INDOOR SPATIAL DATA CONSTRUCTION FROM TRIANGLE MESH

Dongmin Kim, Azamat Bolat, Ki-Joune Li*

Dept. of Computer Science&Engineering, Pusan National University, Kumjeong-Gu, 46241, Busan, South Korea
- (dongmin.kim, azamat.bolat, lik)@pnu.edu

Commission IV, WG IV/4

KEY WORDS: Triangle Mesh, Mesh Simplification, Solid, Indoor Space, Indoor Data, IndoorGML

ABSTRACT:

The 3D triangle mesh is widely used to represent indoor space. One of widely used methods of generating 3D triangle mesh data of indoor space is the construction from the point cloud collected using LIDAR. However, there are many problems in using generated triangle mesh data as a geometric representation of the indoor space. First, the number of triangles forming the triangle mesh is very large, which results in a bottleneck of the performance for storage and management. Second, no consideration on the properties of indoor space has been done by the previous work on mesh simplification for indoor geometric representation. Third, there is no research to construct indoor spatial standard data from triangle mesh data. For resolving these problems, we propose the a method for generating triangular mesh data for indoor geometric representation based in the observations mentioned above. First this method removes unnecessary objects and reduces the number of surfaces from the original fine-grained triangular mesh data using the properties of indoor space. Second, it also produces indoor geometric data in IndoorGML - an OGC standard for indoor spatial data model. In experimental studies, we present a case study of indoor triangle mesh data from real world and compare results with raw data.

1. INTRODUCTION

3D indoor spatial data can be created from a variety of methods. Recently, in order to collect 3D indoor data many studies build point cloud data using LIDAR scanning technology, (Choi et al., 2013). Simple visualization with raw point cloud data is possible, but it is difficult to perform queries about indoor space and to store and manage indoor space data due to its geometric incompleteness. Therefore, instead of point cloud, we need to represent the indoor spatial data with 3D Boundary Representation, B-rep, (Stroud, 2006) geometry data to query indoor space.

We can group points in point cloud in an appropriate way to generate 3D B-rep geometry data consisting of triangle meshes. However, querying this data will increase the computational complexity exponentially. It happens due to composition of numerous triangles, even if it is a single wall. So we have reduce the number of triangles. This could be accomplished through several means. The first method is to sample proper points from the raw data and reconstruct the triangle mesh using sampled points. The second method is to create simple mesh data through previous mesh simplification work. However, these methods are not suitable for constructing indoor space data, because the original geometric information is distorted or properties of the indoor spatial data are not considered.

We construct indoor space data from raw triangle mesh by using properties of indoor space. Several considerations need to be taken into account at the outset. Firstly, the surface of indoor space data should be one of walls, floors, ceiling surfaces (architectural elements) surrounding indoor space, or one of surfaces (non-architectural elements) surrounding furniture in indoor space. We must determine the surfaces, to which each triangle mesh belongs to. Secondly, it is important to distinguish between architectural and non-architectural elements in the constructed point cloud. The method presented in (ChangHyun et al., 2017) creates point cloud data by exploiting elements differences. (ChangHyun et al., 2017) suggests how to divide objects in indoor space into architectural and non-architectural elements. In triangle mesh data constructed from this point cloud, architectural and non-architectural elements do not share boundaries with each other. And thirdly, the points in cloud data collected in indoor space are valid only in the corresponding indoor space. It means the data is not representation of outdoor or other indoor space. Therefore, triangular mesh data generated from point cloud collected in indoor space is also valid only in the same indoor space. We construct indoor spatial data from triangular mesh data considering these observations.

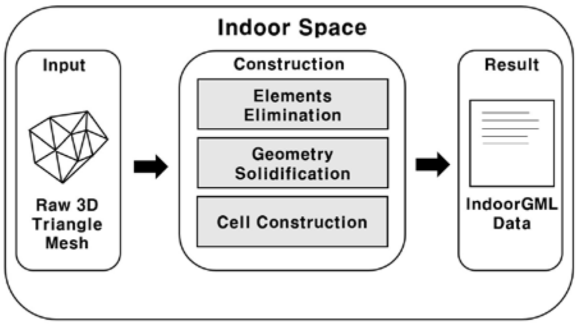


Figure 1. Problem definition

The following approaches were used to build indoor spatial data:

- We merge coplanar surfaces into one surface.
- We construct triangular mesh graph to distinguish architectural and non-architectural elements of interior space.

