3-D BUILDING MODEL AUTOMATICALLY GENERATED FROM BUILDING CONTOUR PARTITIONED

Kenichi Sugihara
Professor
Faculty of Business Administration
Gifu Keizai University
5-50 Kitagata-chou Ogaki
Gifu 503-8550
Japan
Tel: +81-584-77-3511
Fax: +81-584-77-3598
E-mail: sugihara@gifu-keizai.ac.jp

Yoshitugu HAYASHI
Professor
Graduate School of Environmental Studies
Nagoya University
Furou-chou Chikusa-ku Nagoya
Aichi 464-8603
Japan
Tel: +81-52-789-2772
Fax: +81-52-789-3837
E-mail: yhayashi@genv.nagoya-u.ac.jp

Abstract: Based on features on digital map, such as building contour and road polyline, we are proposing the system to generate 3-D Urban Model automatically, integrating GIS and CG. 3-D Urban Model is the important information infrastructure that can be utilized in several fields, such as landscape evaluation, urban planning, civil engineering, architecture, disaster prevention simulation, etc. When a real urban world is projected into 3-D virtual space, buildings are major objects in this space. To realize 3-D CG urban model, it is important to generate building models efficiently. In this paper, we propose the GIS and CG integrated system to generate 3-D building models automatically from building polygons on the digital map stored by GIS. Some building polygons have a lot of vertices and are complicated in shape. We present the algorithm for breaking down complicated polygon into primitive ones. Building models are automatically generated from the building polygon partitioned by this algorithm. In the last chapter, we show proposed 3-D urban models automatically generated for urban planning.

Keywords: GIS, CG, 3-D Urban Model, 3-D building model, Automatic Generation, Building Polygon, Building Contour, Partitioning Method,

3-D BUILDING MODEL AUTOMATICALLY GENERATED FROM BUILDING CONTOUR PARTITIONED

1. Introduction

When a real urban world is projected into 3-D virtual space, buildings are major objects in this space. To realize 3-D urban model, it is important to generate building models efficiently. 3-D urban model is the important information infrastructure that can be utilized in several fields, e.g., landscape evaluation, city planning, architecture, disaster prevention simulation, etc. In addition, disclosure of information about public projects to the public in order to encourage their participation in city planning is a new application area where 3-D urban model can be of great use. However, in order to realize 3-D urban model, enormous time and labor have to be consumed to acquire the spatial data and to design the models. For example, when manually modeling a house with roofs by Constructive Solid Geometry (CSG), one must follow these laborious steps :
1) generation of primitives of appropriate size, such as box, prism or polyhedron that will form parts of a house 2) boolean operation among these primitives to form the shapes of a roof and a house body 3) rotation of parts of a house 4) positioning of parts of a house 5) texture mapping to these parts

In order to save the steps mentioned above, we aim at creating 3-D building models automatically from buildings' contours, i.e., building polygons on 2-D digital map. In our system, the sources of 3-D building models are 2-D digital maps stored and administrated by GIS. 2-D digital maps are made by map production companies. Building polygons on digital map are drawn manually with digitizer, depending on the aerial photos. In case of city planning, urban designer may follow the traditional way of drawing building polygons for the future layout of a city.

Building polygons can be extracted from aerial images by image recognition software. Aerial images are also important sources of 3-D building models. However, due to the complexity of natural scenes and the lack of performance of image recognition software, the fully automated methods cannot guarantee the results stable and reliable enough for practical use (Gruen and Wang, 1998). Recently, many approaches for automated and semi-automated extraction and modeling of buildings from aerial images have been proposed (Gruen et al., 2002 ; Suveg and Vosselman, 2002 ; Fischer et al., 1997).

Gruen and Wang (1998) introduced a semi-automated topology generator for 3-D building model: CC-Modeler. The feature identification and measurement with aerial images is implemented in manual mode. During feature measurement, measured 3-D 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 CC-Modeler fit the faces jointly to the given measurements in order to form 3-D building model.

Suveg and Vosselman (2002) presented a knowledge-based system for automatic 3D building reconstruction from aerial images. The reconstruction process starts with the partitioning of a building in 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 in rectangles. A building 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.

Fischer and et al. (1997) proposed a model-based approach to 3D building reconstruction. The reconstruction step starts with manual finding building corners in the images. Hypothesized 2D building corners of different stereo images are manually matched, leading to 3D building corners. For verification, the 3D building templates are projected back into the different images, resulting in 2D views for buildings. The predicted building templates and their interrelationships are then matched with the extracted image features and their relations. The successful sequence of matching steps lead to 3D generation of buildings.

These proposals and systems will provide us primitive 3D building model with accurate height, length and width, without details such as windows, eaves and doors. The researches on 3D reconstruction are concentrated on reconstructing the rough shape of the buildings neglecting details on the facades such as windows, doors, etc(Zlatanova and Heuvel, 2002).On the other hand, there are some application areas such as urban planning and disaster prevention simulation where the immediate creation and modification of many plausible building models is requested to present the alternative 3D urban models. When generating these models, automatic modelling of many future buildings is to be desired. In these models, approximate 3D building models with details are sometimes necessary to encourage the public involvement so that people in general can recognize them as buildings of alternative urban plans. Usually and traditionally, urban planners design the future layout of a town by drawing the map. So, it is convenient for the urban planners who draw maps if the map can immediately be converted into 3-D urban model. In our research, we aim at creating 3-D building model, especially building models automatically from 2D building contours on digital map. The system automatically generates 3-D urban model so quickly that it meets the urgent demand to realize another alternative plan.

