Straight Skeleton Computation Optimized for Roof Model Generation

Kenichi Sugihara
Gifu Keizai University
5-50 Kitagata-chou
Ogaki city, Gifu-Pref., 503-8550, Japan
sugihara@gifu-keizai.ac.jp

ABSTRACT

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 is 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 monotone polygon nodes sorting.

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

1 INTRODUCTION

3D town models, such as the one shown in Fig.1 right, are important in urban planning and architectural design, e.g., BIM (Building Information Model). However, enormous time and labor are required to create these 3D models, using 3D modeling software such as 3ds Max or SketchUp. For example, when manually modeling a house with roofs by Constructive Solid Geometry (CSG), one must follow the following laborious steps:

(1) Generation of geometric primitives of proper size, such as boxes, prisms or polyhedra that will form parts of a house **(2)** Boolean operations are applied to these primitives to form the shapes of parts of a house such as making rectangular holes in a building body for doors and windows **(3)** Horizontal and vertical rotation of parts of a house **(4)** Placing the parts of a house to appropriate positions **(5)** Texture mapping onto these parts.

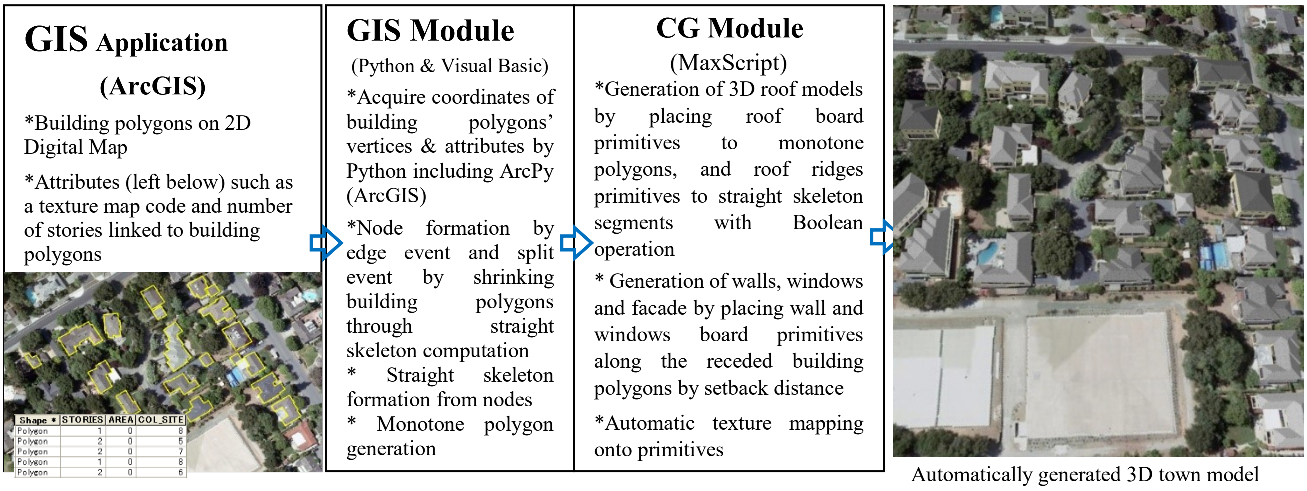
In order to save these laborious steps, a GIS and CG integrated system that automatically generates 3D building models is proposed, based on building polygons (building footprints) on a digital map as shown in Fig.1 left and Fig.7a, which show most building polygons' edges meet at right angles (orthogonal polygon). An orthogonal polygon can be divided or separated into a set of rectangles. The proposed integrated system divides orthogonal building polygons into a set of rectangles and places rectangular roofs and box-shaped building bodies on these rectangles. In the digital map, however, building polygons are not always orthogonal. In either orthogonal or non-orthogonal polygons, the methodology is proposed for automatically creating general shaped roofs by the straight skeleton computation defined by Aichholzer et al. [Aic95].