*Corresponding author

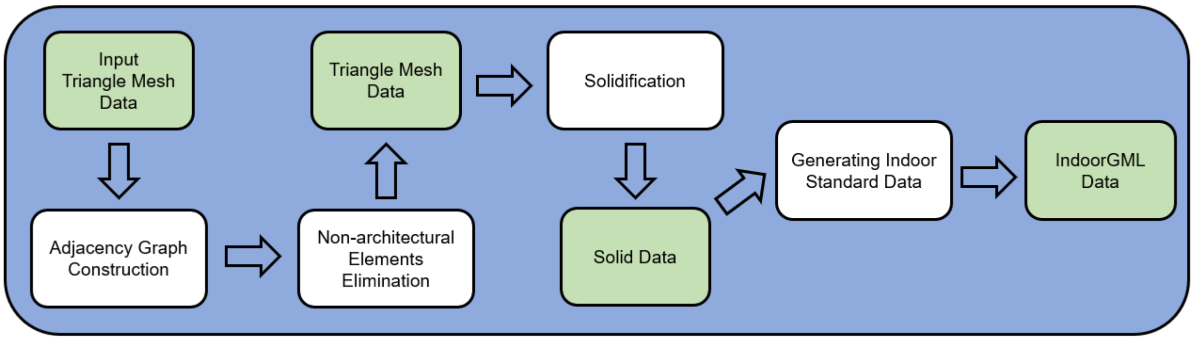

Input Triangle Mesh Data
Adjacency Graph Construction
Non-architectural Elements Elimination
Triangle Mesh Data
Solidification
Solid Data
Generating Indoor Standard Data
IndoorGML Data

Figure 2. Data change and conversion procedure

- We handle noise that can occur during indoor space data collection.
- We create standard data in indoor spatial data model - OGC IndoorGML.

In experiment steps, we compare the original and result data. We show how the proposed method preserves the indoor properties during the simplification and production of indoor spatial data. In addition, we demonstrated the differences in results of method in construction process. Finally, we validate the generated IndoorGML through visualization.

The structure of the paper is as follows: Section II gives a brief overview of related work and the motivations of this study. Section III defines basic concepts. Section IV describes indoor spatial data properties. Sections V and VI describes the overall conversion and the specific process. Section VII presents the experimental results and analyzes the performance. Section VIII concludes our work and discusses future work and open problems.

2. RELATED WORKS AND MOTIVATIONS

In terms of data size, we have to reduce the number of surfaces in the data and consequently need mesh simplification. (Cignoni et al., 1998) introduces several studies for mesh simplification and compares performance and accuracy. In based on controlled vertex/edge/face decimation method, (Schroeder et al., 1992) uses local operations on geometry and topology. Coplanar facets merging method is used in (Hinker and Hansen, 1993) and (Kalvin and Taylor, 1996). (DeCoro and Tatarchuk, 2007) proposed general-purpose data structure designed for streaming architectures called the probabilistic octree for mesh simplification with GPU implementation.

Several mesh simplification studies are underway. However, these studies do not take into consideration the characteristics of the target objects, and therefore they do not properly to produce indoor spatial data. Mesh simplification studies mentioned above are too generic to apply on indoor spatial data. Therefore, it is necessary to study the properties of indoor space for the construction of indoor spatial data. We perform the simplification of triangular mesh more efficiently by using the characteristics of indoor space. The elimination of non-architectural elements of indoor space results in more indoor-meaningful products than in previous studies.

Additionally, there is no study that generates geometric data according to indoor spatial standard data model. When we deal with indoor spatial data, we need standard data model for the interoperability between services and reusability of indoor maps. There are standards, CityGML (Kolbe et al., 2005) and IndoorGML (Kang and Li, 2017), providing frameworks of standard data models for indoor spaces. (Ryoo et al., 2015) compared them with advantages and disadvantages. For utilization of cellular space model, we generate IndoorGML rather than CityGML, and we explain IndoorGML in detail in Section 3.4.

3. BASIC CONCEPT

Figure 2 shows the data change and conversion procedure of this work. First, an adjacent graph is constructed by using adjacent relationship of triangles in mesh data. The graph is used to remove non-architectural elements. Next, we perform solidification to construct solid data from triangle mesh data that contains only architectural elements. Finally, we generate IndoorGML from the solid data. For explanation of this procedure, we define and explain several concepts, Triangle mesh data, Adjacency relationship, Solid data, and IndoorGML.

3.1 Triangle mesh data:

The triangle mesh geometric representation is defined by data model in ISO 19107. Related geometries are defined in ISO 19107 (Spatial Schema) are as follows:

- A **GM_TriangulatedSurface** is a GM_PolyhedralSurface that is composed only of triangles(GM_Triangle). There is no restriction on how the triangulation is derived.
- A **GM_Triangle** is a planar GM_Polygon defined by 3 corners.
- A **GM_Polygon** is a surface patch defined by a set of boundary curves and an underlying surface to which these curves adhere. Curves are coplanar by default.
- A **GM_PolyhedralSurface** is a GM_Surface composed of polygon surfaces (GM_Polygon) connected along their common boundary curves.
- **GM_Surface** is the basis for 2-dimensional geometry. Unorientable surfaces such as the Mobius band are not allowed.

Figure 3 shows relationship of above models simply.

