Straight Skeleton for Automatic Generation of 3-D Building Models with General Shaped Roofs

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

ABSTRACT

3D urban models are important in several fields, such as urban planning and gaming industries. However, enormous time and labor has to be consumed to create these 3D models, using a 3D modeling software such as 3ds Max or SketchUp. In order to automate laborious steps, a GIS and CG integrated system is proposed for automatically generating 3D building models, based on building polygons (building footprints) on digital maps. Digital maps shows most building polygons' edges meet at right angles (orthogonal polygon). In the digital map, however, not all building polygons are orthogonal. In either orthogonal or non-orthogonal polygons, the new system is proposed for automatically generating 3D building models with general shaped roofs by straight skeleton computation. In this paper, the algorithm for shrinking a polygon and forming a straight skeleton are clarified and, the new methodology is proposed for constructing roof models by assuming 'the third event' and, at the end of the shrinking process, the shrinking polygon is converged to 'a line of convergence'.

Keywords

3D urban model, automatic generation, GIS (Geographic Information System), 3D building model, straight skeleton.

1. INTRODUCTION

3D urban models, such as the one shown in Fig.1 right, are important in urban planning and gaming industries. Urban planners may draw the maps for sustainable development. 3D urban models based on these maps are quite effective in understanding what if this alternative plan is realized. However, enormous time and labour has to be consumed 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 use the following laborious steps:

(1) Generation of primitives of appropriate size, such as box, prism or polyhedron 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 holes in a building body for doors and windows **(3)** Rotation of parts of a house **(4)** Positioning of parts of a house **(5)** Texture mapping onto these parts.

In order to automate these laborious steps, a GIS (Geographic Information System) and CG integrated system was proposed for automatically generating 3D building models, based on building polygons or building footprints on a digital map shown in Fig. 1 left [Sug09]. A complicated orthogonal polygon can be partitioned into a set of rectangles. The proposed integrated system partitions orthogonal building polygons into a set of rectangles and places rectangular roofs and box-shaped building bodies on these rectangles. In order to partition an orthogonal polygon, a useful polygon expression (RL expression: edges' Right & Left turns expression) and a partitioning scheme was proposed for deciding from which vertex a dividing line (DL) is drawn [Sug12].

In the digital map, however, not all building polygons are orthogonal. In either orthogonal or non-orthogonal polygons, the new system is proposed for automatically generating 3D building models with general shaped roofs by the straight skeleton defined by a continuous shrinking process, which was introduced and discussed by Aichholzer et al.[Aic95]. However, some roof models are not created by their

