Straight Skeleton Computation Optimized for Automatic Generation of 3D Roof Model

○ Kenichi Sugihara[*1], Zhenjiang Shen[*2]

*1 Professor, Faculty of Information Media, Gifu Kyoritsu University

*2 Professor, Faculty of Geosciences and civil Engineering, Institute of Science and Engineering, Kanazawa University

Summary: 3D building models with roofs are important in several fields, such as urban planning and BIM (Building Information Model). However, enormous time and labor are required to create these 3D models. In order to automate laborious steps, a GIS and CG integrated system has been proposed for the automatic generation of 3D building models, based on building polygons (building footprints) on digital maps. The generation is implemented through straight skeleton computation, in which three events ('Edge' and 'Split', 'Vertex' events) were proposed. In the computation process, usually three edges propagate into a node. Often it causes an acute angle shape that is not appropriate for roof boards. To avoid the inappropriate shape, in this paper, methodologies are proposed for adding 'Line segment' events besides the conventional events, and for monotone polygon nodes sorting.

Keywords: straight skeleton, automatic generation, 3D building model, GIS, 3D CG, building footprint

1. Introduction

3D town models, as shown in Figure 1 right, are important in urban planning and architectural design, e.g., BIM. However, enormous time and labour has to be consumed to create these 3D models, using 3D modeling software such as 3ds Max or SketchUp. In order to automate the laborious steps, a GIS (Geographic Information System) and CG integrated system has been proposed for automatically generating 3D building models, based on building polygons (building footprints) on a digital map shown in Figure 1 left [1)2)3)]. In the digital map, not all building polygons are orthogonal. In either orthogonal or non-orthogonal polygons, methodologies were proposed for automatically generating 3D building models with general shaped roofs by the straight skeleton defined by a continuous shrinking process proposed by Aichholzer et al.[4)]. In their proposal, two events ('*Edge*' and '*Split*' events) will occur during shrinking process. Besides two events, Eppstein et al. [5)] suggested a '*Vertex*' event in which two or more reflex vertices reach the same point simultaneously. A reflex vertex is a vertex whose internal angle is greater than 180 degrees. However, some roofs are not created by these three events proposed. In our paper [3)], the methodology was proposed for constructing roof models by assuming 'the *Third* event' in which a reflex vertex runs into the edge, but the other split polygon is collapsed into a node (an *Edge* event happens in the *Split* event at the same time).

Our contribution lies in a new methodology for adding a '*Line segment*' event by which the inappropriate roof board