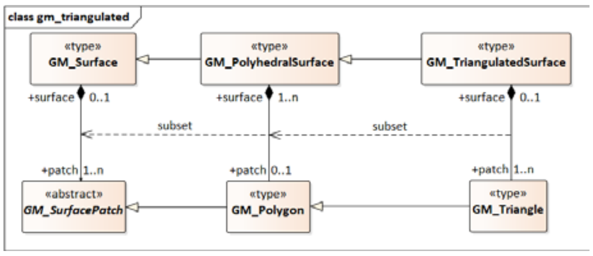


Figure 3. Relationship of GM_TriangulatedSurface and other types in ISO 19107

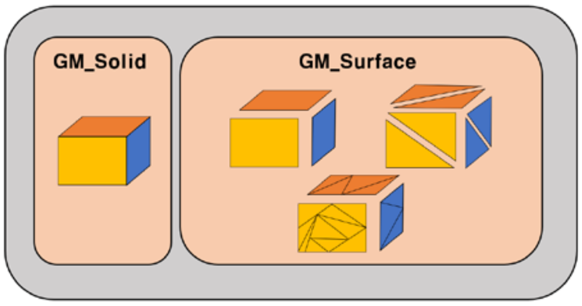


Figure 5. GM_Solid illustration

3.2 Adjacency relationship

In triangle mesh data, two triangles are considered to be **adjacent** when they have the opposite edge in relation to each other. In this study, edge and triangle are defined as follows:

$$\begin{aligned} Edge(a, b) &= \text{a directed edge with start vertex } a \text{ and end vertex } b \\ &= \text{opposite edge of } Edge(b, a). \\ I &= Edge(b, a) \end{aligned} \tag{1}$$

$$Triangle(a, b, c) = \{Edge(a, b), Edge(b, c), Edge(c, a)\} \tag{2}$$

We construct an **adjacent graph** using this adjacency relationship. In this graph, all triangles, in the triangle mesh data that we deal with, have only three neighboring triangles. Figure 4 shows example of triangle mesh and adjacency graph.

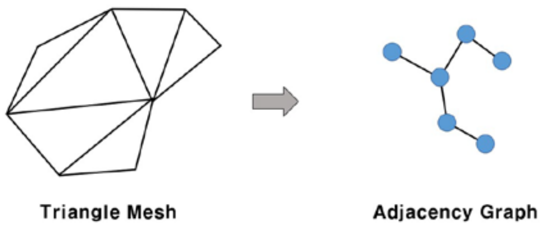


Figure 4. Construction adjacency graph from triangle mesh

3.3 Solid data

The solid geometric representation is defined in this study according the data model in ISO 19107. **GM_Solid** is the basis of 3-dimensional geometry and the extent of a solid is defined by the boundary surfaces. We can illustrate ISO 19107 models as shown in Figure 5. We build surfaces, which surround a solid as Figure 5.

3.4 IndoorGML document

The final goal of this study is to produce data in **IndoorGML** (Kang and Li, 2017), an international indoor standard data model. IndoorGML provides an open data model and XML schema for indoor spatial information. It aims to provide a common framework of representation and exchange of indoor spatial information. Indoor space can be a set of cells, which are defined as the smallest structural indoor unit space. We represent cells in IndoorGML using only the geometric representation. Figure 6 shows an example of an IndoorGML document.

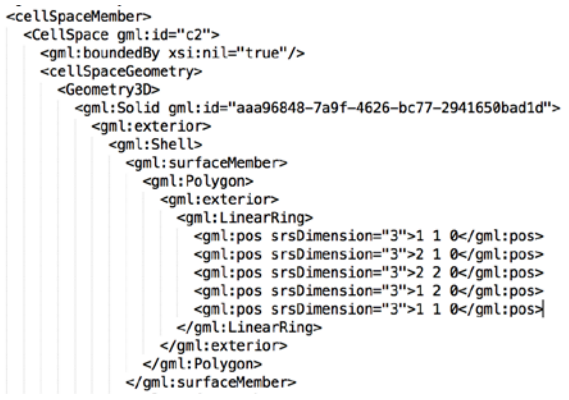

```
<cellSpaceMember>
  <CellSpace gml:id="c2">
    <gml:boundedBy xsi:nil="true"/>
    <cellSpaceGeometry>
      <Geometry3D>
        <gml:Solid gml:id="aaa96848-7a9f-4626-bc77-2941650bad1d">
          <gml:exterior>
            <gml:Shell>
              <gml:surfaceMember>
                <gml:Polygon>
                  <gml:exterior>
                    <gml:LinearRing>
                      <gml:pos srsDimension="3">1 1 0</gml:pos>
                      <gml:pos srsDimension="3">2 1 0</gml:pos>
                      <gml:pos srsDimension="3">2 2 0</gml:pos>
                      <gml:pos srsDimension="3">1 2 0</gml:pos>
                      <gml:pos srsDimension="3">1 1 0</gml:pos>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Polygon>
              </gml:surfaceMember>
```

Figure 6. Part of the IndoorGML document

4. INDOOR SPATIAL DATA PROPERTIES

In this section we discuss the properties of indoor space that we will consider for our approach. First, the surfaces surrounding a cell in IndoorGML data model are defined as follows:

$$\begin{aligned} WS &= \{s_w | s_w \text{ is a planar wall surface}\} \\ FS &= \{s_f | s_f \text{ is a planar floor surface}\} \\ CS &= \{s_c | s_c \text{ is a planar ceiling surface}\} \\ IS &= \{s_i | s_i \text{ is a planar indoor furniture surface}\} \end{aligned} \tag{3}$$

where WS, FS, CS, IS are disjoint sets.