 ISBN 978-80-86943-75-6

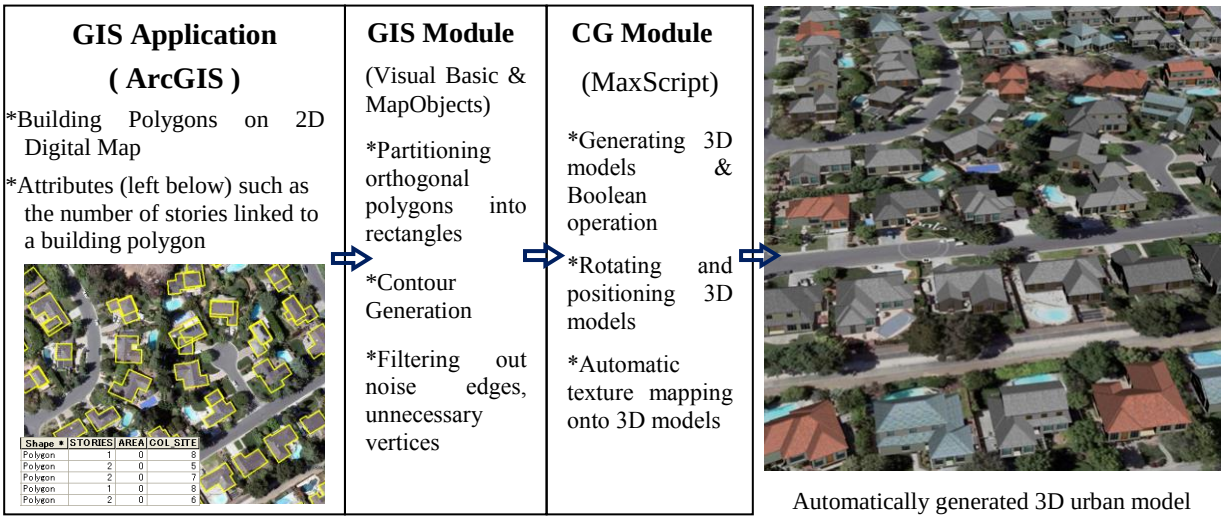


Figure 1. Pipeline of Automatic Generation for 3D Building Models

proposed method. In this paper, the new methodology is proposed for constructing roof models by assuming 'the third event' and, at the end of the shrinking process, the shrinking polygon is converged to 'a line of convergence'.

2. RELATED WORK

Since 3D urban models are important information infrastructure that can be utilized in several fields, the researches on creations of 3D urban models are in full swing. Various types of technologies, ranging from computer vision, computer graphics, photogrammetry, and remote sensing, have been proposed and developed for creating 3D urban models.

Using photogrammetry, Gruen et al. [Gru98, Gru02] introduced a semi-automated topology generator for 3D building models: CC-Modeler. Feature identification and measurement with aerial stereo images is implemented in manual mode. During feature measurement, measured 3D points belonging to a single object should be coded into two different types according to their functionality and structure: boundary points and interior points. After these manual operations, the faces are defined and the related points are determined. Then the CC-Modeler fits the faces jointly to the given measurements in order to form a 3D building model.

Suveg and Vosselman [Suv02] presented a knowledge-based system for automatic 3D building reconstruction from aerial images. The reconstruction process starts with the partitioning of a building into simple building parts based on the building polygon provided by 2D GIS map. If the building polygon is not a rectangle, then it can be divided into rectangles. A polygon can have multiple partitioning schemes. To avoid a blind search for optimal partitioning schemes, the minimum description length principle is used. This principle provides a means of giving higher priority to the partitioning schemes with a smaller number of rectangles. Among these schemes, optimal partitioning is 'manually' selected. Then, the building primitives of CSG representation are placed on the rectangles partitioned.

These proposals and systems, using photogrammetry, will provide us with a primitive 3D building model with accurate height, length and width, but without details such as windows, eaves or doors. The research on 3D reconstruction is concentrated on reconstructing the rough shape of the buildings, neglecting details on the façades such as windows, etc [Zla02].

On the other hand, there are some application areas such as urban planning and game industries where the immediate creation and modification of many detailed building models is requested to present the alternative 3D urban models. Procedural modeling is an effective technique to create 3D models from sets of rules such as L-systems, fractals, and generative modeling language [Par01].

Müller et al. [Mül06] have created an archaeological site of Pompeii and a suburbia model of Beverly Hills by using a shape grammar that provides a computational approach to the generation of designs. 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 [Aic95]. However, there is no digital map description and in-depth explanation about how the skeleton is formed and applied to create roofs in their paper.

More recently, image-based capturing and rendering techniques, together with procedural modeling approaches, have been developed that allow

 ISBN 978-80-86943-75-6

buildings to be quickly generated and rendered realistically at interactive rates. Bekins et al. [Dan05] exploit building features taken from real-world capture scenes. Their interactive system subdivides and groups the features into feature regions that can be rearranged to texture a new model in the style of the original. The redundancy found in architecture is used to derive procedural rules describing the organization of the original building, which can then be used to automate the subdivision and texturing of a new building. This redundancy can also be used to automatically fill occluded and poorly sampled areas of the image set.

Aliaga et al. [Dan07] extend the technique to inverse procedural modeling of buildings and they describe how to use an extracted repertoire of building grammars to facilitate the visualization and modification of architectural structures. They present an interactive system that enables both creating new buildings in the style of others and modifying existing buildings in a quick manner.

Vanega et al. [Car10] interactively reconstruct 3D building models with the grammar for representing changes in building geometry that approximately follow the Manhattan-world (MW) assumption which states there is a predominance of three mutually orthogonal directions in the scene. They say automatic approaches using laser-scans or LIDAR data, combined with aerial imagery or ground-level images, suffering from one or all of low-resolution sampling, robustness, and missing surfaces. One way to improve quality or automation is to incorporate assumptions about the buildings such as MW assumption. However, there are lots of buildings that have cylindrical or general curved surfaces, based on non-orthogonal building polygons.

