Automatic Generation of a 3D Terrain Model From Key Contours

Kenichi Sugihara
Information Media
Gifu Keizai University
Ogaki-city, Japan
sugihara@gifu-keizai.ac.jp

Takahiro Murase
Health Science
ChukyoGakuin University
Mizunami-city, Japan
murase@chukyogakuin-u.ac.jp

***Abstract*— In our research, a 3D terrain model is automatically generated from key contour polygons by straight skeleton computation. Based on key contours given elevation, the faces connecting key contours are automatically formed, and thus 3D terrain models are automatically created. The proposed system performs physical simulations, using a more realistic environment, such as an automatically generated 3D town model placed on a 3D terrain model, and mass granular flow based on Newtonian mechanics.**

Keywords—3D terrain model; straight skeleton; automatic generation; granular flow simulation; 3D town model;

I. INTRODUCTION

Mountainous countries are prone to natural disasters, especially mass granular flow disasters, such as, slope failure, avalanche and debris flow. In field investigations, researchers have to study mass flow disasters after the event has taken place, because direct measurements during the time of occurrence are difficult or even impossible as the work is too dangerous [1]. In addition, due to the stochastic nature of their occurrence and magnitude, it is very difficult to collect detailed data on mass granular flow. Due to these difficulties, numerical and physical simulations are effective measures to predict and control mass granular flow. Numerical simulations focus on exploring the initiation and propagation mechanisms of mass granular flow. Furthermore, the topography plays an important role in controlling the post failure motion of granular materials, which is believed to be essential in understanding the dynamic motion and destructive power of mass granular flow [2].

In our research, the key contours on a 2D map is converted into the topography, i.e., a realistic 3D terrain model. The proposed system performs a physical simulation using a more realistic environment, such as an automatically generated 3D town model placed on a 3D terrain model, and mass granular flow based on Newtonian mechanics. The construction of 3D town models first requires that 3D terrain models are modified for land formation and site preparation, then 3D house models can be put on these sites. However, using 3D modelling software such as 3ds Max or SketchUp, enormous time and labor must be expended to manually create these 3D models. In order to automate laborious steps, we are proposing a GIS (Geographic Information System) and CG integrated system for automatically generating 3D town models, based on contour polygons of a digital map [3],[4]. In our proposal, based on key contour polygons given elevation, the faces connecting key contours are automatically formed, and thus 3D terrain models are automatically created.

3D terrain models can be created from map contours or iso-lines. In our research, key contours are drawn and given a certain elevation, and then the faces connecting key contours are automatically formed, using the straight skeleton computation defined by a continuous shrinking process [5]. Based on the created faces, 3D terrain models are generated, and then 3D house models are placed on these prepared sites. Our methodology saves the laborious steps of drawing contours. The uniqueness of our system is the simulation of mass granular flow applied to realistic automatically generated 3D town models built on a 3D terrain model as shown in Fig. 1, In the proposed 3D model-based simulation, any formation of mass granular layout is possible, depending on 3D terrain models and what kind of flow simulation will be implemented, such as avalanche or debris flow.

II. RELATED WORK

Since 3D town models are important information infrastructure that can be utilized in several fields, e.g., urban planning, landscape evaluation, and disaster prevention simulation, the researches on creations of 3D town 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 town models.

Recently, using an unmanned aerial vehicle (UAV) commonly known as a drone to make 3D models becomes very popular. 3D data acquisition and object reconstruction can be implemented, using stereo images taken by drones. The block of overlapped images are analyzed through stereo photogrammetry, generating 3D models with texture mapping. These techniques such as computer vision, photogrammetry, and remote sensing will provide us with a primitive 3D building model with accurate height, length and width, but without details such as windows, eaves or doors.