In their proposal, two events ('*Edge*' and '*Split*' events described in section 4) will occur during shrinking process. Besides two events, Eppstein et al. [Epp99] 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, 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).

In this paper, a methodology is proposed for adding a '*Line segment*' event besides the conventional events.

https://doi.org/10.24132/CSRN.2019.2901.1.12

ISBN 978-80-86943-37-4



GIS Application (ArcGIS)

*Building polygons on 2D Digital Map

*Attributes (left below) such as a texture map code and number of stories linked to building polygons

Shape *	STORIES	AREA	COL_SITE
Polygon	1	0	8
Polygon	2	0	5
Polygon	2	0	7
Polygon	1	0	8
Polygon	2	0	6

GIS Module (Python & Visual Basic)

*Acquire coordinates of building polygons' vertices & attributes by Python including ArcPy (ArcGIS)

*Node formation by edge event and split event by shrinking building polygons through straight skeleton computation

* Straight skeleton formation from nodes

* Monotone polygon generation

CG Module (MaxScript)

*Generation of 3D roof models by placing roof board primitives to monotone polygons, and roof ridges primitives to straight skeleton segments with Boolean operation

* Generation of walls, windows and facade by placing wall and windows board primitives along the receded building polygons by setback distance

*Automatic texture mapping onto primitives

Automatically generated 3D town model

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

The shrinking process continues if polygons split have non-zero area. The shrinking process ends when polygons split 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 will 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 Fig.3. To avoid the inappropriate shape, a '*Line segment*' event will be proposed for straight skeleton computation. 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 Fig.2c.

2 RELATED WORK

Since 3D building models are used for several different purposes, such as urban planning, archaeological reconstruction and game industries, various types of technologies, ranging from computer vision, computer graphics, photogrammetry, and remote sensing, have been proposed and developed for creating 3D building models. When 3D building models are used for urban planning or reconstruction of ancient cities, we will make non-existent building models which we cannot take a picture, and therefore these models are not created by CV, photogrammetry, and remote sensing. So, we focus on procedural modeling, especially roof creation by straight skeleton computation. Procedural modeling is an effective technique to create 3D models from sets of rules such as L-systems, fractals, and generative modeling language [Par01]. Mueller et al. [Mue06] have created an archaeological site of Pompeii and a suburbia model of Beverly Hills by using a shape grammar. They import building footprints from a GIS database and try to classify imported polygons as basic shapes in their shape vocabulary. In case of misclassification, they use a general extruded footprint together with a general roof obtained by the straight skeleton algorithm defined by a continuous shrinking process [Aic95].

The straight skeleton algorithm is useful in generating hipped roofs, since its resulting 'monotone polygons' correspond to the roof boards of a hipped roof. The downside is that the generated roof is only one of many possible roofs, and there is no way to get another roof [Ede14]. To overcome the downside, a new generalization of straight skeletons is proposed by Helda et al. [Hel17], introducing 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. [Kel11] present a user interface for the exterior of architectural models to interactively specify procedural extrusions. They use a profile editor which controls the sweeping of a plane from the base of the building footprint, and they finally construct a wide variety of roofs and a complex architecture.

By these 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 [Jia09], and the models created are 'surface models' by sweeping or extruding dependent on edited profiles, or revolving 2D primitive geometries. However, 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. Thus, the

GIS and CG integrated system that automatically generates 3D building models immediately by CSG (Constructive Solid Geometry) is proposed. Although surface models can have 'beautifully curved roof' by interactive procedural modeling, in reality these roofs are consisted of hundreds of narrow flat boards in most building design. These narrow boards will be properly placed along the roof curve.

3 PIPELINE of AUTOMATIC GENERATION

As the pipeline of automatic generation is shown in Fig.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 Fig.1 left below. 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 occur, check if *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 (Fig.2a&2b). **(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 edges' ID (e.g. *edgN* in Fig.2a) linked to the node during the shrinking process in which the topology of the polygon will change. In shrinking process, Fig.2b 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 rectangle for 'monotone polygon'. **(7)** Export the coordinates of polygons' vertices, 'monotone polygons' information, and attributes of buildings.