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 [Nia09]. When creating 3D urban models for urban planning or facilitating public involvement, 3D urban models should cover lots of citizens' and stakeholders' buildings involved. This means that it will take an enormous time and labour to model a 3D urban model with hundreds or thousands of building.

Thus, the GIS and CG integrated system that automatically generates 3D urban models immediately is proposed, and the generated 3D building models that constitute 3D urban models are approximate geometric 3D building models that citizens and stakeholder can recognize as their future residence or real-world buildings.

3. PIPELINE OF AUTOMATIC GENERATION

As the pipeline of automatic generation is shown in Fig.1, the source of a 3D urban model is a digital residential map that contains building polygons. The digital maps are stored and administrated by GIS application (ArcGIS, ESRI Inc.). The maps are then preprocessed at the GIS module, and the CG module finally generates the 3D urban model.

To streamline the building generation process, the knowledge-based system was proposed for generating 3D model by linking the building polygons to information from domain specific knowledge in GIS maps: attributes data such as the number of storey and the type of roof.

Preprocessing at the GIS module includes the procedures as follows: **(1)** Filter out an unnecessary vertex whose internal angle is almost 180 degrees. **(2)** Partition or separate orthogonal building polygons into sets of rectangles. **(3)** Generate inside contours by straight skeleton computation for placing doors, windows, fences and shop façades which are setback from the original building polygon. **(4)** Form the straight skeleton for the general shaped roof. **(5)** Rectify the shape of the polygons so that there are no gaps or overlaps between geometric primitives such as rectangles. **(6)** Export the coordinates of polygons' vertices, the length, width and height of the partitioned rectangle, and attributes of buildings.

The attributes of buildings, shown in Fig.1 left below, consist of the number of storeys, the image code of roof, wall and the type of roof (flat, gable roof, hipped roof, oblong gable roof, gambrel roof, mansard roof, temple roof and so forth). The GIS module has been developed using 2D GIS software components (MapObjects, ESRI).

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 gradient of the edges of the partitioned rectangle. The CG module generates a box of the length and width, measured in GIS module.

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 gradient of the

 ISBN 978-80-86943-75-6

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 FOR ROOF GENERATION

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 unexpectedly 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:

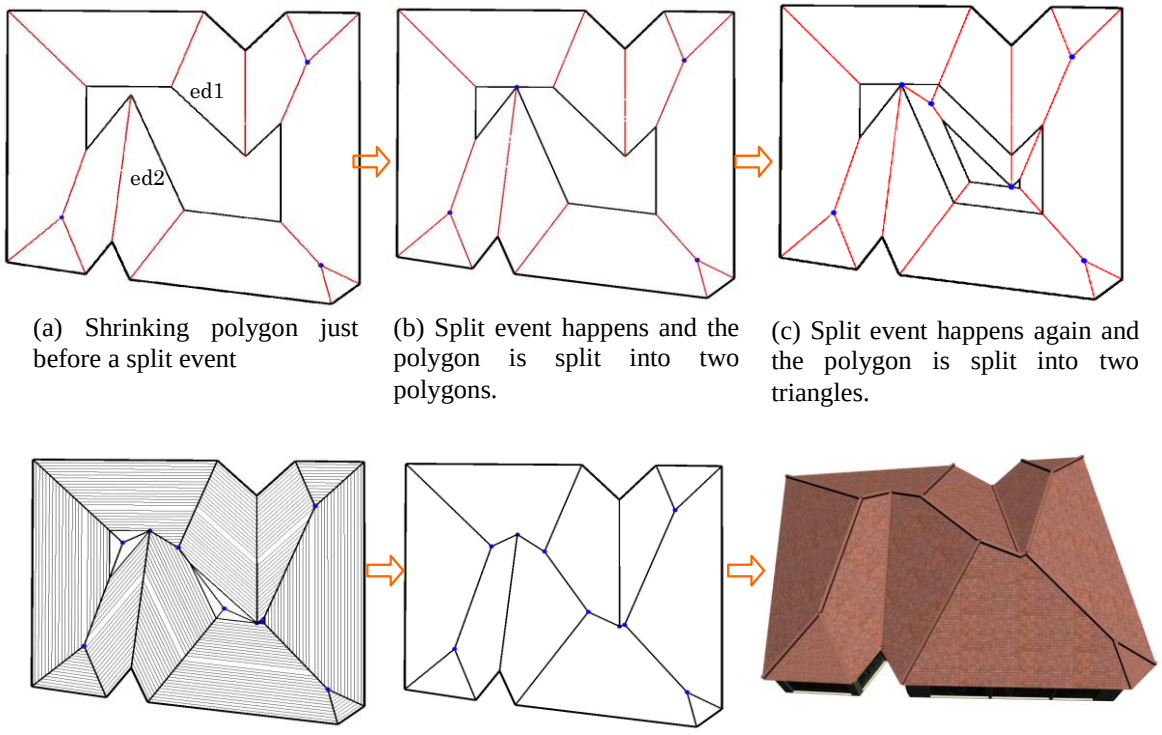
(1) ***Edge event***: *An edge shrinks to zero, making its neighboring edges adjacent now*.