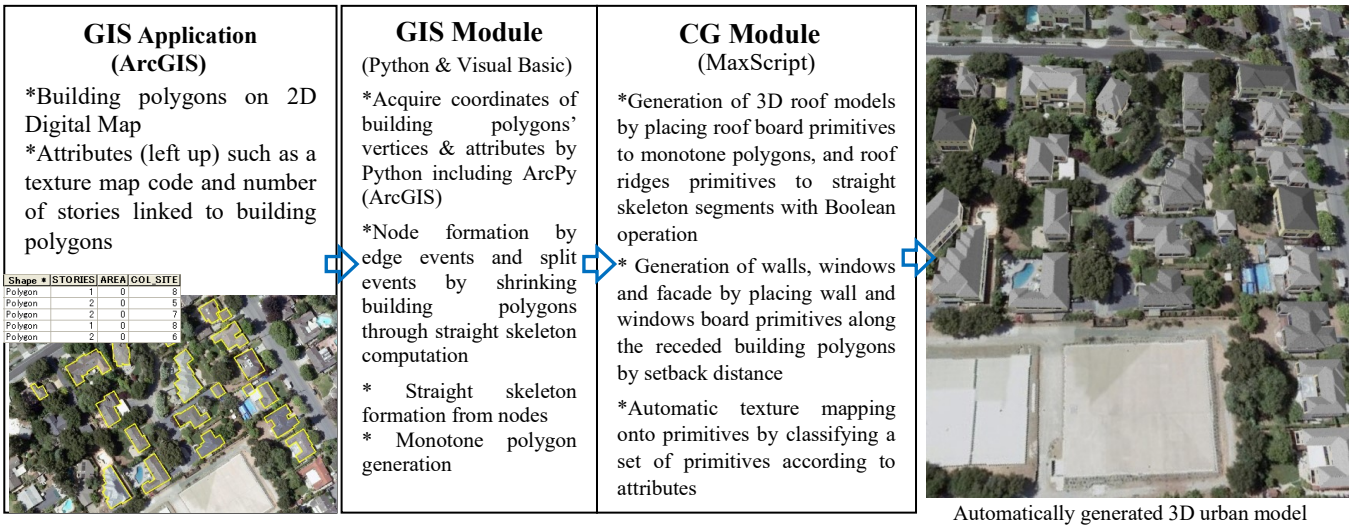

Figure 1. Pipeline of Automatic Generation for 3D Building Models by Straight Skeleton Computation

shape can be avoided. In the straight skeleton computation, the shrinking process continues if split polygons have non-zero area. The shrinking process ends when split polygons fall into 'Vertex' or 'Line segment' since they have no area. Consequently, a 'Line segment' can be a resulting shape of shrinking procedure, and we classify a '*Line segment*' event in which two line segments collapse into one line segment.

Usually three edges propagate into a node. Often it causes an acute angle shape that is not appropriate for roof boards shown in Figure 3(d). To avoid the inappropriate shape, a '*Line segment*' event is proposed. We also propose 'monotone polygon nodes sorting' by which not self-intersecting monotone polygons are formed, where 'monotone polygons' are the areas divided by a straight skeleton, as shown in Figure 2(c).

2. Related Work

Since 3D building models are utilized in several fields, various types of technologies, ranging from computer vision, computer graphics, photogrammetry, and remote sensing, have been proposed and developed for creating 3D building models. Procedural modeling is an effective technique to create 3D models from sets of rules such as L-systems, fractals, and generative modeling language [6]. Mueller et al.[7] have created an archaeological site of Pompeii by using a shape grammar. They import data from a GIS database and try to classify imported mass models as basic shapes in their shape vocabulary. If this is not possible, they use a general extruded footprint together with a general roof obtained by the straight skeleton computation defined by a continuous shrinking process [4].

As a new generalization of straight skeletons, Helda et al.[8] introduce additively-weighted straight skeletons. An additively-weighted straight skeleton is the result of a wavefront-propagation process where wavefront edges do not necessarily start to move at the begin of the propagation, resulting in an automated generation of roofs in which the individual facets have different inclinations and start at different heights.

By using the straight skeleton, Kelly et al.[9] present a user interface for the exterior of architectural models to interactively specify procedural extrusions, a sweep plane algorithm to compute a two-manifold architectural surface.

By the interactive modeling, 3D building models with plausible detailed façade can be achieved. However, the limitation of these modeling is the large amount of user interaction involved[10], and the models created are 'surface models' by sweeping or extruding, revolving 2D primitive geometries. When creating 3D building models for architectural design and BIM, 3D building models should be made up of solid geometries primitives which will be parts of the building, created through Boolean operation. While the 'surface models' trace the surface of the parts of 3D building model, the 3D models used for architectural design should consist of building component created by CSG (Constructive Solid Geometry). Thus, the GIS and CG integrated system is proposed for automatically generating 3D building models by CSG.

3. Pipeline of Automatic Generation

As the pipeline of automatic generation is shown in Figure 1, the source of 3D models is a digital map that contains building polygons linked with attributes data, such as the number of stories and the type of roof, shown in Figure 1 left up. The maps are then preprocessed at the GIS module, and the CG module finally generates the 3D building model.

The preprocessing at the GIS module includes the procedures as follows: **(1)** Calculate the minimum receding distance for an *Edge* event (including a *Third* and *Line segment* event). Until the *Edge* event occurs, check if a *Split* event happens by starting continuous shrinking process. **(2)** Start continuous shrinking process in which edges of the polygon move inward, parallel to themselves at a constant speed shown in Figure 2(a)&2(b). **(3)** Detect any event such as a *Split*, *Edge* or *Line segment* event during shrinking process, and formation of nodes by these events. The position of the node is calculated by the intersection of angular bisectors. **(4)** Inherit and store three or more 'original edge ID' (e.g. *edgN* in Figure 2(a)) linked to the node during the shrinking process in which the topology of the polygon will change. In shrinking process, Figure 2(b) shows *edg2* firstly disappears into *Node1*, and two edges (*edg8 & 9*) secondly result in *Node2*. Since at least three original edges sweep into the node, *edg1,2 & 3* propagation result in *Node1*, and *edg4,5 & 10* propagation result in *Node3* (by *Split* event). **(5)** Every (original) edge will inquire 'each node' having three or more ID to find out which node has the same 'original edge ID'. If so, then nodes of the same ID are collected and the set of nodes are sorted according to the edge vector to form 'monotone polygon' and the straight skeleton. **(6)** Calculate the length, width, center position and inclination of the bounding rectangles for 'monotone polygons'. **(7)** Export the coordinates of polygons' vertices, 'monotone polygons' information, and attributes of buildings.