2. Flow of the Automatic Generation System

Our automatic generation system consists of GIS application (ArcView, ESRI Inc.), GIS module and CG module as shown Figure 1. The source of 3-D urban model is a digital residential map that contains building polygons linked with attributes data such as the number of story, the type of roof. The residential maps with attributes are produced by map makers, consultant companies or urban designers.

GIS module pre-processes building polygons on the digital map. Pre-process includes filtering out 'noise edge' (short edge that is between long edges of almost the same direction) and unnecessary vertices of a building polygon,

dividing a complicated polygon into primitive ones, generating inside contours to form walls and glasses of a building and exporting the coordinates of polygons' vertices and attributes of buildings. The attributes of buildings consist of the number of story, the image code of roof, wall and the type code of roof (flat, gable roof, hipped roof, gable roof with longer width, gambrel roof and penthouse). GIS module has been developed using 2-D GIS software components (MapObjects, ESRI Inc).

CG module receives the pre-processed data that GIS module exports, generating 3-D building models. CG module has been developed using Maxscript that controls 3-D CG software (3D Studio VIZ, Autodesk Inc).

In case of modelling a building with roofs, CG module follows these steps:

1) Generation of primitives whose size are determined by the data preprocessed.

2) Boolean operation among these primitives to form the shapes of a roof, a building body, etc.

3) Rotation of parts of a building

4)Positioning of parts of a building

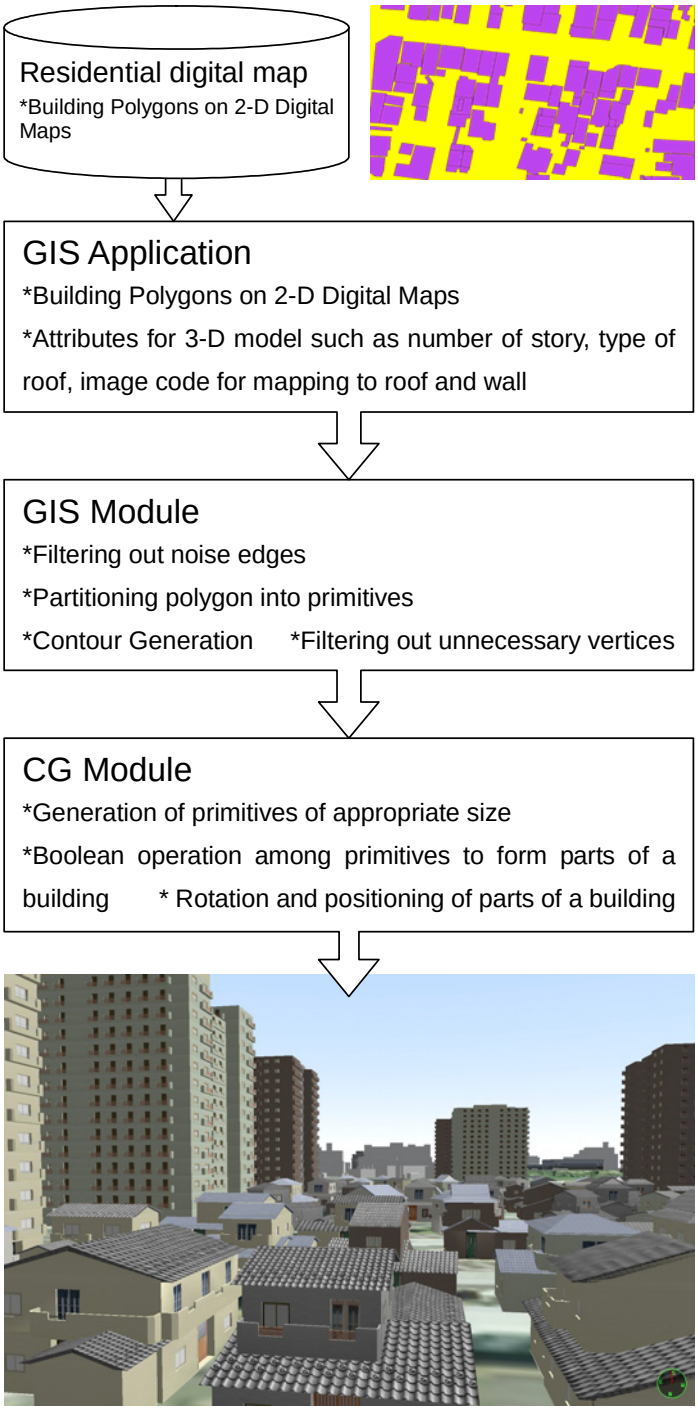

Figure 1 Flow of the Automatic Generation System

5) Texture mapping to these parts according to the attribute received.

6) Copying the 2nd floor to form 3rd floor or more in case of more than 3 stories building, depending on the number of story received.

With the rapid progress of remote sensing and photogrammetrical measurement technology, we can get huge amount of 3-D point cloud data that are not structured and to be recognized as a certain object on the ground. On the other hand, in GIS, point collection are recognized and structured by human power to

form polygons or polylines according to geometry, topology and semantics. Our system executes the process of generating 3-D building model after receiving the data that are recognized as building polygon from GIS application. In the generation process, the system estimates and reconstructs the 3-D shape of alternative building model for the real, based on the shape of the polygon and attributes linked to the polygon.