(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.*

A reflex vertex is a vertex whose internal angle is greater than 180 degrees.

All edge lengths of the polygon do not always decrease during the shrinking process. Some edge lengths of a concave polygon will increase. For example, as shown by 'ed1' and 'ed2' in Fig.2(a), the edges incident to a reflex vertex will grow in length. 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:



(a) Shrinking polygon just before a split event

(b) Split event happens and the polygon is split into two polygons.

(c) Split event happens again and the polygon is split into two triangles.

(d) Polygons shrinking at a constant interval: nodes by an edge event: nodes by a split event: nodes by a collapse of a triangle to a point.

(e) The straight skeleton defined as the union of the pieces of angular bisectors traced out by polygon vertices during the shrinking process

(f) A roof model automatically generated: each roof board is based on an 'interior monotone polygon' partitioned by straight skeleton

Figure 2. Shrinking process and a straight skeleton, a roof model automatically generated

 ISBN 978-80-86943-75-6

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

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.

Fig.2 from (a) to (c) show a shrinking process for a non-orthogonal concave polygon: the polygon just before a split event: the polygon being split into two polygons after a split event happens. Fig.2(d) shows a set of polygons shrinking at the constant interval and nodes by an edge event and a split event, and nodes by a collapse of a triangle into a point.

Fig.2(e) shows the straight skeleton defined as the pieces of angular bisectors traced out by polygon vertices. Fig.2(f) shows the roof model automatically generated. Since the straight skeleton partitions the interior of a polygon with n vertices (n-gon) into n monotone polygons, each roof board that constitutes the roof model is formed based on these partitioned 'interior monotone polygons'.

4.1. ALGORITHM for STRAIGHT SKELETON

Fig.3 shows the overall outline pseudo-code for the straight skeleton computation by split & edge event and collapse of a triangle to a node. At first, one simple polygon (**P**) is given such as shown in Fig.2. If there is any reflex vertex in the **P**, then it can be divided into two or more polygons.

At four lines from the top of the code, the system calculates $e\_\mathbf{d}_{shri}$ for all edges and finds the shortest of them. Then, the system checks if 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 $e\_\mathbf{d}_{shri}$ found. In the process, a split event may happen and the polygon will be divided into some polygons: **P**s. In the upper half of the code (split event process), all divided polygons are checked if they can be divided more. As long as there is some **P**s that can be divided, split event will continue. After that, the system concentrates on the edge event procedure.

In the split event process, during shrinking to the shortest $e\_\mathbf{d}_{shri}$, the system checks if a line segment of an angular bisector from a reflex vertex intersects another edge of the polygon or not. If an edge is found intersected, the system calculates the node position by the split event. However, one edge will be intersected by several 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 position.

After any type of event happens and the polygon changes topologically, there remains one or more new split polygons which are shrunk recursively if they have non-zero area. At that moment, the system recalculates the length of each edge and internal angles of each vertex in order to find the shortest $\mathbf{d}_{shri}$ for next events.

