software systems.

	Original data	SPOTool	MeshLab	Blender
No. of vertices	16789	11353	11372	11364
No. of vertices removed	—	5436	5417	5425
Vertex removal rate	—	32.38	32.27	32.31

duplicate vertex removal, the vertices constituting faces without an area as a result of duplicate vertex removal were additionally eliminated.

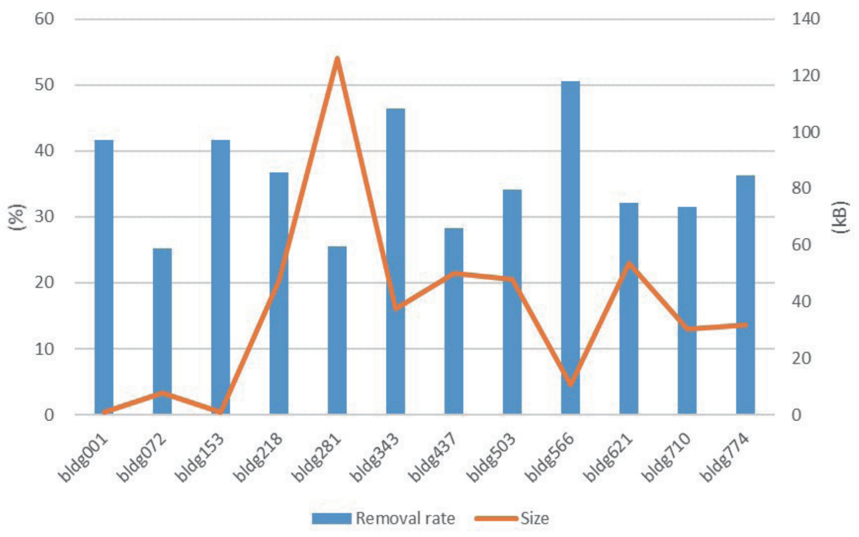


Fig. 5. (Color online) Duplicate vertex removal rates for objects of different sizes.

By comparing the number of duplicate point eliminations, the same number of duplicate points were removed in the three software programs for the bldg153 and bldg218 objects. However, in the other 10 objects, SPOTool removed one or more redundancy points compared with the other two software programs (Table 6). By comparing the results of eliminating duplicate points for all 12 objects, SPOTool showed the performance of removing 19 more duplicate points than MeshLab and 11 more than Blender (Table 7).

Additionally, considering that the object size is proportional to the number of vertices, the correlation between the object size and the duplicate vertex removal rate was analyzed. Although a larger number of vertices were removed from a larger object, given the proportionally larger number of vertices in a larger object, the comparison of removal rates with respect to the ratio between the numbers of vertices before and after duplicate vertex removal revealed that the removal rate does not correlate with the object size (Fig. 5). This suggests that the occurrence of duplicate vertices does not follow a specific mechanism depending on certain conditions but is a random phenomenon.

5. Conclusions

The visualization of 3D object data, which can implement the realistic rendering of objects as a result of recent advancements in computer graphics technologies, requires a cost-intensive large-volume storage space and a broad network bandwidth. Thus, with the continuously increasing demand for 3D data, there is a growing need to improve the efficiency of data production and service structures. In the spatial information market, 3D data should meet the requirements for the accurate representation of collective 3D objects based on geospatial information, such as buildings, streets, and facilities, rather than provide a realistic rendering of the objects themselves, as required in the fields of animation and gaming. It is therefore essential to develop technologies that consider service efficiency so that users can easily and continuously check the collective 3D objects that constitute a virtual city space.

In this study, an algorithm for automated duplicate vertex removal from 3D object models was proposed in an attempt to improve the service efficiency of spatial information services by reducing the volume of 3D data. The test data sets were extracted from the data provided by the Korean 3D map service "V-World" in 3DS format for urban modeling based on aerial imagery. In 3DS format, duplicate vertices should be checked among the combined vertex and texture coordinates (X, Y, Z+U, V). Accordingly, the duplicate vertex removal algorithm was developed in a way to remove duplicate vertices by identifying data points with the same vertex (X, Y, Z) and texture (U, V) coordinate values, and reconstruct the faces linked to the corresponding vertices. To test the duplicate vertex removal performance of the proposed algorithm, we developed SPOTool, an in-house prototype software tool, and performed comparison testing using SPOTool and two widely used 3D graphics software tools, MeshLab and Blender.