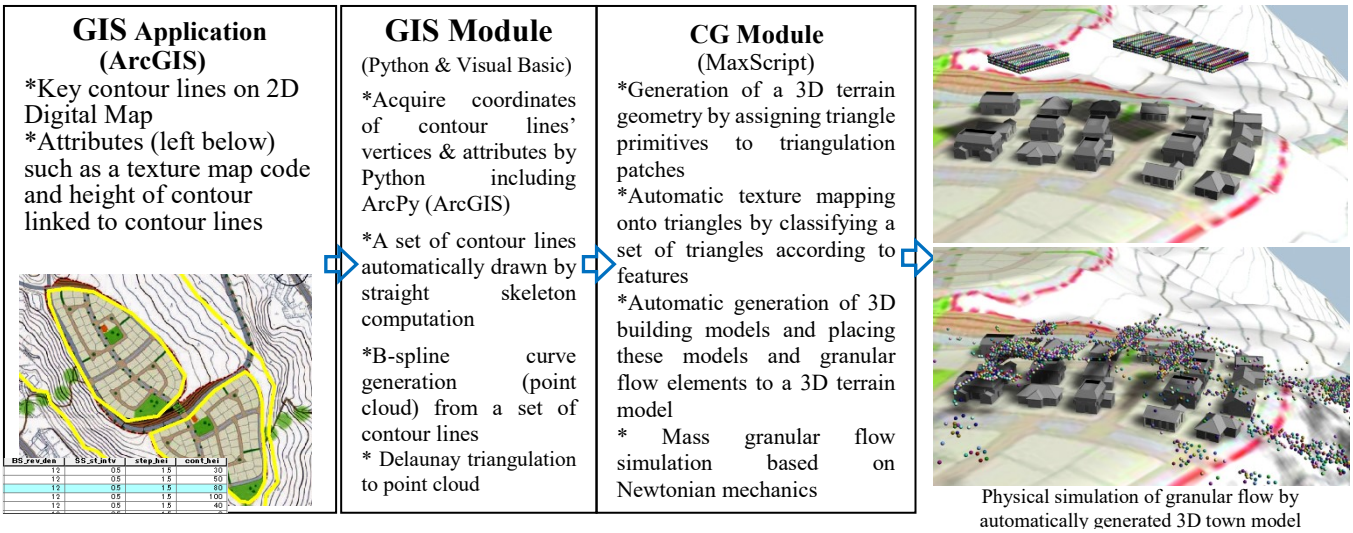


Fig. 1. Pipeline of mass granular flow simulation with realistic 3D models automatically generated

The research on 3D reconstruction is concentrated on reconstructing the rough shape of the buildings, neglecting details on the façades such as windows, and so on [6]. However, there are some application areas such as urban planning and landscape evaluation where the immediate creation and modification of building models with details are requested to present the alternative 3D town model.

Moreover, in case of creating non-existing 3D town models, we cannot use these computer vision technologies, since town models do not exist now, but will be built in future, e.g., according to the disaster prevention planning map.

Procedural modeling is an effective technique to create 3D models from sets of rules such as L-systems, fractals, and generative modeling language [7]. Müller et al. [8] 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 [5].

In our research, using the straight skeleton computation, a 3D terrain model is also automatically generated, which will be used for a 3D town site prepared and developed among hilly area shown in Fig. 1 right. Since the planning 3D terrain model will be developed in future according to the disaster prevention planning map, and does not exist now, it cannot be created by using the computer vision related technologies.

The CAD and GIS software, e.g., AutoCAD Civil 3D [9] and ArcGIS [10] can also create 3D terrain models, and map contours. These software assume that there exists a 2D map in which every points on the map are associated with elevation data, and depending on the elevation of the points, these software can create 3D terrain models and draw contours of equal elevation. Usually, every points in the 2D map are displayed in different colors or brightness, depending on the elevation of the point. Hence, if a planning map has no elevation data in it, these software cannot draw the contour lines. In our proposal, based on manually drawn key contour lines given an elevation, the faces connecting key contours are automatically formed, and thus 3D terrain models are created. The contour lines of equal elevation can be automatically drawn, based on the connecting faces that form a 3D terrain model.

Also, contour lines can be drawn by interpolation between a pair of corresponding main contours, which are assumed be topologically the same contours. A large number of contours will be interpolated between many pairs of points on the corresponding main contour. However, contour lines cannot be drawn by this interpolation between topologically different contours. In our proposal, the faces connecting main (key) contours are automatically created between topologically changing geometries, such that one polygon has two polygons inside, and one polygon is converted into two polygons when one polygon keeps receding.

In our research the straight skeleton computation is used for creating the faces connecting main (key) contours which are given an elevation. Based on the connecting faces, 3D terrain models are generated, and then 3D house models are placed on these prepared sites. Our methodology saves the laborious steps of drawing contour lines.

III. Pipeline of Automatic Generation

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

The knowledge-based system is proposed for generating 3D terrain model by linking the key contour lines to information from domain specific knowledge in GIS maps; the attributes

data such as the elevation, features and control point density of B-spline curve.