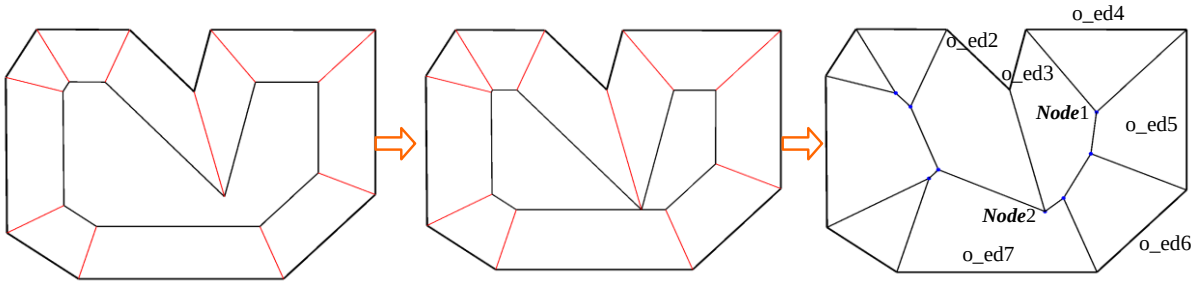
In the code, the **P** has members: 'split event finish flag' (sp_ev_fin_fl) and 'edge event finish flag' (ed_ev_fin_fl) which indicate whether or not the **P** can be processed by 'split event' or 'edge event' respectively, during the shrinking process. If 'sp_ev_fin_fl' is set for the **P**, then the **P** is finished with split event checking. If 'sp_ev_fin_fl' is reset, then the **P** will be checked whether split event is happened or not.

In the upper half of the algorithm, if at least one possibly divided **P** remains unchecked for 'split event', then 'SplitEventLoopFinish_flag' will be reset and the system cannot get out of the 'while loop'. After all **P**s have been checked for 'split event', then all **P**s are checked only for 'edge event' and then 'triangle' procedure for nodes generation as shown in the lower half of the algorithm.

```
While (Event procedure is not finished for all split P) {
 While ('SplitEventLoopFinish_flag' == reset) {
  For all one or more split P: { If (P. sp_ev_fin_fl == reset) {
   Find the shortest e_dshri of the P.
   Check if Split Event occurs by increasing dshri by (e_dshri
    /n_step).
   If (Angular bisector from a reflex vertex intersects
    another edge) {
     Calculate the node position by Split Event.
    }
  }

   For all one or more split P:{ If (P. sp_ev_fin_fl == reset)
     Reset 'SplitEventLoopFinish_flag'
   }
 } /* For " While ('SplitEventLoopFinish_flag' == reset){" */

   For all one or more split P:{ If (P. ed_ev_fin_fl == reset) {
     Find the shortest e_dshri of the P; Shrink P by e_dshri;
      Calculate the node position by Edge Event.}
  }


  For all one or more split P: { If (P is a triangle) {
      Calculate the node position of the triangle.
      Associate the node with the original edge. }
  }
} /* For "While (Event procedure is not finished for all split P) { " */
```

Figure 3. Algorithm for forming straight skeleton by Split & Edge event and Collapse of a triangle to a node

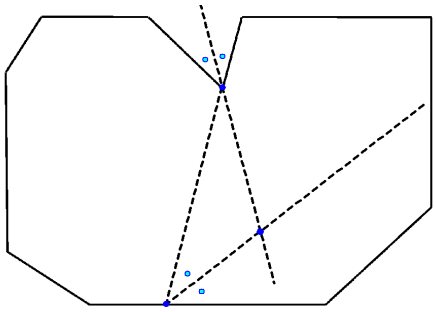
 ISBN 978-80-86943-75-6



(a) Shrinking polygon just before a split event

(b) Split event happens and the polygon is split into two

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



(d) The position of the node is given 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.

Figure 4. How a split event happens, and how the position of the node is calculated

The generated nodes will be associated with the edges of original **P** (original edge: o-edge), since at least three original edges sweep to form the node. Therefore, at each event when the node is generated, at least three o-edges will be linked to the node. When a square or a regular hexagon collapses to a node, four or six o-edges will sweep into a node. This is the case of degeneration.

The third event as 'the simultaneous one' is processed at 'edge event', since the other split polygon disappears into a node in this event. After detecting the split event and edge event have occurred simultaneously, the system deals with the event and links the generated node to three o-edges.