In these procedures, the areas divided by a straight skeleton are called as 'monotone polygons' shown in Fig.2c, and to get 'monotone polygons', the set of the nodes belonging to the same original edge will be aligned depending on the coordinate value on the axis parallel to each original edge vector (the 'node vector projections' onto the original edge vector). These nodes are coplanar and will form roof boards for a 3D building model.

As shown in Fig.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,

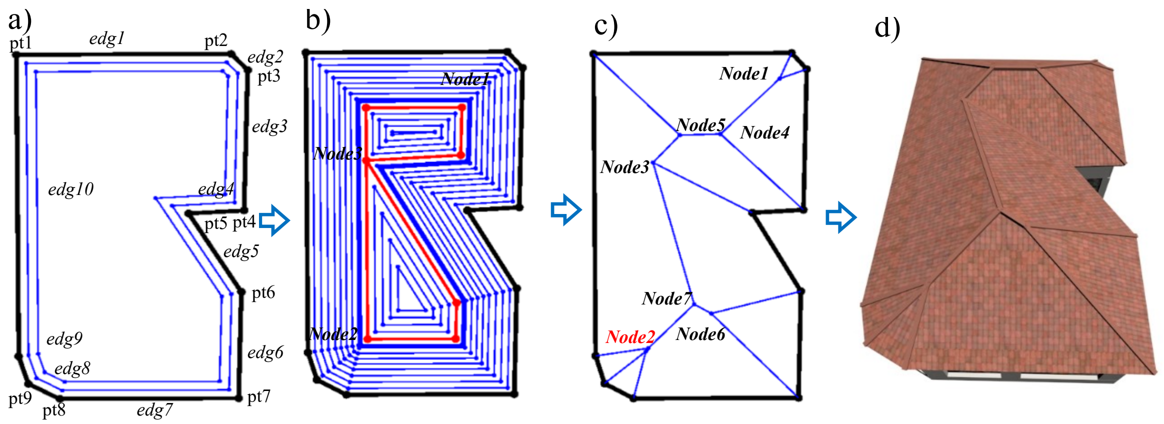


Figure 2: Shrinking process and a straight skeleton, a roof model generated. a) Input polygon (bold) start continuous shrinking process in which edges of the polygon move inward, parallel to themselves at a constant speed. **b)** Shrinking polygons (blue) by no event, and red one by a split event. **c)**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

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.

CG module has been developed using Maxscript that controls 3D CG software (3ds MAX, Autodesk Inc).

4 STRAIGHT SKELETON COMPUTATION

Aichholzer et al. [Aic95] 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 Fig.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. [Aic95], 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.*

The shrinking procedure is uniquely determined by the distance $\mathbf{d}_{\mathrm{shri}}$ between the two edges of before & after shrinking procedure.

The distance e_$\mathbf{d}_{\mathrm{shri}}$ is the $\mathbf{d}_{\mathrm{shri}}$ when an *Edge* event happens in the shrinking process. e_$\mathbf{d}_{\mathrm{shri}}$ for the edge (ed_i) is calculated as follows:

$$\mathrm{e\_}\mathbf{d}_{\mathrm{shri}} = {}^{\mathrm{L}_i}\!/\!_{(\cot(0.5*\theta_i) + \cot(0.5*\theta_{i+1}))} \quad (1)$$

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 Fig.2a. If there is any reflex vertex in the **P**, then it can be divided into two or more polygons.