As shown in Figure 1, the CG module receives the pre-processed data that the GIS module exports, generating 3D building models. In GIS module, the system measures the length and inclination of the bounding rectangle for the monotone

polygon that will be a roof board. The CG module generates a bounding box of the length and width, measured in GIS module. The monotone polygons will be converted into primitives, i.e., thin boxes by Boolean operation between the extrusion of the monotone polygon and the box primitive.

In case of modeling a building with roofs, the CG module follows these steps: **(1)** Generate primitives of appropriate size, such as boxes, prisms or polyhedra that will form the various parts of the house. **(2)** Boolean operations applied to these primitives to form the shapes of parts of the house, for examples, making holes in a building body for doors and windows, making trapezoidal roof boards for a hipped roof and a temple roof. **(3)** Rotate parts of the house according to the inclination of the partitioned rectangle. **(4)** Place parts of the house. **(5)** Texture mapping onto these parts according to the attribute received. **(6)** Copy the 2nd floor to form the 3rd floor or more in case of building higher than 3 stories.

4. Straight Skeleton Computation

Aichholzer et al. [4] introduced the straight skeleton defined as the union of the pieces of angular bisectors traced out by polygon vertices during a continuous shrinking process in which edges of the polygon move inward, parallel to themselves at a constant speed. The straight skeleton is applied to constructing general shaped roofs based on any simple building polygon, regardless of their being rectilinear or not.

As shrinking process shown in Figure 2, each vertex of the polygon moves along the angular bisector of its incident edges. This situation continues until the boundary change topologically. According to Aichholzer et al. [4], there are two possible types of changes:

(1) ***Edge event***: *An edge shrinks to zero, making its neighboring edges adjacent now.*

(2) ***Split event***: *An edge is split, i.e., a reflex vertex runs into this edge, thus splitting the whole polygon. New adjacencies occur between the split edge and each of the two edges incident to the reflex vertex.*

If the sum of the internal angles of two vertices incident to an edge is more than 360 degrees, then the length of the edge increases, otherwise the edge will be shrunk to a point (node). Shrinking procedure is uniquely determined by the distance $\mathbf{d}_{shri}$ between the two edges of before & after shrinking procedure. The distance e_$\mathbf{d}_{shri}$ is the $\mathbf{d}_{shri}$ when an edge event happens in the shrinking process. e_$\mathbf{d}_{shri}$ for the edge (ed_i) is calculated as follows:

$$e\_\mathbf{d}_{shri} = {}^{L_i}\Big/_{(\cot(0.5*\theta_i)+\cot(0.5*\theta_{i+1}))} \quad (1)$$

where L_i is the length of ed_i, and θ_i & θ_{i+1} are internal angles of vertices incident to ed_i.

When $0.5*\theta_i + 0.5*\theta_{i+1} < 180$ degrees, i.e., the sum of the internal angles of two vertices incident to an edge is less than 360 degrees, an *Edge* event may happen unless the edge is intersected by an angular bisector from a reflex vertex and a *Split* event happens.

4.1. HOW STRAIGHT SKELETON IS FORMED

How a straight skeleton and monotone polygons are formed is as follows.

(1) One simple polygon (**P**) is given such as shown in Figure 2(a). If there is any reflex vertex in the **P**, then it can be divided into two or more polygons.

(2) The system calculates e_$\mathbf{d}_{shri}$ (receding distance for an *Edge* event, shown in (1)) for all edges and finds the shortest of them. Then, the system checks if a *Split* event occurs by increasing $\mathbf{d}_{shri}$ by (e_$\mathbf{d}_{shri}$ /n_step). In this way, the shrinking process may proceed until $\mathbf{d}_{shri}$ reaches the shortest e_$\mathbf{d}_{shri}$ calculated.