3. The GIS Module

3.1 Main Functions of GIS Module

In terms of geometric point of view, ‘objects’ are differently classified between GIS and CG. In GIS, ‘objects’ is defined in the sense of database object (Molenaar, 1998) while, in CG, it is defined as an abstraction that models the state and behavior of entities in a system (Schroeder, 1998). Since objects in GIS are related to geometric elements, objects can be treated as the spatial entities that have geometric elements. From the geometric point of view, in GIS world, there are only a few objects such as point, polyline and polygon, while, in CG world, there are many geometric objects (called primitive) such as box, prism, cylinder, sphere, plane, cone, etc. In GIS world, rectangle, trapezoid, pentagon, hexagon and ellipse that may form a building contour are classified as only one category, that is, polygon. Therefore, it is necessary to recognize and pre-process building polygons on 2-D digital map for 3-D CG model generation. If a building polygon has only almost 90 or 270 degrees angles, it can be replaced by the combination of rectangles (2-D primitives) and subdivided into rectangles so that boxes and prisms are placed on them in CG generating process.

Main functions of GIS module are as follows.

1) Filtering out ‘noise edge’.

2) Replacing building polygon that has limited number of vertices and only almost 90 or 270 degrees angles by a combination of rectangles.

3) Subdividing building polygon into rectangles and L-shape polygon if building polygon is replaced by the combination of rectangles.

4) Generating inside and outside contour in order to form the wall and the glass of a building.

5) Filtering out the vertex which has almost 180 degree angle (we call ‘flat point’).

6) Filtering out the vertex which is almost overlapped with another vertex.

7) Estimation of the center and the minor and major axes of the ellipse that will be placed on a circular polygon.

8) Exporting coordinates and attribute data of building polygons from 2-D GIS to CG module.

3.2 Filtering of 'noise edge'

Satellite and aircraft equipped with CCD cameras and electromagnetic sensors are daily monitoring and taking photos of earth's surface. Thanks to the advancement of remote sensor technology, we can get satellite and aerial imagery easily. The digital image processing and pattern recognition research of these photos are in full swing. Some image recognition software provides us image object extraction as shown in Figure 2 (eCognition, Definiens Inc.). On the other hand, at map production companies, many technicians are drawing digital residential maps. Building polygons shown in Figure 3 on residential map are drawn manually with digitizer, depending on the aerial photos. Since the images that recognition software provides contain too much noise to be filtered out, we use residential maps for the source of 3-D urban model.

Figure 2 Building contours extracted from aerial photo by pattern recognition software (eCognition, Definiens Inc.)

Figure 3 Building polygons on digital residential map drawn by map producer

These maps show that edges of a building polygon are not always straight lines. Many building polygons contain short edges that are between long edges of almost the same direction. We call short edge 'noise edge'. Noise edge exists because of the shape of roofs or eaves of a house or jaggies of image recognition system. However, inside the noisy building polygon, there is a combination of boxes whose contours are not jagged. In other words, under the complicated shape of roofs or eaves, there may exist a simple box-combined house. After filtering out noise edges, core polygons are given inside original polygons. CG and GIS integrated system will generate 3-D roof model from original building polygon and 3-D house body model from core polygon filtered.

The process for filtering out noise edge is executed as follows:

1) The process tries to find the range where normal edges (longer than threshold length) of almost the same direction are in sequence. We call the range 'same direction range'

2) Within the same direction range, the average direction of edges is calculated. Since the most inner vertex within the range is in the most right side towards the average direction, the inner product between unit vector orthogonal to the average direction and edge vector is calculated, deciding the most inner vertex. That leads to the most inner line with the average direction in each range.

3) The intersections between the most inner lines in each range are calculated. The set of these intersections will form core polygon.

The algorithm for filtering out noise edge is as follows: In Figure 4, vertices (v_i) of a polygon are numbered clockwise. An edge which connects vertices v_i and v_j is denoted by e_{ij}. As for noise edge, e.g., e_{23} is the noise edge that is between long edges (e_{12}, e_{34}) of almost the same direction.