3D format object models of 12 buildings located in the Yeouido area of Seoul were used as test data. In the comparison testing, the software developed in this study showed a mean duplicate vertex removal rate of 32.38%, outperforming the other two software tools. For all 12 objects, SPOTool removed the same or slightly higher numbers of duplicate vertices. However, the large standard deviations of the removal rate distribution, which ranged from 23 to 51% (more than double), do not allow for the generalization of the duplicate vertex removal rate. Moreover, although a larger number of vertices were removed from larger objects, given the proportionally larger number of vertices in a larger object, no correlation was observed between the file size and the removal rate. This suggests that duplicate vertices occur in an unpredictable, random manner and are a common phenomenon affecting most object models. To conclude, duplicate vertex removal is an indispensable process after conducting 3D building modeling. The process of removing such unnecessary information can make the size of the 3D data small, which is expected to slightly improve the service speed.

3D data generation and service delivery are cost-intensive and time-consuming and therefore subject to many limitations. To overcome such limitations, there is a need for continuous research efforts to develop technologies that can meet the requirements of the constantly growing 3D spatial information services market, specifically to improve the service efficiency by minimizing the 3D spatial information data volume.

Acknowledgments

This research was supported by a grant (19DRMS-B147287-02) from the development of customized realistic 3D geospatial information update and utilization technology based on consumer demand, funded by the Ministry of Land, Infrastructure and Transport of the Korean government.

References

1 MarketsandMarkets: https://www.marketsandmarkets.com/Market-Reports/smart-cities-market-542.html (accessed August 2019).

2 Frost & Sullivan: https://ww2.frost.com/news/press-releases/frost-sullivan-experts-announce-global-smart-cities-raise-market-over-2-trillion-2025/ (accessed August 2019).
3 F. Biljecki, H. Ledoux, and J. Stoter: Comput. Environ. Urban Syst. **59** (2016) 25. http://doi.org/10.1016/j.compenvurbsys.2016.04.005
4 A. P. McClune, J. P. Mills, P. E. Miller, and D. A. Holland: Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XLI-B3, 2016 XXIII ISPRS Congress, 641–648. https://doi.org/10.5194/isprs-archives-XLI-B3-641-2016
5 E. J. Yoo, S. G. Yun, and D. C. Lee: J. Korean Soc. Surv. Geode. Photo. Carto. **30** (2012) 1. https://doi.org/10.7848/ksgpc.2012.30.1.001
6 L. Gimenez, S. Robert, F. Suard, and K. Zreik: WIT Trans. Built Environ. **149** (2015) 437. https://doi.org/10.2495/BIM150361
7 D. C. Lee: J. Korean Soc. Surv. Geode. Photo. Carto. **29** (2011) 687. https://doi.org/10.7848/ksgpc.2011.29.6.687
8 J. Fernández-Hernandez, D. González-Aguilera, P. Rodríguez-Gonzálvez, and J. Mancera-Taboada: Archaeometry **57** (2015) 128. https://doi.org/10.1111/arcm.12078
9 J. Y. Na and C. H. Hong: Spat. Inf. Res. **21** (2013) 1. https://doi.org/10.12672/ksis.2013.21.6.001
10 B. Kim, M. Jeong, and D. Shin: Spat. Inf. Res. **24** (2016) 323. https://doi.org/10.1007/s41324-016-0034-x
11 H. Jang, D. Kim, J. Kim, and I. Jang: Spat. Inf. Res. **24** (2016) 367. https://doi.org/10.1007/s41324-016-0038-6
12 H. D. Kim, J. H. Kang, and H. J. Kim: J. Korean Soc. Geosp. Inf. Syst. **25** (2017) 63. https://doi.org/10.7319/kogsis.2017.25.1.063
13 J. T. Kim, S. H. Yi, J. I. Kim, and S. W. Bae: J. Korean Soc. Geosp. Inf. Syst. **22** (2014) 137. https://doi.org/10.7319/kogsis.2014.22.3.137
14 J. E. Kim, C. H. Hong, and S. D. Son: WIT Trans. Built Environ. **149** (2015) 355. https://doi.org/10.2495/BIM150301
15 J. E. Kim, T. W. Kang, and C. H. Hong: J. Korea Acad. Ind. Coop. Soc. **15** (2014) 5333. https://doi.org/10.5762/KAIS.2014.15.8.5333

About the Authors

Jihun Kang received his B.E. and M.S. degrees from Sungkyunkwan University, Korea, in 2007 and 2009, respectively. Since 2012, he has been a senior researcher at Korea Land and Geospatial Informatix Corporation, Korea. His research interests are in 3D GIS, geospatial data analysis, and natural hazard analysis. (kangdaejang@lx.or.kr)

Sewon Lee received his Ph.D. degree from Seoul National University, Korea, in 2015. Since 2015, he has been a principal researcher at Korea Land and Geospatial Informatix Corporation, Korea. His research interests are in 3D GIS, geospatial data analysis, urban planning, and urban spatial structure. (leesewon@lx.or.kr)

Jinhyeok Park received his B.S. degree from Hongik University, Korea, in 2008. Since 2009, he has been a senior research engineer at LoDiCS Corporation. His research interests are in remote sensing, distributed environment, and 3D rendering systems. (daniel@lodics.com)