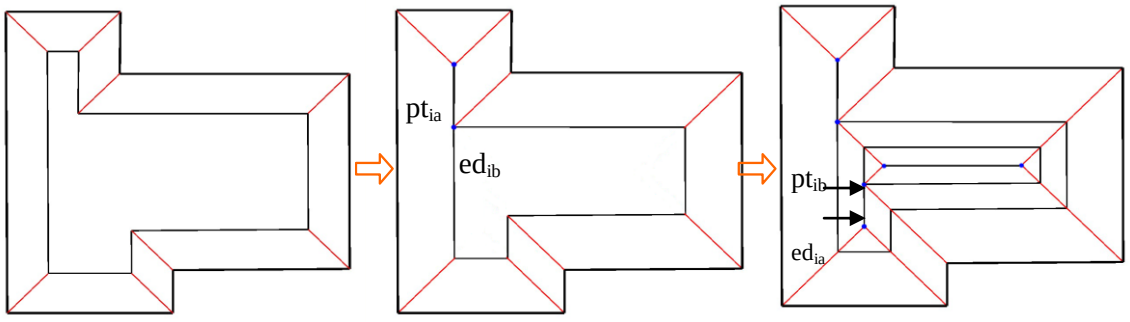
4.2. HOW MONOTONE POLYGONS are FORMED

Fig.4(c) shows how these interior monotone polygons are formed. When a shrinking process starts, the edges of the polygon sweep inwards from their original edge (o_ed_i). Nodes arise from the edge event or the split event. For example, *Node*1, arisen by the edge event, is the convergent point into which consecutive three original edges (from o_ed3 to o_ed5) sweep. On the other hand, *Node*2, arisen by the split event, is the intersection point between the angular bisector from the reflex vertex (between o_ed2 & o_ed3) and the intersected edge (o_ed7). Since at least three original edges (o_ed_i) sweep into a node, the node keeps information about which o_ed_i makes up the node itself. In order to form monotone polygons, following the original edge one by one, the system searches which node has the same original edge number. For example, o_ed4 has an only one node (*Node*1) that has the same original edge number, whereas o_ed3 has four nodes that have the same original edge number including such as *Node*1, *Node*2. The nodes belonging to each o_ed_i are sorted according to the coordinate value on the axis parallel to each original edge (o_ed_i) vector. These nodes are coplanar, and will form a roof board for a 3D building model.

Fig.4(d) shows how a split event happens, and how the position of the node arisen by the split event is calculated. The position of the node is given by the intersection of two angular bisectors: one from the reflex vertex and the other bisector between the intersected edge and one of two edges incident to the reflex vertex.

For some polygons in Fig.5 showing a shrinking process of an orthogonal polygon, the event different from the two events mentioned will happen. In this research, it is proposed to add the third event in which a reflex vertex runs into the edge, but the other split polygon is collapsed into a node since an edge event happens in the split polygon at the same time. This event happens at an orthogonal part of the polygon as shown in Fig.5. In the process, as shown in Fig.5(b) & (c), the system detects 'the third event' by checking if pt_{ia} (vertex) is on ed_{ib} (edge) or pt_{ib} is on ed_{ia} where pt_{ia} & pt_{ib} are the vertices next to two vertices coherent by the edge event, and ed_{ia} & ed_{ib} are the edges adjacent to these two coherent vertices.

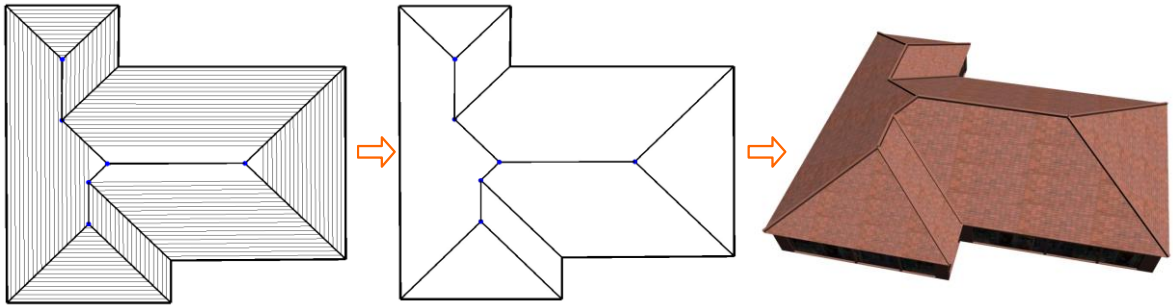
 ISBN 978-80-86943-75-6



(a) Shrinking polygon just before a third event

(b) A third event happens and the polygon is split into one node and one polygon.