Indoor spatial triangle mesh T satisfy the following conditions.

$$\text{For any triangle t in T, there is always one surface S, which is one of } WS, FS, CS, IS. \tag{4}$$

Architectural and non-architectural elements have the following characteristics.

- Architectural elements should be solid bounded by surfaces of WS, FS, CS.
- Non-architectural element should be solid only bounded by surfaces of IS.

Figure 7 shows these geometric properties of indoor spatial data.

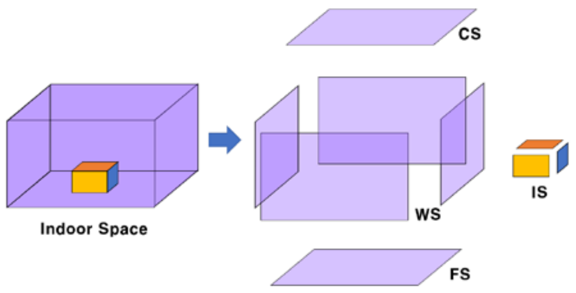


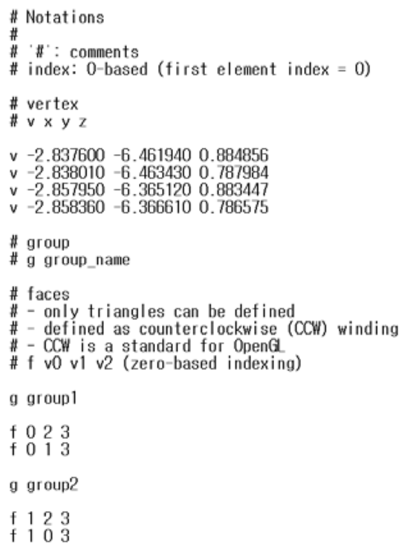
Figure 7. Surface types of indoor spatial data

5. CONSTRUCTION PROCEDURE

This section provides a detailed description of the building process introduced in Section 3. In the whole process we use the Computational Geometry Algorithms Library (The CGAL Project, 2018). CGAL is a software project that provides easy access to efficient and reliable geometric algorithms in a form of C++ library.

5.1 Input Triangle Mesh data

We load triangle mesh data stored in one of the file formats TMD (Triangle Mesh Datafile), 3DS and COLLADA as input. 3DS and COLLADA are widely used file formats which can store 3D triangle mesh. And TMD is file format which is defined by ourselves. In Figure 8, there is an example of TMD file format structure. We utilize data during whole procedure including the loaded triangle mesh data as a data model which is shown in Figure 2. In this section, *Solid*, *Surface*, *Triangle*, *Vertex* and *Edge* are class or class instances of this model, in Figure 9.



```
# Notations
#
# `#`: comments
# index: 0-based (first element index = 0)

# vertex
# v x y z

v -2.837600 -6.461940 0.884856
v -2.838010 -6.463430 0.787984
v -2.857950 -6.365120 0.883447
v -2.858360 -6.366610 0.786575

# group
# g group_name

# faces
# - only triangles can be defined
# - defined as counterclockwise (CCW) winding
# - CCW is a standard for OpenGL
# f v0 v1 v2 (zero-based indexing)

g group1

f 0 2 3
f 0 1 3

g group2

f 1 2 3
f 1 0 3
```

Figure 8. An example of TMD file format

5.2 Construction triangle adjacency graph

Two triangles with adjacent relationship are represented as two node and connecting edge in the adjacency graph. In this process, we establish adjacent relationships for all adjacent *Triangle* pairs,

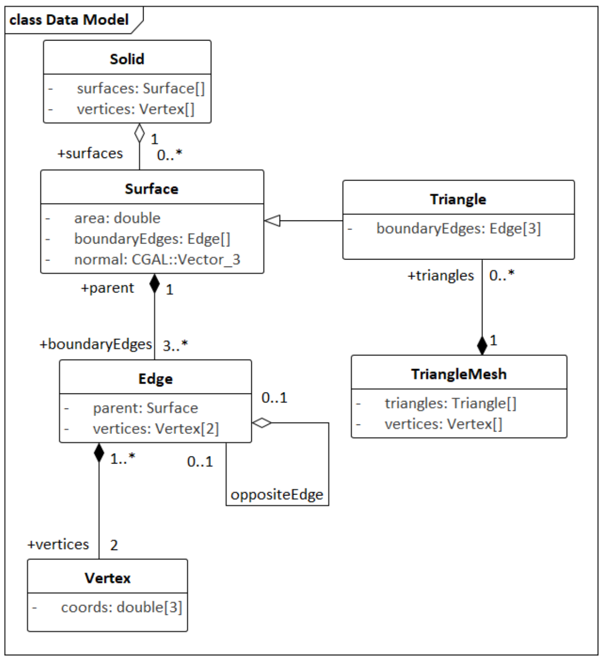


Figure 9. Data model for construction procedure

and then a triangle mesh graph is constructed based on these relationships. Within this graph, there is one or more connected components. Each connected component represents an architectural cell in IndoorGML or a non-architectural element.