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

(3) During shrinking until $\mathbf{d}_{\mathrm{shri}}$ reaches the shortest e_$\mathbf{d}_{\mathrm{shri}}$, the 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 $\mathbf{d}_{\mathrm{shri}}$ 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 concentrate 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 $\mathbf{d}_{\mathrm{shri}}$ 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 will be associated with the edges of original **P** (original edge: o-edge) which are identified by original edges' ID (e.g. *edg1* & *edg2* in Fig.2a), since at least three original edges sweep to form a node. Therefore, at each event when the node is generated, at least three o-edges will be linked to the

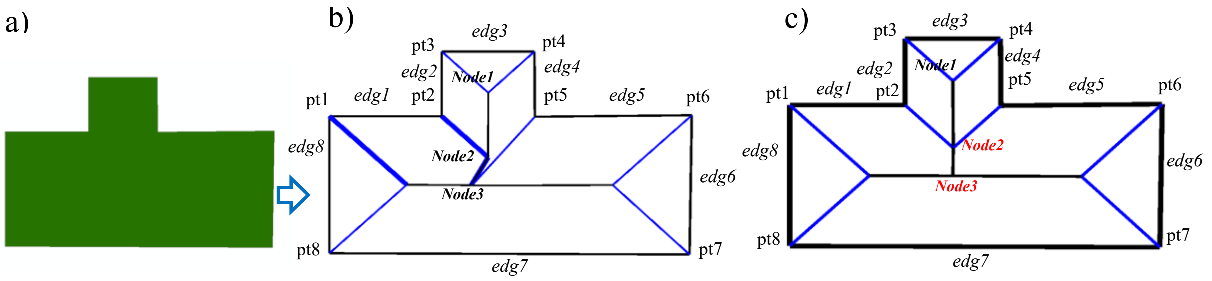


Figure 3: a) An orthogonal building polygon b) a monotone polygon with an acute angle c) rectified monotone polygons by '*Line segment*' event

node. This means more than three o-edges ID will be stored in the node with a suitable structure.

In our system, a node has the following properties; **(a)** '*Node Type*' (how the node is risen; by *Edge* event or *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 *Edge* event) or 'o-edge ID of one of the edge incident to the reflex vertex' (by *Split* event) **(d)** 'o-edge ID following the vanishing edge' (by *Edge* event) or 'o-edge ID of the other edge incident to the reflex vertex' (by *Split* event) **(e)** 'o-edge ID of at least one vanishing edge' (by *Edge* event) or 'o-edge ID of a split edge' (by *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. The system is looking for the node which has the same 'o-edge ID' as each original edge of **P** to form monotone polygons.

In special cases, four or more edges collapse into nodes, such as *Node2* in Fig.2 and *Node1,2,3* in Fig.4c. 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, and more than three 'o-edge IDs' are stored in the property of the 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. [Epp99] defined as a 'degenerate case' in which the straight skeleton can have vertices of degree higher 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 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 Fig.3. In Fig.3c, pt5 propagates to join pt2 and four edges (*edg1,2,4,5*) propagate into *Node2*, whereas, in Fig.3b, pt5 does not join pt2 and goes off *Node2*, and three edges (*edg1,4,7*) result in *Node3* with acute angle shape. This acute angle shape is also found at the