(3) During shrinking until $\mathbf{d}_{shri}$ reaches the shortest e_$\mathbf{d}_{shri}$, the

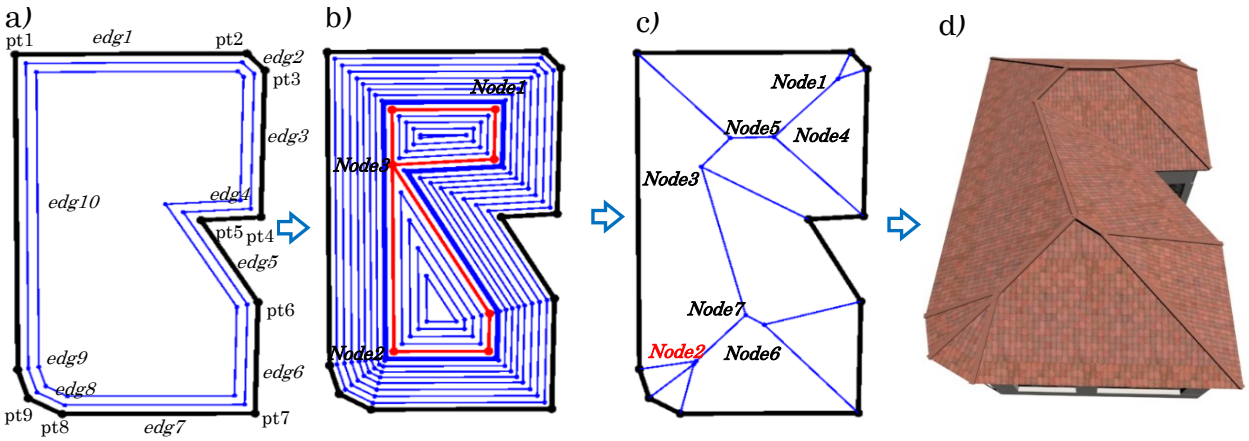

Figure 2. Shrinking process and a straight skeleton, a roof model generated. **a)** Input polygon (bold) start continuous shrinking process in which edges of the polygon move inward, parallel to themselves at a constant speed. **b)** Shrinking polygons (blue) by edge or no event, and red one by a split event. **c)** The straight skeleton (blue) and monotone polygons. **d)** A roof model automatically generated: each roof board is based on an 'monotone polygon'.

system checks if a 'checking angular bisector' from a reflex vertex intersects another edge of the polygon or not. If an edge is found intersected, then the system calculates the node position by the *Split* event. The position of the node is calculated by the intersection of two angular bisectors: one from the reflex vertex and the other between the intersected edge and one of two edges incident to the reflex vertex. However, edges may be intersected by several 'checking angular bisectors' from several reflex vertices. Among the several reflex vertices, the reflex vertex that gives the shortest $\mathbf{d}_{shri}$ will be selected for calculating the node position.

(4) In the process of (2), a *Split* event may happen and the polygon will be divided into some polygons: **P**s.

In this '*Split* event checking' process, all divided polygons are checked if they can be divided more. As long as there are some **P**s that can be divided, '*Split* event checking' routine will continue. After that, the system concentrates on the *Edge* event procedure.

(5) In this stage, since the number of polygons divided does not increase by the *Split* event, the system can focus on the *Edge* event including *Third* and *Line segment* event procedures. If the polygon divided has only three vertices, then the polygon (triangle) collapses to a node; this is the final stage for the polygon divided.

(6) While the *Edge* events are being executed, the topology of the polygon will change. If the change happens, then the system re-implement the process from (2) to (5) for the polygon whose topology has changed. At that moment, the system recalculates the length of each edge and the internal angle of each vertex in order to find the shortest $\mathbf{d}_{shri}$ for next events. This re-implementation process continues until all polygons changed collapse to a node or a line segment.

4.2. NODE STRUCTURE

The generated node is associated with the edges of an original polygon **P** (original edge: o-edge) which are identified by 'o-edge ID', e.g. *edg1* & *edg2* in Figure 2(a), since at least three o-edges (original edges) sweep to form a node. Therefore, at each event when the node is generated, at least three o-edges are linked to the node. This means more than three o-edges ID are stored in the node with an appropriate structure.