5.3 Non-architectural elements elimination

We must determine whether each connected component of the graph represents either an architectural cell or a non-architectural element. We assume that normal vectors of all *Surfaces* that make up architectural cell are oriented towards the inside of the cell. From this assumption, if normal vectors of all connected triangles are directed to inward, this component represents an architectural cell. If not, it represents a non-architectural element. Figure 10 visualizes this assumption by example. In this process, we eliminate triangles corresponding to non-architectural elements.

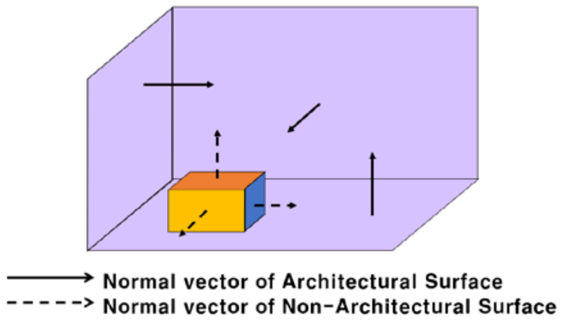

Normal vector of Architectural Surface
Normal vector of Non-Architectural Surface

Figure 10. An Example of architectural and non-architectural normal vector

5.4 Solidification

This section presents a series of steps to build solid data from fine-grained triangle mesh data to planar surfaces. Figure 11 shows these steps in this section. *Solid* data is constructed by performing these steps for each triangle mesh data corresponding to each connected component of the graph. The first step is to make multiple fine-grained *Surfaces* into a single *Surface* by merging them. The second step simplifies edges between the created *Surfaces* by decimating intermediate Vertices. The third step is to construct solid data composed of planar surfaces by polygonizing each *Surface*. The following sections describe each step in detail.

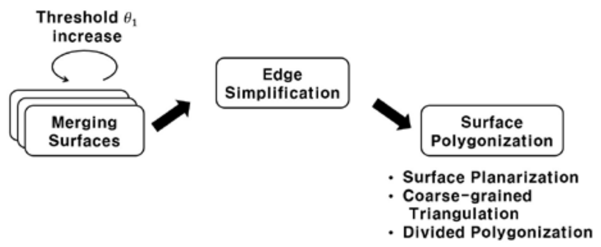


Figure 11. Solidification

5.4.1 Merging Surfaces (Step 1): This step merges multiple *Surfaces* into a single *Surface*. We propose a greedy algorithm 1 for this. This algorithm first selects two adjacent *Surfaces* and verifies that those *Surfaces* can be merged. If possible, we merge two *Surfaces* and repeat the same operation for all *Surfaces*. At this stage we ease the mergeability condition of this algorithm gradually to the limit and iterate the algorithm until there are no more *Surfaces* to be merged. We describe these steps in detail in Section 6.

```
input  : $SL$ : List of $Surfaces$
output: $SL$‘ : List of $Surfaces$ after merging
$hasMerged \leftarrow true$ ;
while $hasMerged$ do
    $hasMerged \leftarrow false$ ;
    sort $SL$ by the number of $Vertexs$ of each $Surface$;
    foreach Surface Pair($S_i, S_j$) in $SL$ do
        if $S_i$ can be merged with $S_j$ then
            $hasMerged \leftarrow true$ ;
            merge($S_i, S_j$) ;
        end
    end
end
```

Algorithm 1: mergeSurfaces

5.4.2 Edge simplification (Step 2): This step simplifies *Edges* between every two adjacent *Surfaces*. The edge between two adjacent *Surfaces* not merged in the preceding Step 1 should be straight if both *Surfaces* are planar. All *Surfaces* in current data are not on a plane, but we assume that all *Surfaces* are planar at this stage because they represent surfaces in WS, FS, or CS. Based on that assumption, this step makes edges of all adjacent *Surface* pairs straight. In connected *Edges* shared by two *Surfaces*, we decimate all points except the start *Vertex* of the first *Edge*and the end *Vertex* of the last edge, and recreate new edge between two surfaces. However, if a self-intersection occurs in one of the two *Surfaces* after simplification, it returns to the previous state. This step proceeds until every shared edge of all adjacent *Surface* pairs is simplified.

5.5 Surface Polygonization (Step 3):

The goal of solidification is to create solid data surrounded by planar surfaces. However, *Surfaces* after completing Step 2 may not be planar. So, the purpose of step 3 is to make each *Surface* consist of one polygon (planar *Surface*) or polygons. In other words, this step converts solid data which consists of non-planar surfaces into solid data consisting only of planar surfaces. We propose three ways to achieve the purpose of this step.

***Surface* Planarization :** The first method is to make each *Surface* into a single plane. Then we can easily build solid data surrounded by planar *Surfaces*. There are several ways to convert a *Surface* to a plane (PCA, plane creation using normal vectors and sample points on *Surface*). However, the problem of making every *Surfaces* planar at once, with taking into account the topological information is NP-problem. Sometimes it may not be possible to flatten all the *Surfaces*. Therefore, we are implementing a heuristic algorithm. This work is currently in progress and requires verification.