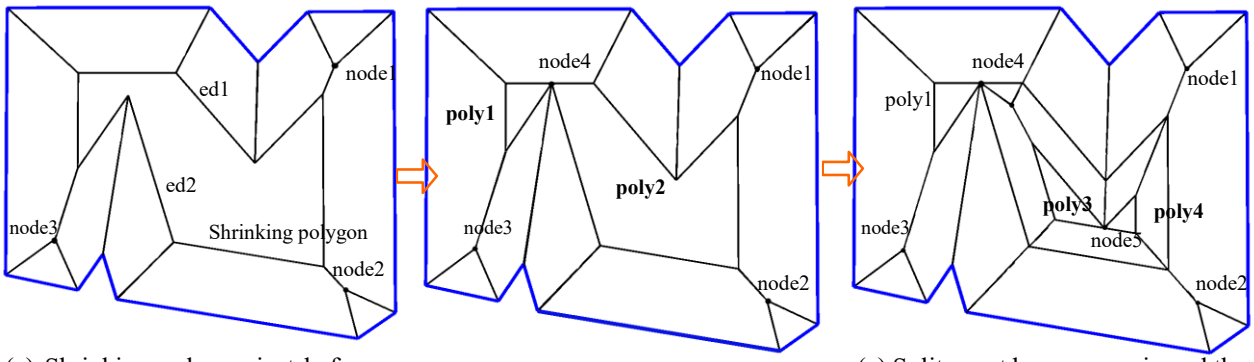
As shown in Fig. 1 & 2, preprocessing at the GIS module includes the procedures as follows: **(1)** Find out the direct parent contour and equivalent contour between which "me" contour to create connecting faces. **(2)** Calculate the shortest receding distance at which an Edge event will happen. **(3)** Check if a Split event happens until an Edge event will happen while shrinking the polygon shape. If a Split event happens, then the polygon is divided into two polygons, which will be active polygons to be checked (Fig. 2(a), 2(b)). **(4)** If a Split event does not happen during shrinking, then an Edge event will happen and the polygon's topology will be changed to be an active polygon to be checked. **(5)** The shrinking process continues until all active polygons are triangles which will converge to a point (Fig. 2(c)). **(6)** Classify the converge points (nodes) according to original edges to which the points belong. **(7)** An original polygon will be divided into several monotone polygons (faces). **(8)** Based on these faces, points of contour lines of equal elevation are generated. **(9)** From contour line points, B-spline curve are formed for smooth map contours depending on the control point density (Fig. 2(d), 2(e)). **(10)** The delaunay triangulation of B-spline curve points (Fig. 2(e)) **(11)** Convert delaunay triangulation data to fit for the export format of MAXScript

As shown in Fig. 1, the CG module receives the pre-processed data that the GIS module exports, generating 3D terrain models. The CG module follows these steps: **(1)** Generation of a 3D terrain geometry by assigning triangle primitives to triangulation patches (Fig. 2(f)). **(2)** Automatic texture mapping onto triangles by classifying a set of triangles according to features such as top flat area, slope, flat site, slackly inclined site, and steep incline site (Fig. 2(f)). **(3)** Automatic generation of 3D building models and placing these models and granular flow elements to a 3D terrain model for mass granular flow simulation based on Newtonian mechanics (Fig. 1 right).

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

IV. STRAIGHT SKELETON COMPUTATION

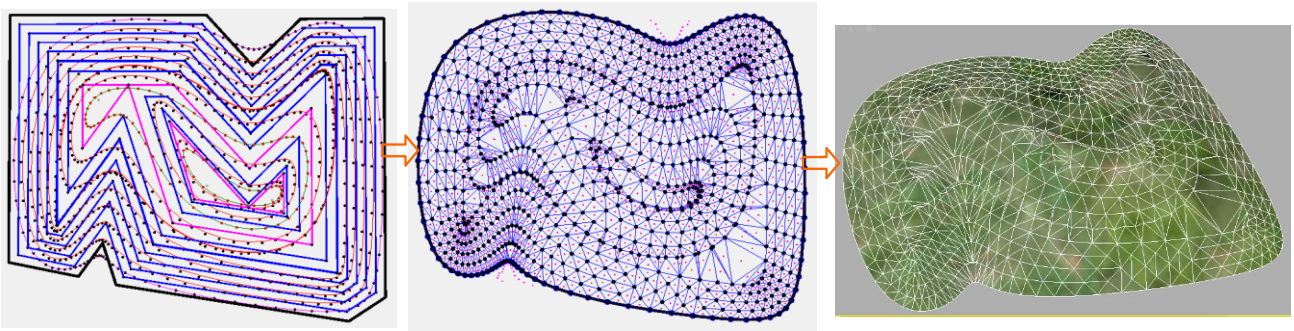
How the faces connecting key contours are automatically formed from key contours is as follows. In most cases, the map contours are receded when the altitude increases. If a receded map contour line is crossed by itself at a certain height, then a contour line or a contour polygon is divided into two or more contour polygons. In this way, a contour polygon will change topologically during the receding process. If a hill with two peaks is surrounded by one contour polygon, then one polygon