In our system, a node has the following properties; **(a)** '*Node Type*' (how the node is risen; by an *Edge* event or a *Split* event, *Vertex* event, *Multiple Edge* event and so on) **(b)** 'Number of forming edges' (usually three edges sweep to form a node, but more than three edges sweep in case of *Multiple Edge* event) **(c)** 'o-edge ID preceding the vanishing edge' (by an *Edge* event) or 'o-edge ID of the edge incident to the reflex vertex' (by a *Split* event) **(d)** 'o-edge ID following the vanishing edge' (by an *Edge* event) or 'o-edge ID of the other edge incident to the reflex vertex' (by a *Split* event) **(e)** 'o-edge ID of at least one vanishing edge' (by an *Edge* event) or 'o-edge ID of a split edge' (by a *Split* event)

Since three edges usually sweep into the node, three 'o-edge IDs' are stored in the property of a node. These IDs are used for forming a monotone polygon by collecting the nodes which has the same 'o-edge ID' as each own o-edge ID.

In special cases, four or more edges collapse into a node, such as *Node2* in Figure 2 and *Node2* & *Node5* in Figure 3(e). In extreme cases, such as a hexagon or a regular polygon, a star-shaped polygon collapses to a node, four or more o-edges will sweep into a node. Therefore, a node needs 'Number of forming edges' property. This is the case of a multiple *Edge* event or the case Eppstein et al.[5] defined as a 'degenerate case' in which the straight skeleton can have vertices of degree higher

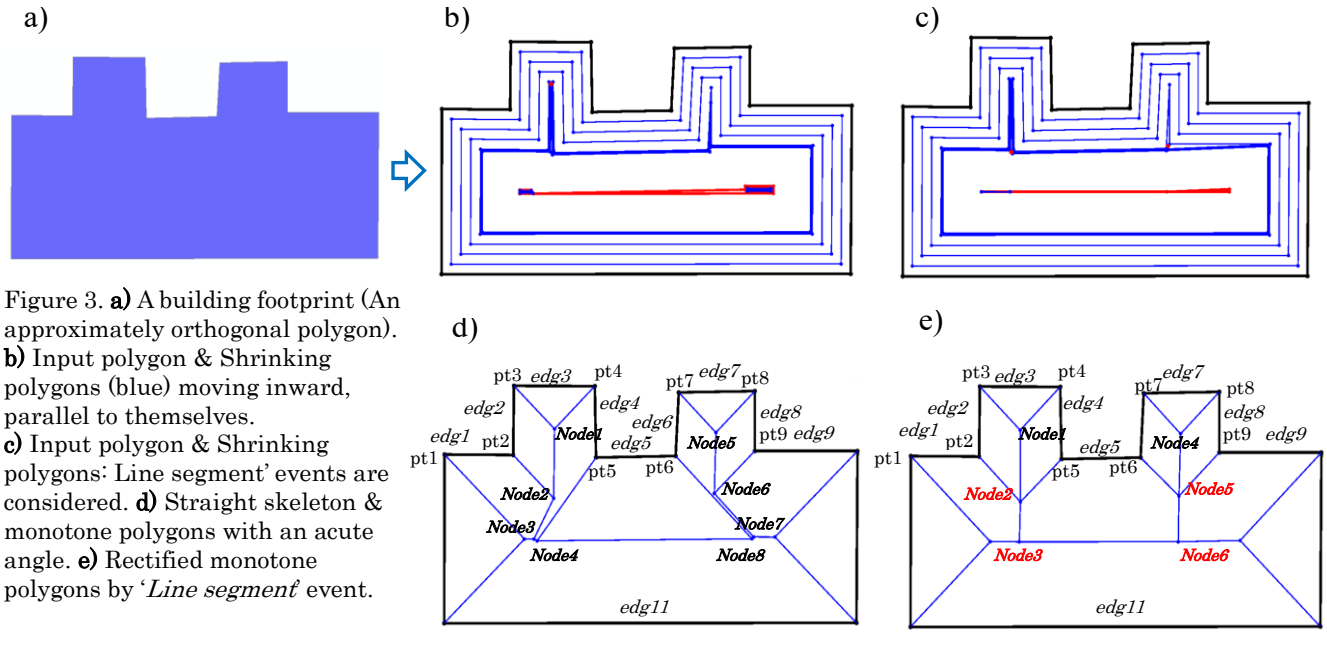

Figure 3. **a)** A building footprint (An approximately orthogonal polygon). **b)** Input polygon & Shrinking polygons (blue) moving inward, parallel to themselves. **c)** Input polygon & Shrinking polygons: Line segment' events are considered. **d)** Straight skeleton & monotone polygons with an acute angle. **e)** Rectified monotone polygons by '*Line segment*' event.