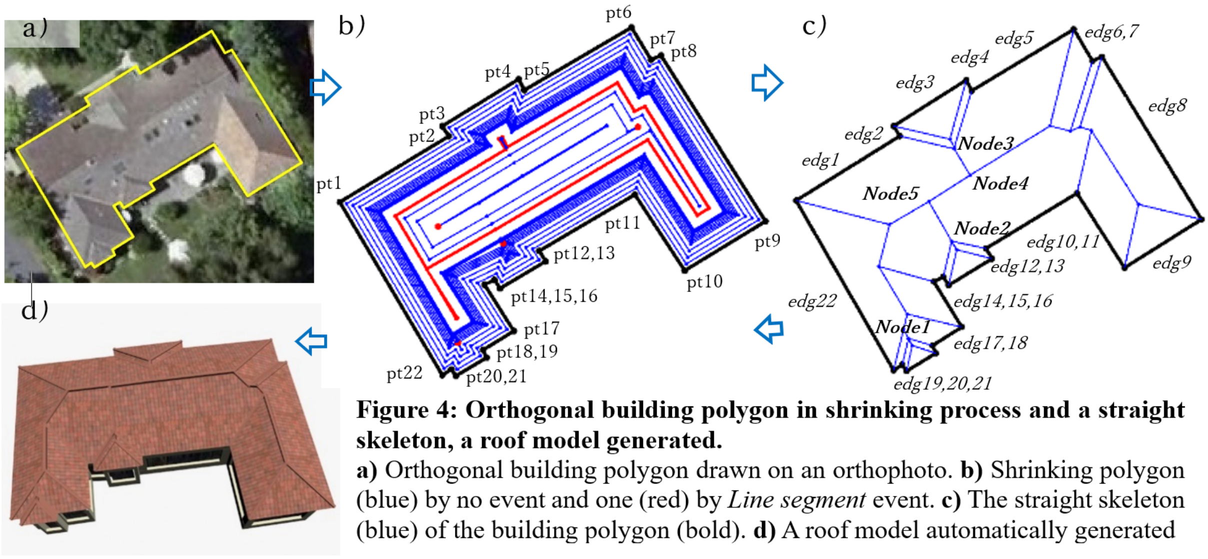


Figure 4: Orthogonal building polygon in shrinking process and a straight skeleton, a roof model generated.
a) Orthogonal building polygon drawn on an orthophoto. **b)** Shrinking polygon (blue) by no event and one (red) by *Line segment* event. **c)** The straight skeleton (blue) of the building polygon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

figure of Eppstein et al. [Epp99], 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 node 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 supposed to converge into the same line segment.

By a '*Line segment*' event, two parallel edges converge into one edge (line segment), and the convergent line segment will be 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 Fig.4c and Fig.5b; 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 Fig.4c & Fig.5b, 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 Fig.5a; three edges (*edg12,14,15*) result in *Node2*, and three edges (*edg11,12,15*) 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. [Aic95], 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. Fig.5c 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-intersection is found at *edg29* in Fig.5c and *edg21* in Fig.5b.

To avoid self-intersection, the azimuth angle of the nodes belonging to the same monotone polygon is proposed, where 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 Fig.5c, 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 one in Fig.4c and Fig.6c.

5 APPLICATION

Here are the examples of 3D building models automatically generated by the integrated system. Fig.6 & Fig.7 show the examples of 3D building models automatically generated. In generating these models, we classify the case of '*Line segment* event'. In *Split* event as mentioned in section 4.1, it is assumed to calculate the intersection of two non-parallel line

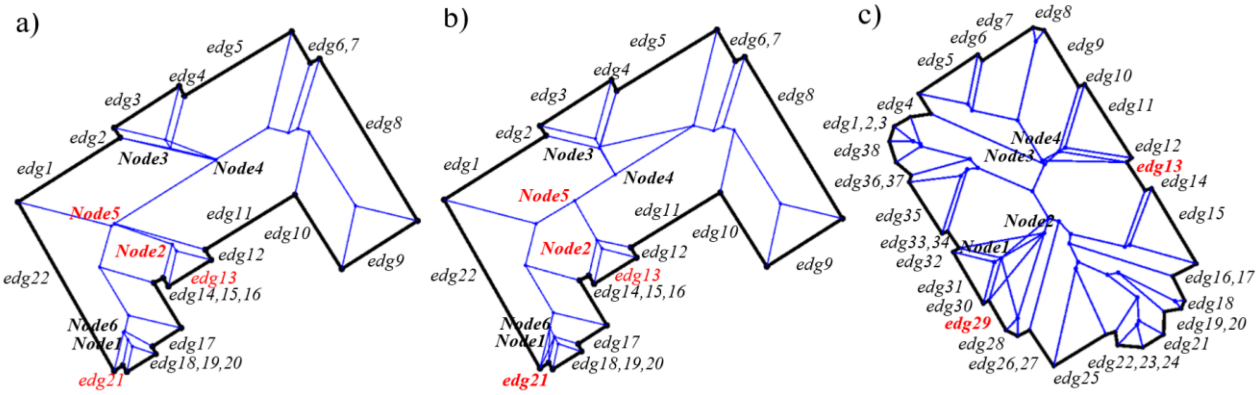


Figure 5: Monotone polygons with acute angle and self-intersecting monotone polygons. a) Some monotone polygons have acute angle if a *Line segment* event does not occur. **b) & c)** Some monotone polygons are self-intersecting if the set of the nodes are sorted according to the 'node vector projections'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