Coarse-grained triangulation (method CT) : When we flatten a non-planar *Surface*, distortions in the geometry are almost inevitable. But it is possible to maintain *Surface* geometry without distortion when *Surface* is triangulated. The second method is to triangulate each *Surface*. Since computer graphics studies perform triangulation to store and visualize 3D geometry data, storing results as a triangle mesh does not cause much damage. Here, the coarse-grained triangulation means dividing *Surface* into minimum number of *Triangles*, rather than going back to the original fine-grained triangle mesh data. These triangles are distinguished from the original triangle mesh data by not only geometric data information, but also using belonging to architectural *Surface* (FS, WS, CS). Our triangulation function utilizes CGAL triangulation function (Hert and Seel, 2018), which uses only point location information. To keep the topology information, we divide the *Surface* into convex hulls and triangulate each convex hull. Finally, we construct solid data with triangles created by triangulating all surfaces.

Divided polygonization (method DP) : When we triangulate *Surface*, some planar portions of the *Surface* are also divided into multiple *Triangles*, so the *Triangles* are not the minimum number of polygons. We propose a way to make the polygons be as large as possible. Resulting polygons are made by applying Algorithm 1 to *Triangles*, which are made by method CT in each *Surface*. Because these polygons should be planar, we should tighten the planarity decision. For more strict planarity decision, we set threshold to lower value than Section 5.4.1 in execution of Algorithm 1.

5.6 Generating IndoorGML data

We show how to generate IndoorGML data. In order to generate IndoorGML data, we have to create cells. The creation of a cell can be divided into two stages:

Cell identification : In order to create a cell, we have to determine which part of the total data each cell occupies (e.g. room). This study considers one architectural connected component of the graph constructed in Section 5.2 as one cell. Because one connected component become one *Solid* data, we also think of one *Solid* as one cell.

Determination of cell geometry : We also need to determine cell geometry. In Cell identification, we regard one *Solid* as one cell, so the cell geometry can be completed from the *Solid* geometry.

Finally, we produce one IndoorGML document from all cells defined in this way.

6. MERGING SURFACES METHOD

In this section, we describe the procedure of Section 5.4.1 in detail. This section shows how to merge two adjacent *Surfaces*, one of the core processes of this study. In addition, we present problems that may arise during merging process and their solutions.

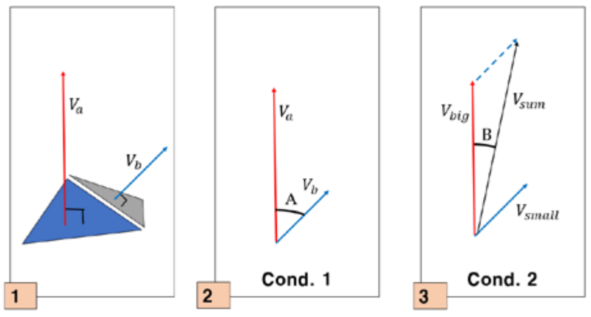


Figure 12. Illustration of threshold and angles.
1) Two normal vectors of *Surfaces*. 2) $\angle A$ 3) $\angle B$

We proceed with this step using the normal vectors of two *Surfaces*. Let two *Surfaces* be S_a, S_b, and normal vectors of each *Surface* is V_a, V_b. For explanation of this section, we utilize definition as follows:

$$V_{sum} = V_a + V_b \tag{5}$$

$$V_{big} = \begin{cases} V_a & \text{if } S_a.area > S_b.area \\ V_b & \text{otherwise} \end{cases} \tag{6}$$

There are several problems to consider at this stage. The first problem is to determine feasibility of merging *Surfaces*. We say that merging is possible if *Surfaces* pair satisfies the following conditions:

$$\begin{aligned} &\text{Condition 1 } : \angle A \leq \theta_1 \\ &\text{Condition 2 } : \angle B \leq \theta_2 \end{aligned} \tag{7}$$

where $\angle A$ is angle between two *Surfaces* normal vector,
$\angle B$ is angle between V_{sum} and V_{big},
θ_1 and θ_2 are thresholds

Figure 12 shows these conditions and angles. If two conditions are satisfied, two adjacent *Surfaces* are merged into one. In condition 1, another studies that merge coplanar adjacent facets set a low value (e.g. 0.1°, 0.01°) for a θ_1. However, since our project is aimed at indoor space and most of data is collected in real space, the data maybe has some noise surfaces.

Noise surface is a surface that has wrong geometry about the actual indoor space. Therefore, even the noise surface is planar with its neighbor in the actual indoor space, $\angle A$ can be large. So, for assumption of noise surface, we set θ_1 to high value (e.g. 10.0°, 20.1°). But, if θ_1 is set to a large value, two different non-noise surfaces may be merged. So we exclude that case through condition 2 and apply only to a noise surface. The characteristic of noise surface is usually smaller than surrounding *Surface*. To use it, we calculate the angle $\angle B$ between V_{sum} and V_{big}. We also let normal vector of *Surface* imply *Surface* area information (This is explained in detail in the next section). Thus, if one of the two *Surfaces* is overwhelmingly large and the other is small (if it is a Noise surface), the angle B is calculated to be a low value. We set θ_2 to a low value, and merge only if $\angle A$ is large but $\angle B$ is small. If $\angle A$ is small, both conditions are satisfied since $\angle A \geq \angle B$.