than three, introduced by simultaneous events at the same location. However, in single or double precision floating point calculation for the position of the node, it is quite rare for four or more vertices to reach exactly the same point simultaneously. To rectify monotone polygons to be appropriate shape for roof boards, in our system, if multiple edges collapse into a certain area considered as a point for a node, then they are considered to converge into the same point and the node is formed.

4.3 LINE SEGMENT EVENT

Since three edges usually sweep into a node, very often this causes a quite acute angle shape that is not appropriate for roof board shape shown in Figure 3. In Figure 3(e), pt5 propagates to join pt2 and four edges (*edg1,2,4,5*) propagate into *Node2*, whereas, in Figure 3(d), pt5 does not join pt2 and goes off *Node2*, and three edges (*edg4,5,11*) result in *Node4* with acute angle shape.

This acute angle shape is also found at the figure of Eppstein et al.[5], which uses perturbation techniques, replacing the high-degree node with several nodes of degree three, connected by zero-length edge. In our system, using the technique completely opposite to Eppstein's perturbation, a '*Line segment*' event is proposed where edges are overlapped and collapse into a line segment instead of a vertex to avoid the acute angle shape. This so-called snapping function is done by setting up a certain range for possible '*Line segment*' events, in which edges converge into a certain area considered as a line segment, then they are regarded as converging into the same line segment.

By a '*Line segment*' event, two parallel edges converge into one edge (line segment), and the convergent line segment is detached from a next shrinking body polygon. But if the detached line segment leaves no vertex for next shrinking process, then the line segment is disconnected from a body skeleton. Therefore, the detached line segment leaves at least one vertex for next shrinking process. Examples are shown in the line segment between *Node2* and *Node5* in Figure 4(c) and Figure 5(b); one node whose interior angle is flat will remain for the next shrinking process so as to create the border of monotone polygons. For example, in Figure 4(c) & Figure 5(b), four edges (*edg11,12,14,15*) propagate into *Node2*, and two overlapping edges (*edg12,14*) turn into the line segment incident to *Node2* & nearby *Node* after *edg13* disappeared.
If a configurable range is quite narrow, then *edge* propagation will be extended, ending in *Node5* as shown in Figure 5(a); three edges (*edg12,14,15*) result in *Node2*, and three edges (*edg1,11,15*) result in *Node5* whose inner angle is quite acute, which is improper for roof board shape.

4.4 MONOTONE POLYGON NODES SORTING

According to Aichholzer et al.[4], the area divided by a straight skeleton will be a 'monotone polygon'. To get the monotone polygons, the set of the nodes belonging to each original edge will be sorted according to the 'coordinate value of node vector projections' onto the original edge vector parallel to each original edge. These nodes are coplanar and will form roof boards for a 3D building model. However, for some polygon, this methodology does not work, resulting in self-intersecting polygons. Figure 5(c) shows monotone polygons for *edg13* are self-intersecting. This is because the edge (connecting *Node3* & *Node4*) of the monotone polygon is perpendicular to the original edge (*edg13*) of the polygon, and the nodes are connected in the order of 'node vector projection'. The self-intersections are also found at *edg29* in Figure 5(c) and *edg21* in Figure 5(b).