segments, i.e., two non-parallel angular bisectors. However, for some orthogonal polygons, two parallel edges will be overlapped when shrunk by e_**d**$_{shri}$, and two parallel angular bisectors will be overlapped. If we do not classify the case of *Line segment* event, then we end up with numerical error by trying to calculate the intersection of two parallel line segments.

Once 3D models with roofs are created, a top view of these models can be a roof report as shown in Fig.7i & 7j, which can be used for the rapid assessment of roof damages by insurance companies. Automated generation of simple and complex roof geometries will be utilized for rapid roof area damage reporting by the length measurements and area calculations of all roof surfaces. The roof board area will be easily calculated, since a roof board is a monotone polygon, and can be partitioned into a set of trapezoids or triangles. The roof board area will be calculated by adding these trapezoids, and subtracted or added by two triangles, depending on the shape of the monotone polygon.

The advantage of our generation system is that our 3D building models created are utilized for architectural design, i.e., BIM (Building Information Model), 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.

Now, architectural design world is experiencing a shift from 2D CAD drawings to BIM 3D modeling. BIM revolution is happening in the construction industry that is producing a step change in efficiency and accuracy. In our research, 3D building models automatically created can be used for BIM 3D modeling. There is no automatic generation system for 3D building solid models with complicated roofs as far as we know. Automatic generation will be compared with manual creation which are a series of manual operations mentioned in section 1 by 3ds Max, and broken down into functions of the program (Maxscript) described as CG module's process in section 3. It will take about more than one hour to create one hipped roof house including making intricately shaped ridges, while several seconds to automatically generate one by the personal computer. If given digital maps with attributes being inputted, as shown in Fig.1&7, the system automatically generates one hundred 3D building models within less than 10 minutes by the personal computer with Intel(R) Core(TM) i7-7820HK CPU 2.90GHz.

6 CONCLUSION

In this paper, the new and extended methodology is proposed for adding '*Line segment*' event besides the conventional events, and 'monotone polygon nodes sorting' by which self-intersecting monotone polygons are not formed. Thus, the proposed integrated system succeeds in automatically generating 3D building models.

The roofs created by the straight skeleton are limited to hipped roofs with their roof ridges parallel to nearby long edges of the building contour. However, there are many roofs whose ridges are perpendicular to long edges. In the residential area all over the world, there are many roofs the straight skeleton method cannot create. For example, in the middle of the top edge in Fig.4a and Fig.6a, Fig.7e, there are some branch roofs which are not slanting and drop 90 degrees vertically, i.e., gable roofs. These are not created by the straight skeleton.

In order to create various shape of roof, we propose a couple of schemes to create roofs by straight skeleton computation or partitioning or separating of orthogonal polygons. A complicated orthogonal building polygon can be partitioned or separated into a set of rectangles. Our proposed system partitions orthogonal polygons into a set of rectangles and places various shapes of roofs which include gable & hipped roofs or the roof whose ridges are perpendicular to nearby long edges on these partitioned rectangles. Thus, in order to create the 3D building models that have hipped and gable branch roofs, the future work is for developing the system in which we can select a couple of various schemes; dividing or separation