(a) Shrinking polygon just before a split event. Edge events give rise to node1～node3

(b) Split event happens and the polygon is split into two polygons; poly1 and poly2.

(c) Split event happens again and the polygon is split into two triangles; poly3 and poly4.



(d) Polygons shrinking at the constant speed; Polygons (pink) at Split event. B-Spline curves (pale pink) from these polygons

(e) B-Spline curves from these polygons. Delaunay triangulation to points on B-spline curves

(f) Generation of a 3D terrain geometry by assigning triangle primitives to triangulation patches. Automatic texture mapping onto triangles

Fig. 2. Automatic generation process of a 3D terrain model by straight skeleton computation and so on

will be divided into two polygons as the altitudes increases. In these cases, the polygon has to perform crossing detection with itself since the polygon reduces its size gradually. In this case, a straight skeleton or medial axis computation is quite useful for crossing detection and shape reduction while the polygon is shrinking.

Aichholzer et al. [5] 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(a) - 2(d), 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. [5], 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.*
where 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:

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

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, and nodes by an edge event and a split event. Fig. 2(d) shows a set of polygons shrinking at the constant interval.

Fig. 2(e) shows B-Spline curves from these polygons, and Delaunay triangulation to points on B-spline curves. Fig. 2(f) shows the 3d terrain model automatically generated.

A. 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

```
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) { " */
```

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

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.

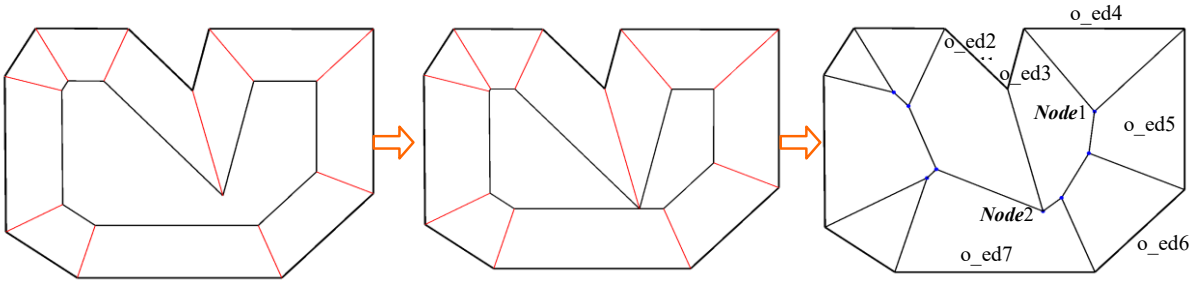
The generated nodes will be associated with the edges of original **P** (original edge: o-edge), since at least three original edges sweep to form the node. Therefore, at each event when the node is generated, at least three o-edges will be linked to the node. When a square or a regular hexagon collapses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

B. HOW MONOTONE POLYGONS are FORMED

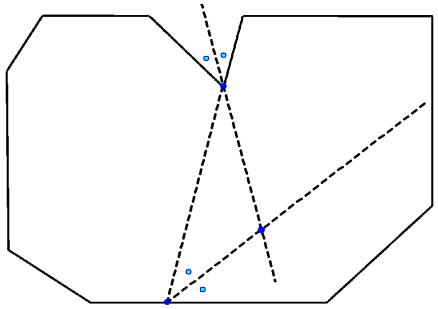
Fig. 4(c) shows how these interior monotone polygons are formed. When a shrinking process starts, the edges of the polygon sweep inwards from their original edge (o_ed_i). Nodes arise from the edge event or the split event. For example, *Node*1, arisen by the edge event, is the convergent point into 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



(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.

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

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.