The second problem is *Surface* normal vector calculation. Most of the *Surfaces* used at this stage are not planar. To deal with the noise in indoor spatial data we merge *Surfaces* even if they are not coplanar. So it is infeasible to calculate singal normal vector of each *Surface*. Instead, we set V_{sum} to normal vector of the merged *Surface*. Since *Triangle* in raw triangle mesh is always planar, we can compute normal vector of each *Triangle*. Then, we select two consecutive *Edges* of one *Triangle*, convert them into vectors, cross-product them, and set the resulting vector as *Triangle* normal vector. The magnitude of the result vector means the area of *Triangle*. (In fact, magnitude of result vector / 2 = *Triangle* area) Since we merge *Triangles* into *Surfaces* and repeat merging, each *Surface* in last iteration has a normal vector. The magnitude of the normal vector is the sum of all triangles area that have been merged to form the *Surfaces*. Thus, this value implies the area of the *Surface*.

The greedy merging method in solving two problems presented above makes different results depends on *Surfaces* selection order. We present a heuristic method to determine the proper merge order by gradient ascent of θ_1. This method iterates the algorithm 1 with gradually increasing θ_1 from a small to a large value. Merging order properly adjusted because pair of *Surfaces* with a smaller $\angle A$ merged first. In addition, there is a possibility to erroneously merge *Surfaces* when θ_1 is large. In contrast, if the value is small, only a small number of *Surfaces* are merged. By sequentially setting values we address issue of determination θ_1 too.

7. EXPERIMENTS

7.1 Dataset

In this work we used indoor spatial data with several rooms and objects. Data represents triangle mesh made from point cloud collected inside of apartment. The data is in TMD format, the number of data vertices is 21732, and the number of triangles is 43368. The range of total space is (11.66, 10.12, 3.49) and the average size of triangles is 0.00832192.

Data Spec.	Value
Vertices number	21732
Triangles number	43368
Range of Space	(11.66, 10.12, 3.49)
Average triangle size	0.00832192

Table 1. Data spec.

During the process of Section 5.2 an adjacent graph of a triangle mesh with 18 connected components constructed. Triangles of 17 connected components among them belong to IS (Indoor furniture surfaces). In order to eliminate them we perform the process of 5.3, leaving only one connected component of 18094 triangles. We summarize the apartment data in Table 1.

7.2 Performance Analysis

In Table 2, we show the workload parameter. Values in bold are default. Figure 13 shows variation of the number of surfaces in the whole process of a case. In this case, θ_2 is 1.0 and Surface Polygonization is done by DP method.

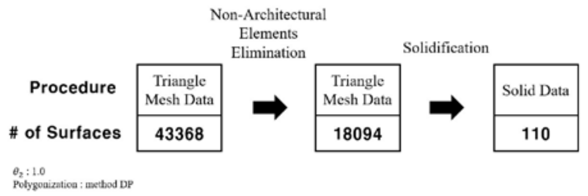


Figure 13. Surfaces number according to process

We specifically explored Step 1 and Step 3 of the 5.4 Solidification process. In step 1, one iteration is a one-time operation of Algorithm 1. Our measure in this experiment is defined as follows:

$$i\text{th iteration reduction ratio } f = \frac{N_i}{N_0} \qquad (8)$$

where N_i = the number of surfaces in ith iteration

Parameters	Settings
θ_1 (°)	**1.0**, 3.0, 5.0, . . . , 39.0
θ_2 (°)	**1.0**

Table 2. Parameters & their settings

Figure 14 changes in reduction ratio of Step 1. Figure 14 also shows changes in the average size of all surfaces.

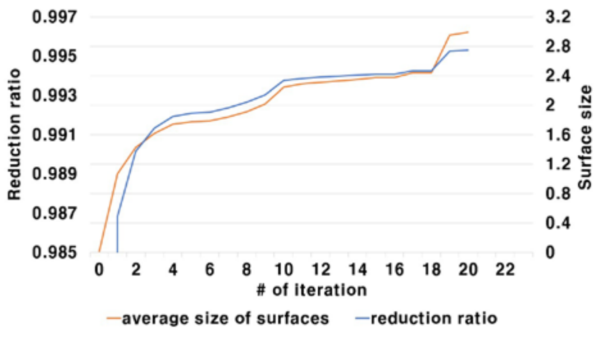
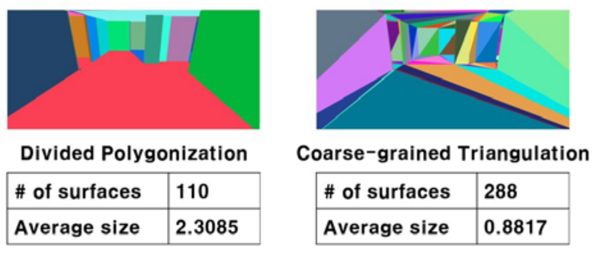


Figure 14. Changes of the number of surfaces and average Surface size in Soldification step 1

Figure 15 compares results of two polygonization methods CT and DP. The method CT makes from each surface multiple triangles, whereas DP makes from each surface multiple polygons. In this experiment, we demonstrate that DP provides a stronger reduction than CT.