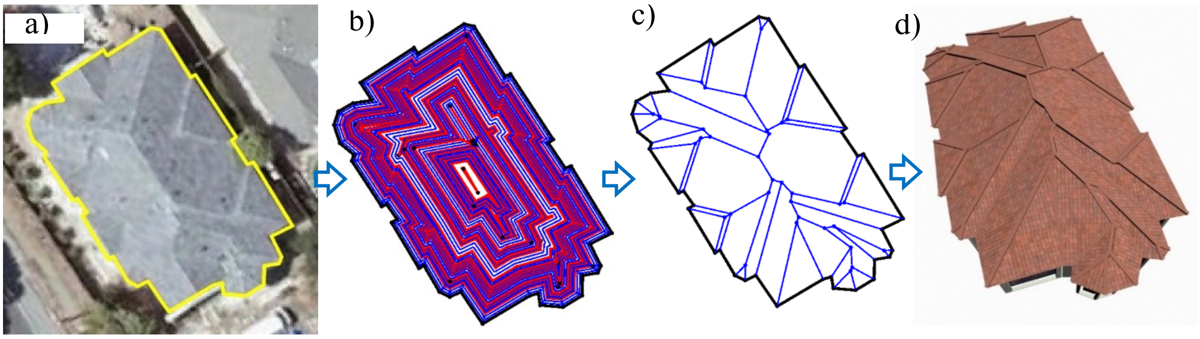


Figure 6: ***Complicated shape polygon in shrinking process and a straight skeleton, a roof model generated.***

a) Complicated building polygon drawn on an orthophoto. b) Shrinking polygons (blue) by no event and red ones by events. c) The straight skeleton (blue) of the building polygon (bold). d) A roof model automatically generated scheme or straight skeleton scheme to automatically create the different styles of roofs for one building footprint.

6 REFERENCES

[Aic95] Aichholzer O., Aurenhammer F., Alberts D., Gärtner B. A novel type of skeleton for polygons. Journal of Universal Computer Science (1995), 1 (12): 752–761.

[Ede14] Johannes Edelsbrunner, Ulrich Krispel, Sven Havemann, Alexei Sourin. Constructive Roof Geometry. Cyberworlds (CW) 2014,·DOI: 10.1109/CW.2014.17

[Epp99] Eppstein David, Erickson Jeff. Raising roofs, crashing cycles, and playing pool: applications of a data structure for finding pairwise interactions. Discrete and Computational Geometry (1999), 22 (4): 569–592