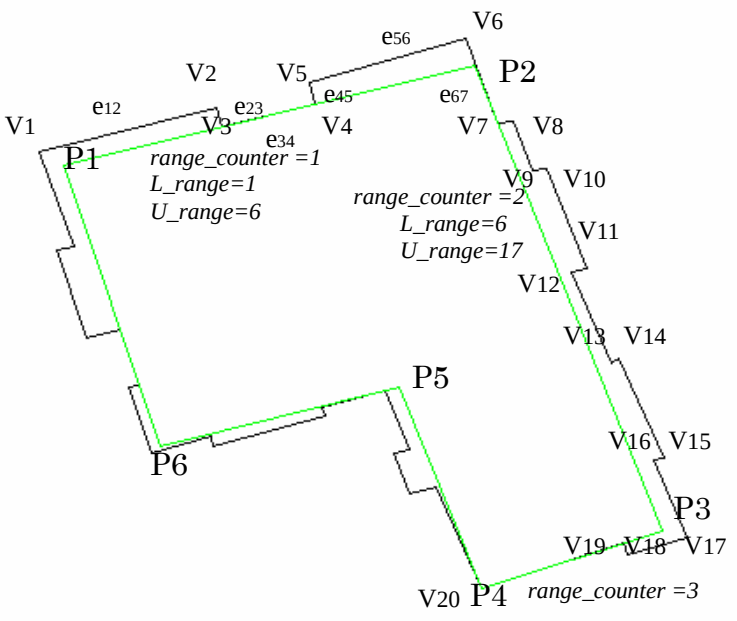

Figure 4 Noise edge filtering of building polygon and forming of core polygon

According to the algorithm mentioned in Figure 5, a program tries to find out the lower limit of the same direction range (L_range) and the upper limit of the same direction range (U_range). A set of successive edges, (e_{12}, e_{23}, e_{34}, e_{45}, e_{56}) is the same direction range where normal edges of almost the same direction are in sequence. Since this is the first range, L_range is v_1, U_range is v_6 and range_counter=1. In this range, v_4 is the most inner vertex because it is in the most right side towards the average direction calculated. When a program finds out that e_{67} is a normal edge with the different direction, L_range, U_range and range_counter will be updated. In the range (range_counter=2), the edges e_{78}, e_{9_10}, e_{11_12}, etc. is noise edges.

A set of successive edges, (e_{67}, e_{78},… , e_{16_17}) is the same direction range that starts at $_{v6}$ and ends at v_{17}. In this range, v_7 is the most inner vertex. Then, the most inner line is determined. The intersection P2 between the line (range_counter: 1) and the line (range_counter: 2) is calculated. Thus, the core polygon made up of P1 to P6 is formed.

GIS application stores and administrates building polygons with noise edges. GIS module executes filtering out noise edges and CG module receives these pre-processed data. Based on the position of vertices of original building polygon and core polygon and attribute data, CG module will generate 3-D roof model from original building polygon and 3-D house body model from core polygon. The process is shown in Figure 6.

Algorithm: Filtering of noise edge

Let the suffix of vertices of a polygon numbered clockwise.

Threshold length ('Lth') is the length to distinguish between normal edge and noise edge.

The same direction range has lower limit of the range(L_range) and upper limit of the range(U_range). L_range and U_range are the number of vertex.

Let 'range_counter' the counter for range. The range_counter counts the number of 'same direction range'.

1) Calculate the length of edge one by one clockwise.
2) If the length of edge > Lth then
3) Calculate the direction of edge and compare it with previous direction.
4) If calculated direction == previous direction then U_range is updated.
5) Else L_range and U_range and range_counter is updated. // calculated direction != previous direction
6) End if
7) Else U_range is updated. // This is the case of ' length of edge < Lth'; Noise edge.
8) End if
9) For i=1 to range_counter
10) At the each range ('same direction range'), the average direction of edges is calculated and the most inner vertex is decided. Then, the most inner line with the average direction is decided. The most inner vertex is in the most right side towards the average direction calculated. The inner product between unit vector orthogonal to the average direction and edge vector decides the vertex in the most right side.
11) Next i
12) For i=1 to range_counter
13) The intersections between the most inner lines are calculated.
14) Next i
15) The set of intersections forms 'core polygon'.

Figure 5 Algorithm for noise edge filtering of building polygon and forming of core polygon

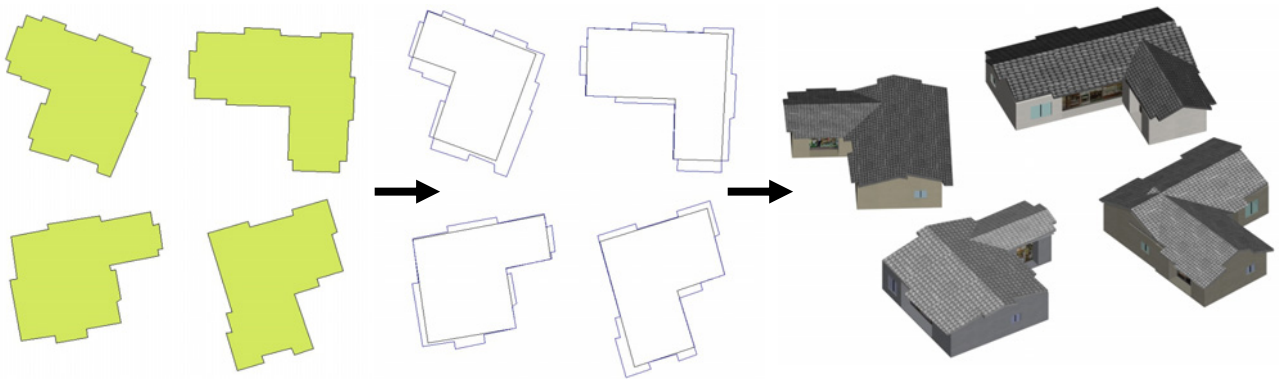

Figure 6 Building polygons that contain noise edges (Left). Core polygons that are filtered out noise edges (Center). Automatically generated 3-D house model (Right). 3-D roof model is generated from original building polygon and 3-D house model from core polygon

3.3 Proposed Polygon Representation

After filtering out noise edges of a building polygon, the number of vertices of a polygon is reduced. For most of building polygons filtered, the angles of vertices of polygon are almost 90 or 270 degrees and the number of vertices is small. Such a building polygon can be replaced by a combination of rectangles in GIS module. When following edges of a polygon clockwise, an edge turns to the right or to the left by 90 degrees. So, it is possible to assume that the building polygon with right angle only can be expressed as a set of its edges' turning direction. The polygon with 10 vertices (10 vertices polygon) shown in the Figure 7 is expressed as a set of its edges' turning direction, i.e., RLRRLRRRLR where R and L mean a change of an edge's direction to the right and to the left, respectively. For the building polygon that has only right angle and can be described as RL expression, the following relationship stands up among the number (Num.) of vertices, the number of right turn edges and the number of left turn edges.