# of surfaces	110
Average size	2.3085

# of surfaces	288
Average size	0.8817

Figure 15. Two polygonization methods comparison

7.3 IndoorGML Document Validation

As a final step of our work we create an IndoorGML document. The following Figure 16 shows generated IndoorGML document visualization. We visualize IndoorGML data to confirm there were no problems for representing the indoor space.

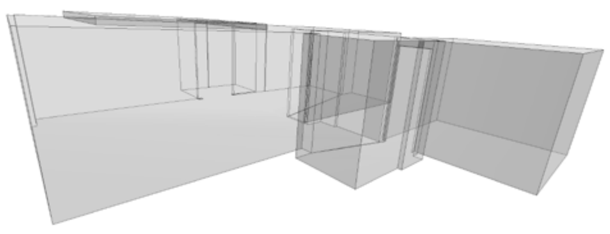


Figure 16. Visualization of IndoorGML Document

8. CONCLUSION & FUTURE WORK

In this paper, we proposed a novel method for the simplification of triangular mesh data collected from point cloud. While a number of researches have been done for the triangular mesh simplification, the main difference of our work is that we considered the properties of indoor space based on the data model defined in OGC IndoorGML. The main contributions of this work are as follows:

- We built appropriate indoor spatial geometry data from fine-grained triangle mesh.
- We simplified indoor spatial data by using properties of indoor space.
- We suggested the method to generate indoor spatial standard data model from triangle mesh.

We demonstrated the achieved level of compression in experiments. In addition, we visualize our intermediate and final output. Experimental results show that DP method is more suitable for the original purpose.

In future work, we plan to design a method for complex indoor spaces (e.g. large exhibition halls, sports stadiums, etc.). We will also implement the function of constructing a door or a virtual door for data that is not distinguished from the outside or inside, or data that cannot be not divided into cells.

We also open work to contribute to the open source spatial information ecosystem. We release the source code on Github, see : https://github.com/STEMLab/TMI-Converter.git.

9. ACKNOWLEDGMENT

This work was partially supported by BK21PLUS, Creative Human Resource Development Program for IT Convergence, a grant (18NSIP-B135746-02) from National Spatial Information Research Program (NSIP) funded by Ministry of Land, Infrastructure and Transport of Korean government and the Global Frontier R&D Program on <Human-centered Interaction for Coexistence> funded by the National Research Foundation of Korea grant funded by the Korean Government(MSIP) (2017M3A6A3052686).

REFERENCES

ChangHyun, J., Kang, J., Yeon, S., Choi, H., Chung, T.-Y. and Doh, N. L., 2017. Towards a realistic indoor world reconstruction: preliminary results for an object-oriented 3D RGB-D mapping. *Intelligent Automation & Soft Computing* 23(2), pp. 207–218.

Choi, H., Jun, C., Li Yuen, S., Cho, H. and Doh, N. L., 2013. Joint solution for the online 3D photorealistic mapping using SfM and SLAM. *International Journal of Advanced Robotic Systems*.

Cignoni, P., Montani, C. and Scopigno, R., 1998. A comparison of mesh simplification algorithms. *Computers & Graphics* 22(1), pp. 37–54.

DeCoro, C. and Tatarchuk, N., 2007. Real-time mesh simplification using the GPU. In: *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, ACM, pp. 161–166.

Hert, S. and Seel, M., 2018. Convex Hulls and Delaunay Triangulations. In: *CGAL User and Reference Manual*, 4.12 edn, CGAL Editorial Board.

Hinker, P. and Hansen, C., 1993. Geometric optimization. In: *Proceedings of the 4th conference on Visualization'93*, IEEE Computer Society, pp. 189–195.

Kalvin, A. D. and Taylor, R. H., 1996. Superfaces: Polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Applications* 16(3), pp. 64–77.

Kang, H.-K. and Li, K.-J., 2017. A Standard Indoor Spatial Data ModelOGC IndoorGML and Implementation Approaches. *ISPRS International Journal of Geo-Information* 6(4), pp. 116.

Kolbe, T. H., Gröger, G. and Plümer, L., 2005. Citygml: Interoperable access to 3d city models. In: *Geo-information for disaster management*, Springer, pp. 883–899.

Ryoo, H.-G., Kim, T. and Li, K.-J., 2015. Comparison between two OGC standards for indoor space: CityGML and IndoorGML. In: *Proceedings of the Seventh ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness*, ACM.

Schroeder, W. J., Zarge, J. A. and Lorensen, W. E., 1992. Decimation of triangle meshes. In: *ACM Siggraph Computer Graphics*, Vol. 26number 2, ACM, pp. 65–70.

Stroud, I., 2006. *Boundary representation modelling techniques*. Springer Science & Business Media.

The CGAL Project, 2018. *CGAL User and Reference Manual*. 4.12 edn, CGAL Editorial Board.