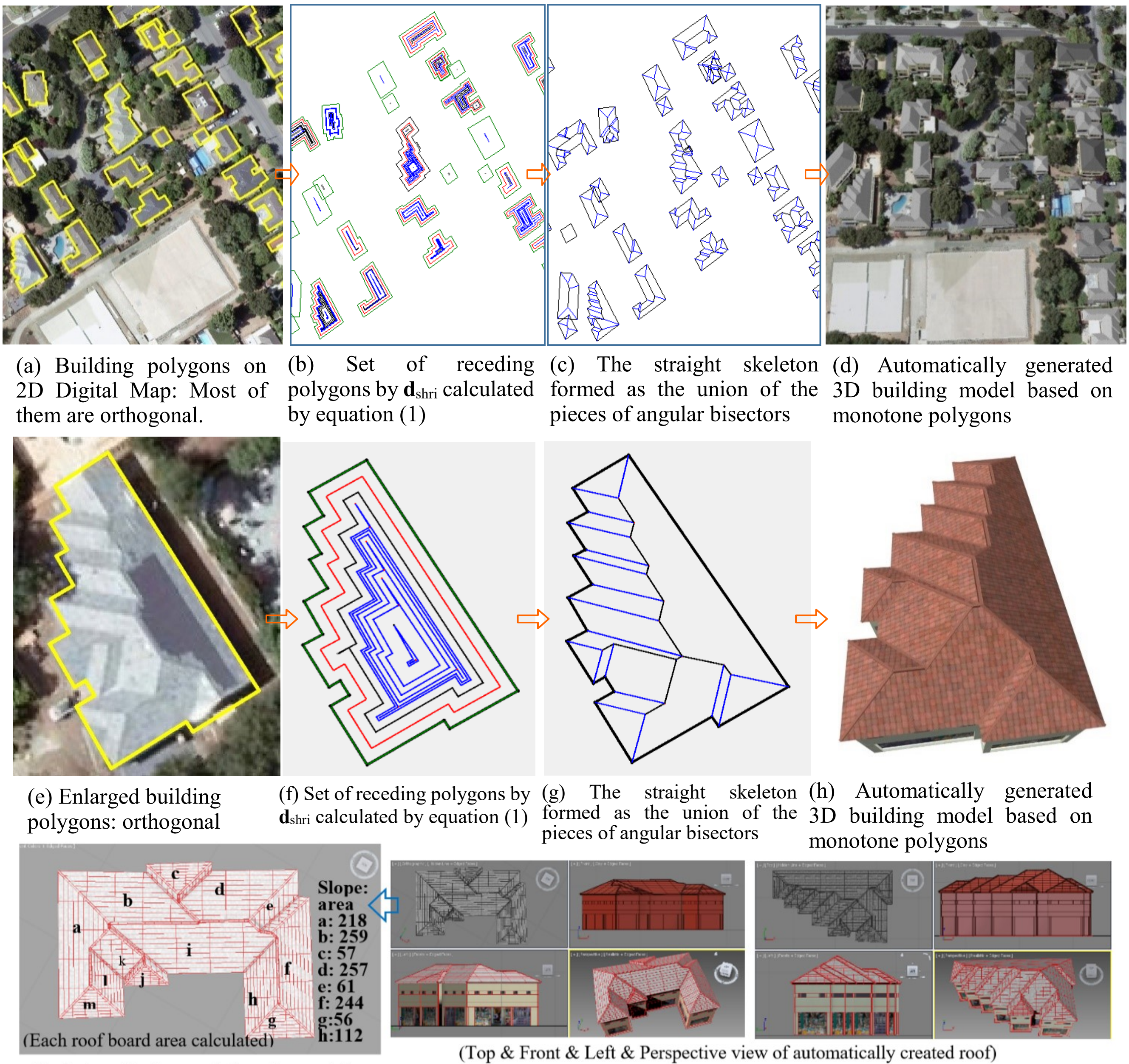
[Hel17] Helda M., Palfradera P. Straight Skeletons with Additive and Multiplicative Weights and Their Application to the Algorithmic Generation of Roofs and Terrains. Computer-Aided Design, Elsevier B.V. (Nov 2017), Vol 92, pp. 33-41

[Jia09] Jiang N., Tan P., Cheong L. F. Symmetric architecture modeling with a single image. Proc. SIGGRAPH Asia '09 (2009), papers No. 113

[Kel11] Kelly T., Wonka P. Interactive Architectural Modeling with Procedural Extrusions. ACM Transactions on Graphics (TOG) (2011), 30(2), 14-28.

[Mue06] Mueller P., Wonka P., Haegler S., Ulmer A., Gool L.V. Procedural modeling of buildings. ACM Transactions on Graphics (2006), 25, 3, 614–623.

[Par01] Parish Y. I. H., Müller P. Procedural modeling of cities. Proceedings of ACM SIGGRAPH (2001), ACM Press, E. Fiume, Ed., New York, 301–308



(a) Building polygons on 2D Digital Map: Most of them are orthogonal.

(b) Set of receding polygons by $\mathbf{d}_{\mathrm{shri}}$ calculated by equation (1)

(c) The straight skeleton formed as the union of the pieces of angular bisectors

(d) Automatically generated 3D building model based on monotone polygons

(e) Enlarged building polygons: orthogonal

(f) Set of receding polygons by $\mathbf{d}_{\mathrm{shri}}$ calculated by equation (1)

(g) The straight skeleton formed as the union of the pieces of angular bisectors

(h) Automatically generated 3D building model based on monotone polygons

(i) Automatically created roof report for damage evaluation

(Top & Front & Left & Perspective view of automatically created roof)

(j) Ground [floor] plan (Top view) of automatically created 3D building model

Figure 7: Shrinking process and a straight skeleton, a roof model generated, roof report