(c) A third event happens again and the polygon is split into one node and one polygon



(d) Polygons shrinking at a constant interval: nodes by an edge event & a third event: nodes by a collapse of a rectangle to a line.

(e) The straight skeleton defined as the union of the pieces of angular bisectors traced out by polygon vertices during the shrinking process

(f) A roof model automatically generated: each roof board is based on an 'interior monotone polygon' partitioned by straight skeleton

Figure 5. Shrinking process and a straight skeleton for third events

Aichholzer et al. [Aic95a] demonstrated three edge events let a triangle collapse to a point in the last stage of each split polygon as shown in Fig.2(d). In this paper, it is proposed to add the case in which two edge events let a rectangle collapse to a line segment ('a line of convergence') in the last stage, a rectangle whose opposite sides have the same and the shortest e_$\mathbf{d}_{shri}$.

Since a line segment does not have area, it is not shrunk anymore. The central area of an orthogonal polygon in Fig.5(d) shows a line of convergence to which the shrinking polygon (rectangle) is converged.

5. APPLICATION AND CONCLUSION

Here are the examples of 3D building models automatically generated by the integrated system. Fig.6 shows the examples of 3D building models automatically generated by the straight skeleton computation from non-orthogonal building polygons. To ease the discussion, Aichholzer et al. [Aic95a] exclude degeneracies caused by special shapes of polygon, e.g., a regular polygon. In this paper, we deal with the degenerate cases in which more than three edges are shrunk to a point. Ideally, simultaneous n edge events cause a regular n-gon to collapse to a point but it is difficult to draw such a perfect regular n-gon. Accordingly, the system rectifies the shape of the regular n-gon so as to let n edge events at the same time. Fig.6 center shows the 3D dodecagon building model automatically generated based on the degeneracy of 12 edges being shrunk to one node.

Fig.7 shows proposed digital map for the town and an automatically generated 3D urban model: town houses with doom roofs created by straight skeleton computation in the middle of the image.

For everyone, a 3D urban model is quite effective in understanding what if this alternative plan is realized, what image of a sustainable city will be. Traditionally, urban planners design the city layout for the future by drawing building polygons on a digital map. Depending on the building polygons, the integrated system automatically generates a 3D urban model so instantly that it meets the urgent demand to realize another alternative urban planning for sustainable development.

 ISBN 978-80-86943-75-6

Figure 6. Non-orthogonal building footprints and 3D building models automatically generated by straight skeleton computation

If given digital maps with attributes being inputted, as shown in 'Application' section, the system automatically generates two hundreds 3D building models within less than 30 minutes.

In either orthogonal or non-orthogonal building polygons, the new system is proposed for automatically generating general shaped roof models by the straight skeleton computation. In this paper, the algorithm for 'forming straight skeleton by split & edge event' is clarified and the new methodology is proposed for constructing roof models by assuming the third event in addition to two events and, at the end of the shrinking process, some rectangles are converged to a line of convergence. Thus, the proposed integrated system succeeds in automatically generating alternative city plans.

The limitation of the system is that automatic generation is executed based only on ground plans or top views. There are some complicated shapes of buildings whose outlines are curved or even crooked. To create these curved buildings, the system needs side views and front views for curved outlines information.

Future work will be directed towards the development of methods for the automatic generation algorithm to model curved buildings by using side views and front views.

6. REFERENCES

[Aic95a] Aichholzer, O., Aurenhammer, F., Alberts, D., and Gärtner, B.: 'A novel type of skeleton for polygons', Journal of Universal Computer Science, 1 (12): 752–761 (1995).

[Car10a] Carlos, V. A., Daniel, A. G., and Bedřich, B.: 'Building reconstruction using Manhattan-world grammars', Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on: 358 – 365 (2010)

[Dan05a] Daniel, B. R., and Daniel, A. G.: 'Build-by-number: rearranging the real world to visualize novel architectural spaces', Visualization, 2005. VIS 05. IEEE, 143 – 150 (2005)

[Dan07a] Daniel, A. G., Paul, R. A., and Daniel, B. R.: 'Style Grammars for interactive Visualization of Architecture', Visualization and Computer Graphics, IEEE Transactions on Volume:13, 786 – 797 (2007)

[Gru98a] Gruen, A., Wang, X.: 'CC-Modeler: A topology generator for 3-D city models', ISPRS Journal of Photogrammetry & Remote Sensing, Vol.53, No.5, pp.286-295 (1998).