The number of shapes that a polygon can take depends on the number of vertices that a polygon has. By the formula of circular permutation, we can calculate the number of the patterns of shape that a polygon may take. In case of a polygon with 6 vertices (6 vertices polygon), the edges' direction set of the polygon is LRRRRR. Since the left turn edge appears only once in 6 vertices polygon, the shape pattern is unique, that is, L-shape. This L-shaped polygon is the basic polygon into which a polygon with more vertices is divided.

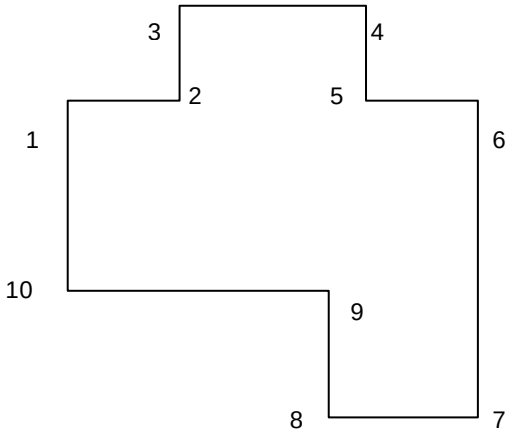

Figure 7 Building polygon is expressed as a set of its edges' turning direction ; RLRRLRRRLR

In case of 8 vertices polygon, the following four kinds of polygon shape pattern are possible. LLRRRRRR, LRLRRRRR, LRRLRRRR, LRRRLRRR.

Four cases can be calculated by the following formula of circular permutation.

No reiteration pattern $\left(\frac{8!}{6!2!}-\frac{4!}{3!1!}\right)\div 8=3$ 4 reiteration patterns $\frac{4!}{3!1!}\div 4=1$

Total number of cases: 3 + 1 = 4 cases

As for 10 and 12 vertices polygon, the number of shape pattern is also calculated by the same formula as shown below. In proportion to the number of vertices, the number of shape pattern will increase by geometric progression.

(1) 10 vertices polygon: $\frac{10!}{7!3!}\times\frac{1}{10}=12$ cases

(2) 12 vertices polygon:

3 reiteration patterns $\frac{3!}{2!1!}\div 3=1$ 6 reiteration patterns $\left(\frac{6!}{4!2!}-\frac{3!}{2!}\right)\div 6=2$ No reiteration pattern $\left(\frac{12!}{8!4!}-\frac{6!}{4!2!}\right)\div 12=40$

Total number of cases: 1 + 2 + 40 = 43 cases

The advantage of this RL expression is as follows.

1) The shape of a polygon with right angle only can be described as simple RL expression. That means RL expression specifies the shape of a polygon.

2) This expression decides from which vertex a dividing line is drawn. From ‘L’ vertex, a dividing line is drawn so that a polygon with more than 8 vertices is broken down into rectangles and L-shaped polygon. This is explained in the next section.

3.4 How to Divide Polygon

RL expression is useful in deciding the vertex from which a dividing line is drawn. In order to break down a polygon with more than 8 vertices into rectangles and L-shaped polygon, a dividing line is drawn from ‘L’ vertex. As for the polygon consists of more than 8 vertices, the polygon will be divided into a central area and attached branches. The polygon with right angle only is supposed to be expressed as a dataset of turning direction of its edges. In case that a dataset take L (Left turn) after or before consecutive R (Right turn), we assume this pattern as a branch. In other words, we take notice of the vertex that turns conversely. From this vertex, the dividing line is drawn. For example, ‘*RRL*’ pattern is recognized as a branch. From ‘L’ vertex, a dividing line is drawn to the backward direction in terms of vertices numbered clockwise. Also, ‘*LRR*’ pattern is recognized as a branch. From ‘L’ vertex, a dividing line is drawn to the forward. These dividing line are sure to have intersections with one of ‘RR’ edges, resulting in dividing into rectangles and the rest of a polygon as branches and a central area.

At first, we have applied this dividing algorithm to 8 vertices polygon. Since 8 vertices polygon takes four types of shape pattern according to RL expression, the algorithm has been applied to four types respectively. Figure 8 shows three types that have no reiteration pattern and two types that have reiteration pattern. In case of the type without reiteration, the module looks up the vertex that turns to the left (‘L’) after the consecutive vertex that turns to the right (‘RR’).