To avoid self-intersection, the azimuth angle of the nodes belonging to the same monotone polygon is proposed, where

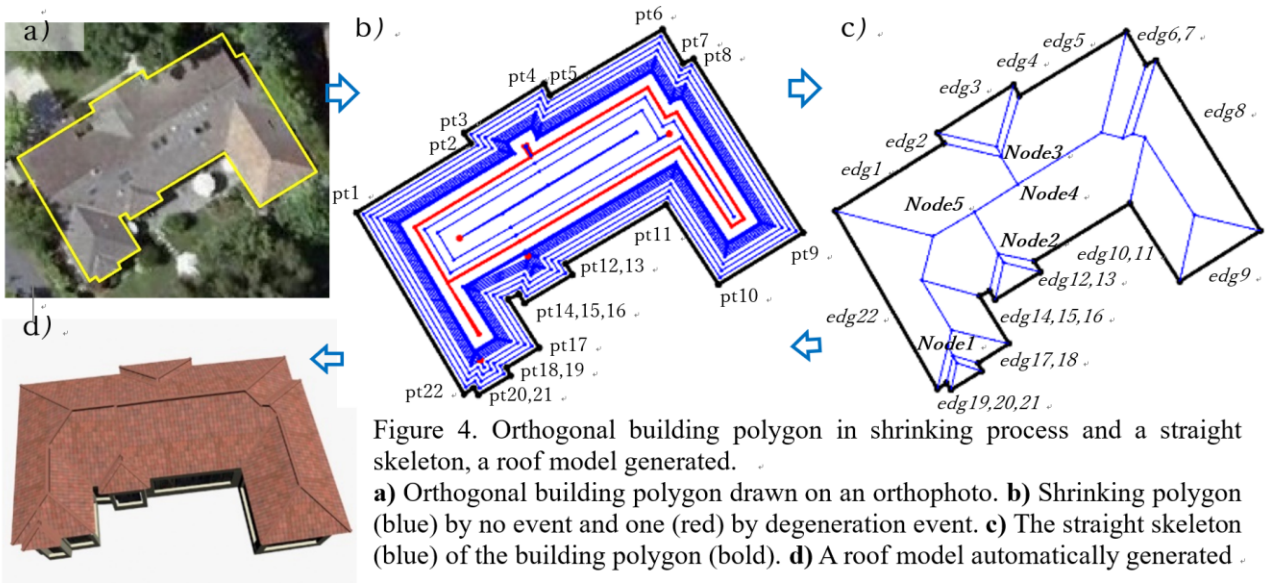

Figure 4. Orthogonal building polygon in shrinking process and a straight skeleton, a roof model generated.
a) Orthogonal building polygon drawn on an orthophoto. **b)** Shrinking polygon (blue) by no event and one (red) by degeneration event. **c)** The straight skeleton (blue) of the building polygon (bold). **d)** A roof model automatically generated

the azimuth is the angle between each original edge vector and a node vector. The first node in the monotone polygon vertices numbering is selected from the node with least azimuth, and the last node is the node with greatest azimuth, since the nodes near the both ends on an original edge may wrap around both ends for some monotone polygons, and wrapping around nodes may not have simply increasing 'coordinate value'. For example, in Figure 5(c), the edge (connecting *Node1* & *Node2*) of the monotone polygon is perpendicular to the original edge (*edg29*), and *Node1* & *Node2* have the same 'coordinate value', resulting in self-intersection at nodes sorting. Thus, the nodes at ends are sorted by the azimuth angles. Then, the sorting of the nodes is found successful in a complicated shape polygon such as the ones in Figure 1 and Figure 4(c).

5. Conclusion

Our system does not always aim at creating current 3D town models (e.g. 3d models in Google Earth) but generating hipped roof building models. The roof models created by the straight skeleton computation are limited to hipped roofs, since short edges are disappearing when shrinking process proceeds and long edges are remain as the ridges of the roof shown in Figure 2 & 4. When creating hipped roof models, our system instantly generates them from building polygons.

The advantage of our generation system is that our 3D building models created can be utilized for architectural design, i.e., BIM, while 3D models created by procedural modeling are not solid models but surface models which are to be converted into geometric primitives (CSG) when they are used for construction design. In this paper, the straight skeleton computation optimized for automatic generation of 3D roof model is proposed by adding '*Line segment*' event besides the conventional events, and 'monotone polygon nodes sorting' by which self-intersecting monotone polygons are not formed.

Acknowledgements

This work was funded by JSPS KAKENHI; Grant-in-Aid for Scientific Research (C) Grant Numbers 18K04523, 16K01045.