[Gru02a] Gruen, A., and et al.: 'Generation and visualization of 3D-city and facility models using CyberCity Modeler', MapAsia, 8, CD-ROM (2002)

[Mül06a] Müller, P., Wonka, P., Haegler, S., Ulmer, A., and Van Gool, L.: 'Procedural modeling of buildings', ACM Transactions on Graphics, 25, 3, 614–623 (2006)

[Nia09a] Nianjuan, J. P. T., and Loong-Fah, C.: 'Symmetric architecture modeling with a single image', ACM Transactions on Graphics - TOG, vol. 28, no. 5 (2009)

[Par01a] Parish, I. H. Y., and Müller, P.: 'Procedural modeling of cities', Proceedings of ACM SIGGRAPH 2001, ACM Press, E. Fiume, Ed., New York, 301–308 (2001)

[Sug09a] Kenichi SUGIHARA: "Automatic Generation of 3D Building Models with Various Shapes of Roofs", ACM SIGGRAPH ASIA 2009, Sketches DOI: 10.1145/1667146.1667181 (2009)

[Sug12a] Sugihara, K. and Kikata, J.: "Automatic Generation of 3D Building Models from Complicated Building Polygons", Journal of Computing in Civil Engineering, ASCE (American Society of Civil Engineers), DOI: 10.1061/(ASCE)CP.1943-5487.0000192 (2012)

[Suv02a] Suveg, I., and Vosselman, G.: 'Automatic 3D Building Reconstruction', Proceedings of SPIE, 4661, 59-69 (2002)

[Zla02a] Zlatanova, S., and Heuvel Van Den, F.A.: 'Knowledge-based automatic 3D line extraction from close range images', International Archives of Photogrammetry and Remote Sensing, 34, 233 – 238 (2002)

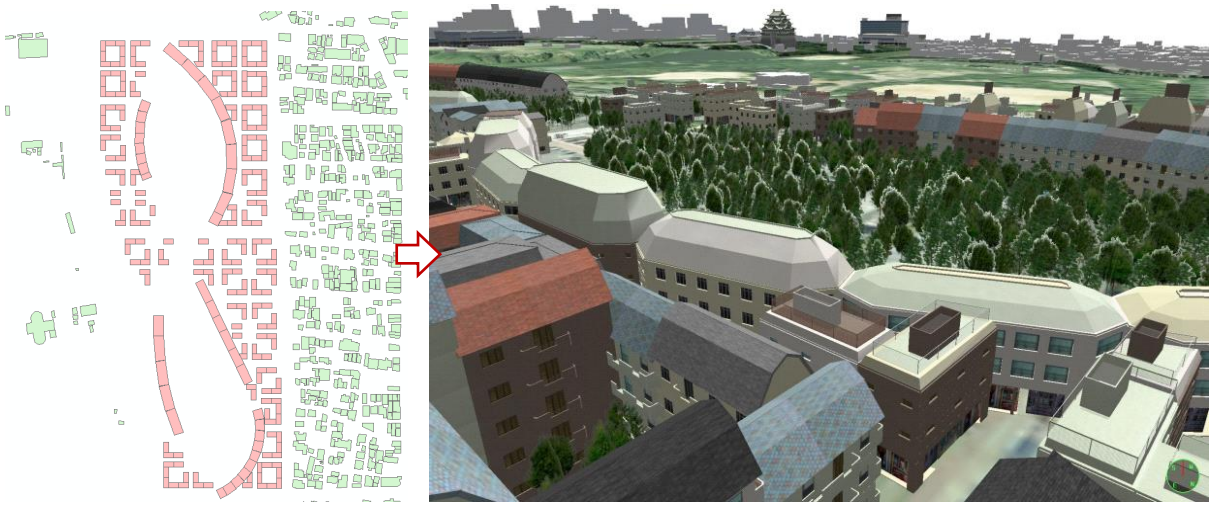


Figure 7. Proposed digital map for the town and an automatically generated 3D urban model: town houses with doom roofs created by straight skeleton computation in the middle of the image

 ISBN 978-80-86943-75-6