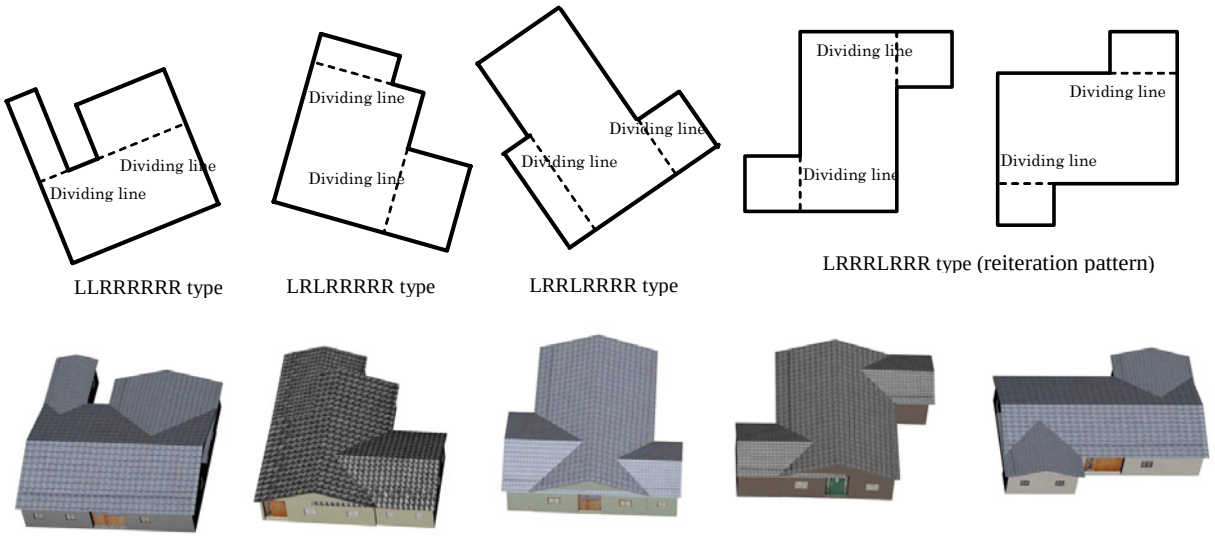

Figure 8 8 vertices polygon divided into a central area and branches (upper), 3-D building model automatically generated from the polygon divided(below)

From this vertex, the dividing line is drawn to the backward direction. From the next vertex that turns to the left (‘L’), the dividing line is drawn to the forward direction. In the type that has reiteration pattern of ‘LRRR’, there are two shape patterns as shown Figure 8. These two patterns cannot be distinguished by the RL expression. We have to consider the length of the preceding edges of ‘L’ vertex. We distinguish two patterns by the sum of the length of two edges preceding ‘L’ vertex. From two ‘L’ vertices, dividing lines is drawn to the same direction.

After dividing into branches and the rest of a polygon, the coordinates of 6 vertices of the polygon are provided as a set of coordinates of 6 vertices polygon. The algorithm for assigning to L-shaped polygon is applied to this set of coordinates.

Figure9 shows that the dividing algorithm mentioned is applied to 8 vertices polygon. The houses at first and second column from left belong to ‘LRRRLRRR’. The houses at third column belong to ‘LRLRRRRR’. The houses at fourth column belong to ‘LLRRRRRR’. The houses at right end column belong to ‘LRRLRRRR’. The houses at first row from the bottom have a gable roof. The houses at second row have a hipped roof. The houses at third row have a gable roof whose width is longer than roof length.

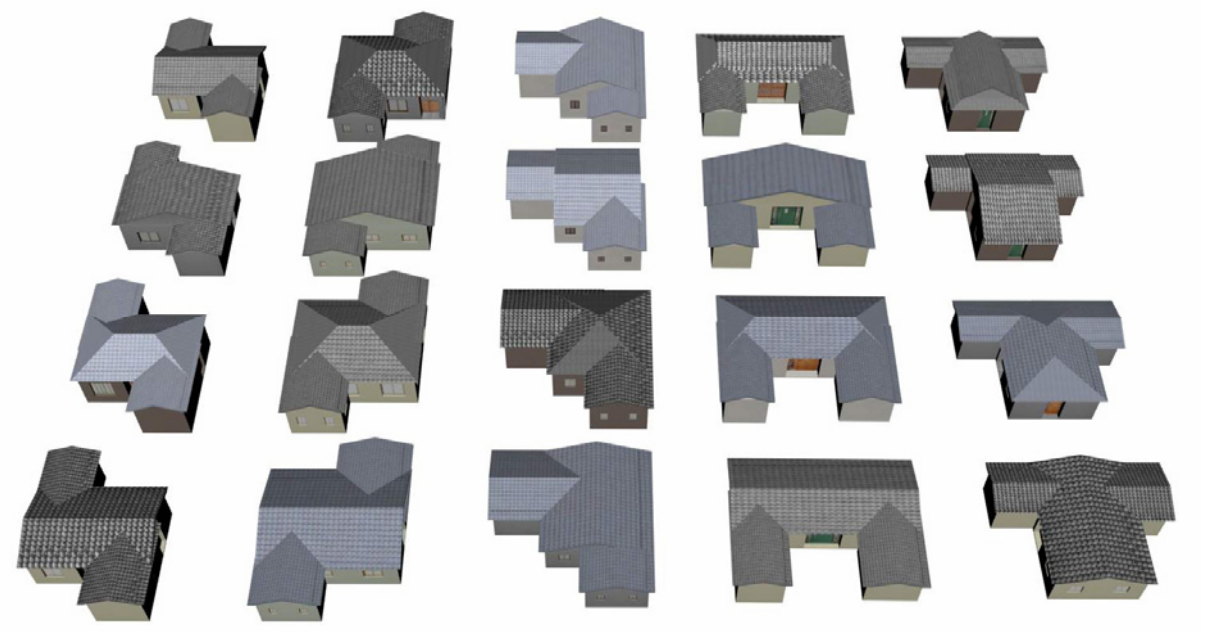

Figure 9 3-D building models with roofs generated from 8 vertices polygons

4. Application of the system and Conclusion

3-D urban model reflecting the real 3-D world offers us the important tool to display the results of simulation of urban planning, disaster prevention and public works projects. For residents, citizen or even students as well as the specialist of urban planning, 3-D urban model is quite effective in understanding what will be built, what image of the town will be or what if this alternative plan is realized. This model can act as the simulator to realize the alternative ideas of urban

planning virtually or to examine the current zoning system, building regulations and building agreements.