V. APPLICATION AND CONCLUSION

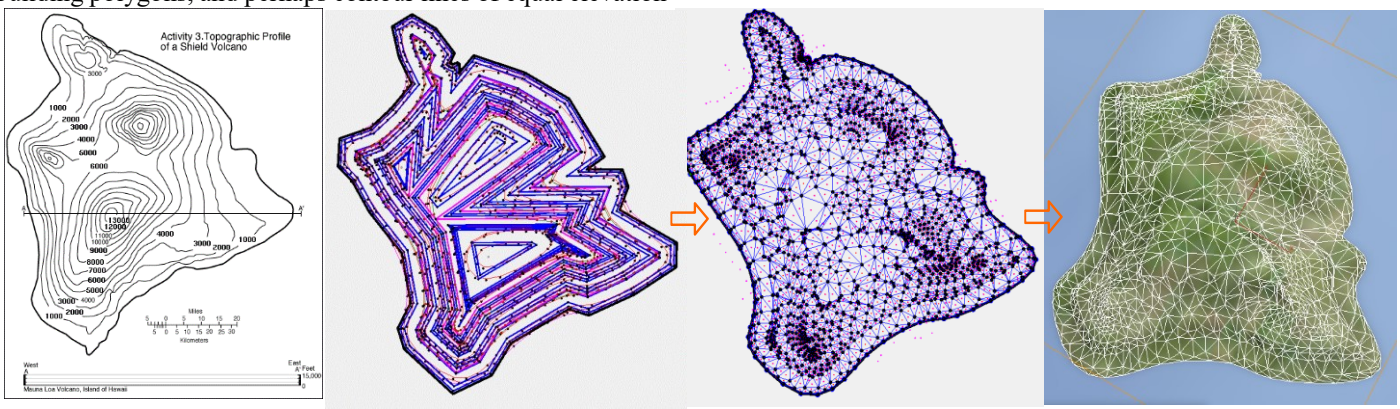
Here is the example of a 3D terrain model automatically generated by the GIS and 3DCG integrated system. Here is the example of a 3D terrain model automatically generated by the GIS and 3DCG integrated system. Fig. 5 shows automatic generation process of a 3D terrain model by straight skeleton computation, B-spline curve, and Delaunay triangulation. The target of automatic generation of a 3D terrain model is the island. The outline polygon of the island is manually drawn in reference to the topographic map of Mauna Loa with 300m elevational contour lines (*andreamaryy.blogspot.com*).

The outline polygon is shrinking at the constant speed with the polygons at Split event shown in pink. From a series of shrinking polygons, B-spline curves shown in pale pink are formed for smooth map contours. Finally at GIS module, Delaunay triangulation to points on B-spline curves is implemented. At CG module, a 3D terrain geometry is generated by assigning triangle primitives to triangulation patches, and finally automatic texture mapping onto triangles is implemented.

For everyone, a 3D town model is quite effective in understanding what if this alternative town plan is realized, what image of a sustainable town will be. Traditionally, urban planners design the town layout for the future by drawing building polygons, and perhaps contour lines of equal elevation for a 3d terrain model on a digital map. Depending on the building polygons and contour lines, the proposed system automatically generates a 3D town model placed on a 3d terrain model so instantly that it meets the urgent demand to realize another alternative urban planning for sustainable development or disaster prevention.

If given digital maps with attributes being inputted, as shown in 'Introduction' section, the system automatically generates two hundreds 3D building models within less than 30 minutes by the personal computer with 7th Gen Intel® Core™ i7 processors.

In our proposal, based on key contour lines manually drawn, the faces connecting key contours are formed, and then 3D terrain models are automatically created. The multiple contour lines of equal elevation can be automatically drawn, based on the connecting faces that form a 3D terrain model. In our research the straight skeleton computation is used for forming faces connecting key contours. Based on the automatically drawn map contour lines, 3D terrain models are generated, and then 3D house models are placed on these prepared sites. Our methodology saves the laborious steps of creating 3D terrain models. The uniqueness of our system is that the proposed system performs a physical simulation using a more realistic environment shown in Fig. 6, such as an automatically generated 3D town model placed on a 3D terrain model, and mass granular flow based on Newtonian mechanics. This is realized by using MassFX of 3ds MAX which provides tools for animating objects to behave as they do in the physical world. In this environment, a 3D terrain model is set to static rigid bodies, and granular is dynamic rigid bodies. In the proposed 3D model-based simulation, any distribution or shape of mass granular formation is possible, depending on 3D terrain models and what kind of simulation will be implemented such as slope failure, avalanche or debris flow.