References

1) Sugihara K., Hayashi Y.: 2008, Automatic Generation of 3-D Building Models with Multiple Roofs, Journal of Tsinghua Science & Technology, 13, 368-374.
2) Sugihara, K. and Kikata, J.: 2013, Automatic Generation of 3D Building Models from Complicated Building Polygons, Journal of Computing in Civil Engineering, ASCE (American Society of Civil Engineers), 27(5), 476–488.
3) Sugihara K.: 2012, Straight Skeleton for Automatic Generation of 3-D Building Models with General Shaped Roofs, Communication Proc., WSCG'2013, 175–183.
4) O. Aichholzer, F. Aurenhammer ,and D. Alberts, B. Gärtner: 1995, A novel type of skeleton for polygons, Journal of Universal Computer Science, 1 (12), 752–761.
5) Eppstein David, Erickson Jeff: 1999, Raising roofs, crashing cycles, and playing pool: applications of a data structure for finding pairwise interactions, Discrete and Computational Geometry, 22 (4): 569–592.
6) Yoav I. H. Parish, and Pascal Müller: 2001, Procedural modeling of cities, Proceedings of ACM SIGGRAPH 2001, ACM Press, E. Fiume, Ed., New York, 301–308.
7) Pascal Mueller, Peter Wonka, Simon Haegler, Andreas Ulmer, Luc Van Gool.: 2006, Procedural modeling of buildings, ACM Transactions on Graphics 25, 3, 614–623.
8) Helda M., Palfradera P.: 2017, Straight Skeletons with Additive and Multiplicative Weights and Their Application to the Algorithmic Generation of Roofs and Terrains, Computer-Aided Design, Elsevier B.V., 92, 33-41.
9) Tom Kelly, Peter Wonka: 2011, Interactive Architectural Modeling with Procedural Extrusions, ACM Transactions on Graphics (TOG), 30 (2), 24-41.
10) Nianjuan, Jiang, Ping, Tan, and Loong-Fah, Cheong: 2009, Symmetric architecture modeling with a single image, ACM Transactions on Graphics (TOG), 28(5)

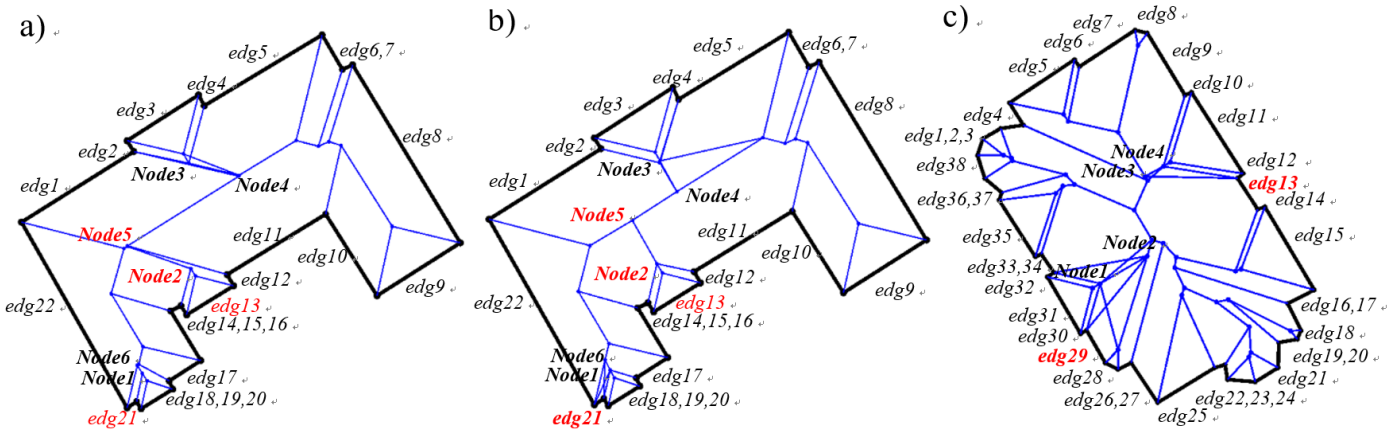

Figure 5. Monotone polygons with acute angle and self-intersecting monotone polygons**. a)** Some monotone polygons have acute angle if a degenaration event does not occur. **b)** & **c)** Some monotone polygons will be self-intersecting if the set of the nodes are sorted according to the 'node vector projections' onto the original edge vector.