Here is the example of proposed 3-D urban model produced by the automatic generation system. Figure 10 shows aerial photo of the central area of Ogaki city where is declining because of motorization. There is a castle, but it is surrounded by buildings and not outstanding. Figure 11 shows 3-D urban model of present central area of Ogaki city. Figure 12 shows proposed 3-D urban model where the castle is surrounded by the 2-story houses with roofs so as to be seen from the area around the castle. The road to the castle is widened so that people can easily reach the castle. In 3-D urban model, top roof of the castle is the gambrel roof that is a combination of gable roof and hipped roof.

The roof of 2nd floor is formed by copying the roof of 1st floor and setting the appropriate value to its height property.

Figure10 Aerial photo of Ogaki castle, Ogaki city, Japan (Chunichi press, 2000)

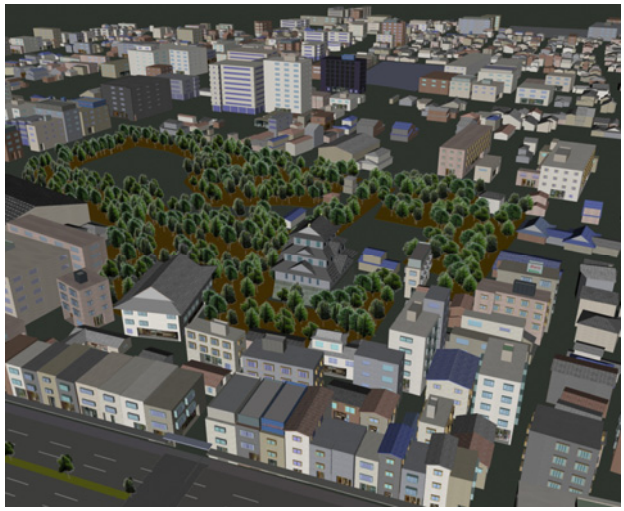

Figure11 Automatic generated 3-D Urban Model simulating the area around the Ogaki castle

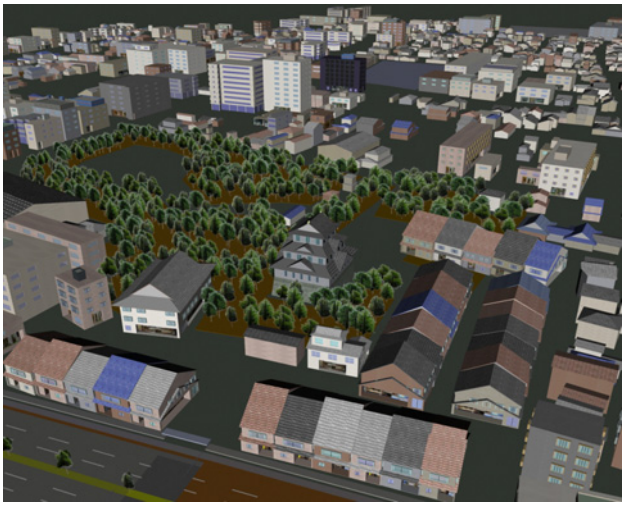

Figure12 Proposed 3-D Urban Model (Ogaki castle surrounded by 2-story houses so as to be seen from the area around the castle)

Figure13 Proposed 3-D Urban Model

(The river that restores its natural meander plays an important role for the revitalization of a city.)

Figure14 Aerial photo of Meijo area, Nagoya city, Japan (MapCube, 2003)

Figure15 Automatic generated 3-D Urban Model simulating present Meijo area

Figure16 Proposed 3-D Urban Model (3 story terrace houses stand sharing green, sunshine and open space in courtyard)