(a) Target of Automatic Generation of 3D Terrain Model: Topographic map of Mauna Loa with 300-m elevational contour lines (andreamaryy.blogspot.com)

(b) Polygons shrinking at the constant speed; Polygons (pink) at Split event. B-Spline curves (pale pink) from these polygons

(c) From shrinking polygons, B-spline curve are formed for smooth map contours. Delaunay triangulation to points on B-spline curves

(d) Generation of a 3D terrain geometry by assigning triangle primitives to triangulation patches. Automatic texture mapping onto triangles

Fig. 5. Process of Automatic Generation of a 3D Terrain Model and Example of a 3D Terrain Model

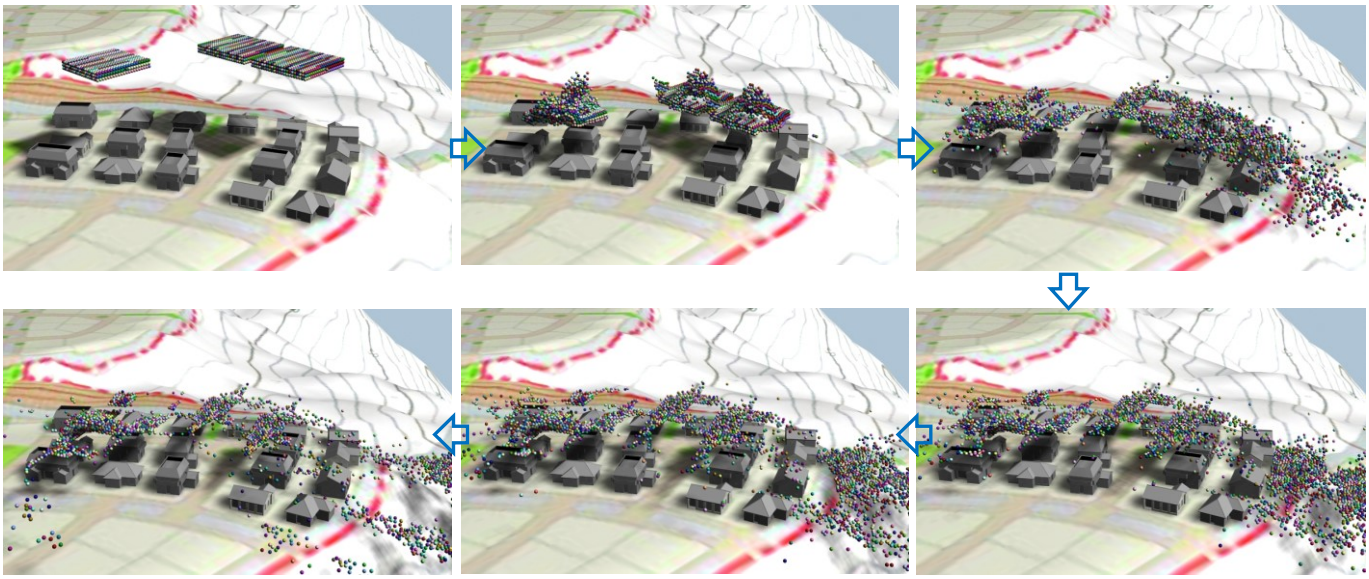


Fig. 6. Mass granular flow simulation under the environment similar to reality, such as an automatically generated 3D town model placed on a 3D terrain model

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI; Grant-in-Aid for Scientific Research (C) Grant Numbers JP15K06260, 15K06354, 16K01045.

REFERENCES

[1] Zenit, R., 'Computer simulations of the collapse of a granular column', Physics of Fluids, 17 (3) (2005)

[2] Thompson, E.L. and Huppert, H.E., 'Granular column collapses: further experimental results', Journal of Fluid Mechanics, 575, 177-186 (2007).

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

[4] 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)

[5] 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).

[6] 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)

[7] 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)

[8] 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)

[9] Autodesk Knowledge Network: Autodesk AutoCAD Civil 3D: https://knowledge.autodesk.com/support/autocad-civil-3d/learn-explore/

[10] ArcGIS websites: ArcGIS resources: https://pro.arcgis.com/ja/pro-app/tool-reference/3d-analyst/how-contouring-works.htm