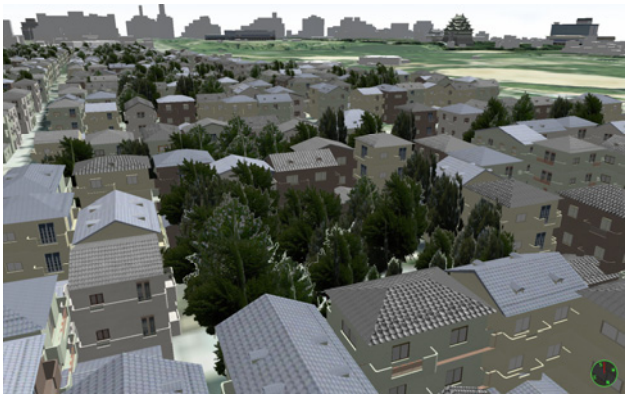

Figure17a Proposed 3-D Urban Model from lower angle

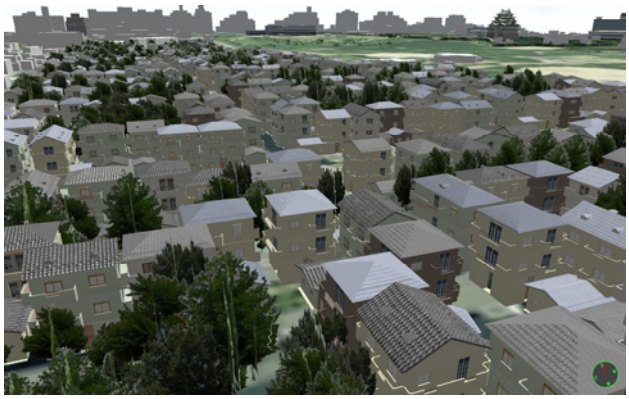

Figure17b Proposed 3-D Urban Model from lower angle

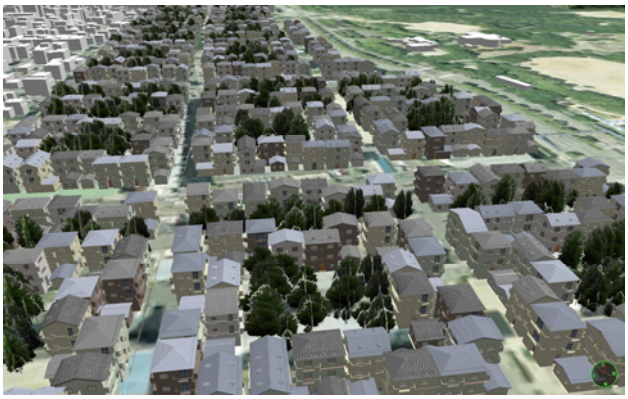

Figure17c Proposed 3-D Urban Model from lower angle

Figure18 Proposed 3-D Urban Model (5 story apartment blocks stand sharing green, sunshine and open space in courtyard)

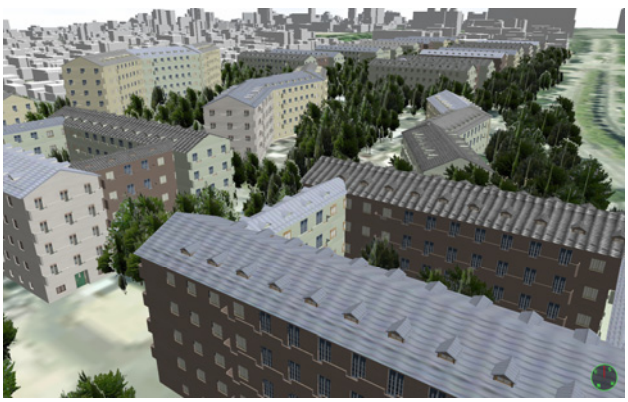

Figure19 Proposed 3-D Urban Model from lower angle

Figure 14 shows aerial photo of Meijo area of Nagoya city where urban revitalization plan will be implemented. This area is located at a center of Nagoya city 2 million people living. There is one of the largest castles in Japan which forms historical and cultural origin in central Japan. Figure 15 shows 3-D urban model of present Meijo area of Nagoya city. Figure 16 and Figure 17 show proposed 3-D urban model where 3 story terrace houses stand sharing green, sunshine and open space in courtyard. Figure 18 shows proposed 3-D urban model where 5 story apartment blocks stand sharing green, sunshine and open space in courtyard.

In this paper, we proposed the system that automatically generates 3-D building model from noisy building polygons. By applying noise edge filtering algorithm and then polygon dividing algorithm, we can get core polygon with smaller vertices, which then can be divided into basic polygon and rectangles. After breaking down into L-shaped polygon and rectangles, the module is placing primitives i.e. building parts on these polygons to form 3-D building model.

Future work will be directed towards the development of methods for:

1) importing elevation data from DEM or some other source of remote sensing in stead of using the number of story inputted by consultant company

2) dividing algorithm that will be applied to 10 or more vertices polygon, although the number of vertices of most polygon filtered on digital map is less than 10

3) generating building with balcony or veranda whose building body is made from core polygon and whose balcony is from original building polygon

References

Aerial photo of Ogaki castle, Ogaki city (2000) in Furusato Aerial photo collection, Chunichi press.

Fischer, A., Kolbe,T. H., Lang, F. (1997) Integration of 2D and 3D Reasoning for Building Reconstruction using a Generic Hierarchical Model, Workshop on Semantic Modeling for the Acquisition of Topographic Information from Images and Maps SMATI, 5/97, 1-21

Gruen, A. and Wang, X. (1998) CC Modeler: A topology generator for 3-D city models, ISPRS J. of Photogrammetry and Remote Sensing, 53, 286-295.

Gruen, A. and et al. (2002) Generation and visualization of 3D-city and facility models using CyberCity Modeler, MapAsia, 8.

Molenaar, M. (1998) An Introduction to the Theory of Spatial Object Modelling, 2-3, London: Taylor&Francis Ltd.

Schroeder, W., Martin, K., Lorensen, B. (1998) Visualization Toolkit 2nd Editions, 619-619, NJ07458 USA, Prentice Hall PTR.

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

Zlatanova, S. and Heuvel van den, F.A. (2002) Knowledge-based automatic 3D line extraction from close range images, International Archives of Photogrammetry and Remote Sensing, 34, 233 - 238