Irregular Vertex Editing and Pattern Design on Mesh

Yoshihiro Kobayashi, Arizona State University

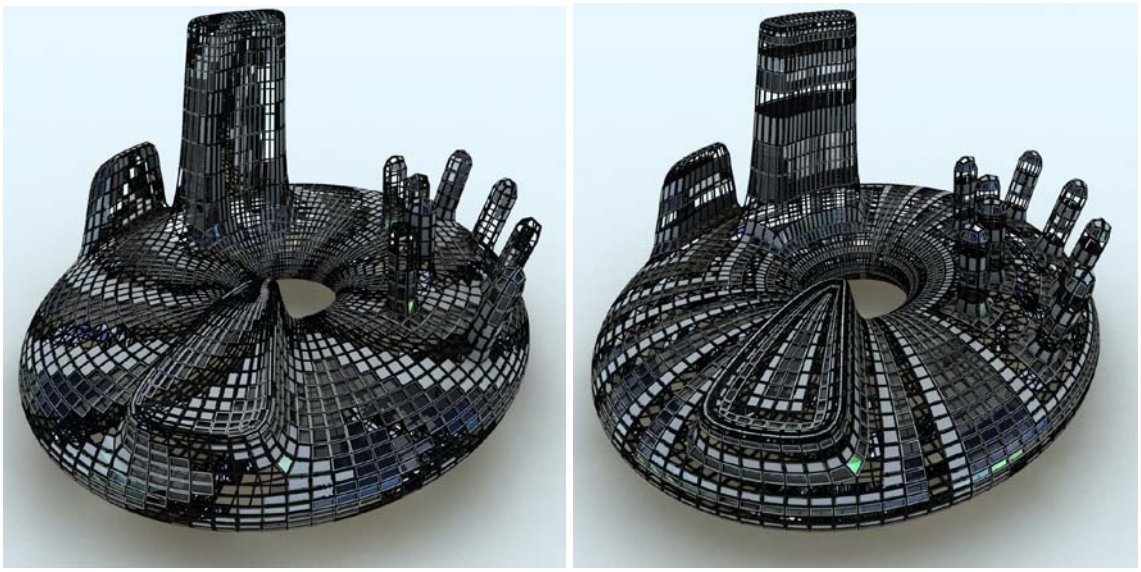

Figure 1. Vertex Growing (left) and Edge Growing Pattern (right)

Abstract

This paper introduces an innovative computational design tool used to edit architectural geometry by addressing the problem of irregular vertices. An irregular vertex is a special kind of vertex which is connected with fewer or greater less or more edges than regular vertices on a mesh object. Irregular vertices create problems with further surface rationalization, as well as structural analysis and constructability of the surface. Geometry created using other tools can also be remeshed upon import. Using the developed tool, the user is able to identify irregular vertices, interactively change the type, and then move or remove these irregular vertices. Additionally, a computational tool to make various design patterns on the mesh after the topology has been edited is also developed. The workflow is illustrated step by step in the pipeline. The advantages and disadvantages of editing mesh topology on architectural geometry design including the limitations are discussed at the end.

1 Introduction

Currently, very complicated free surfaces have been applied to architectural design by using high-end computational design tools such as building information modeling (BIM) and parametric modeling. Most of the designs are made by using a bottom-up approach in which the form is defined from a list of small components. For example, architectural geometries are generated in three steps in parametric modeling: 1) specifying a location of control points, 2) generating a surface from section curves mathematically defined by using the control points, and 3) subdividing the surface as a set of units and assigning the element to each unit. This approach has the advantage that the designer can more strictly control the properties and attributes of the created geometry. However, there are several problems in this approach and it only works well in cases when the surface can be subdivided into a set of units by using UV mapping. Otherwise, some un-subdivided parts are generated. Designers and structural engineers have to find a solution to control these unexpected parts without degrading the design. The images of figure 2 show two typical solutions. One solution is to use other shapes to cover the un-subdivided part. Two small triangles and one pentagon are used to cover the corner even though quad faces are used throughout the rest of the structure as shown in the left image of Figure 2. The other solution is to cut the units at the edge of the surface as shown in the right image. Unfortunately there is no general solution for every kind of surface.

Figure 2. Sample Problems on Architectural Geometry

On the other hand, the top-down approach is to design the components from a given form. In an architectural firm, a master designer sketches a form, and a team then proposes several design solutions for building that form. This kind of procedure, however, does not usually function well in a computational design workflow. There are several problems with implementing the tool to support the top-down design approach. First, it is difficult to subdivide a surface into a set of developable units. In fact, it is still a challenging topic in computer science. Second, it is difficult to control inevitable

irregular vertices on a mesh surface. An irregular vertex is a special kind of vertex which is connected to less or more edges than regular vertices on a mesh surface. Such vertices create problems with further surface rationalization, as well as structural analysis and constructability of the surface. Finally, a pipeline of workflow has not been established.

Therefore, in order to realize the top-down approach in computation design, it is important to find the solution to control irregular vertices on a surface, implement the computational tool, and develop a pipeline of workflow.

2 Related Work

Our research group has developed several contributions on computational design in architecture and computer science. The first contribution was to develop a Field-Guided Shape Grammar framework and implement a computational tool to allocate objects on any kind of free-surface using Shape Grammar and tensor/vector fields as the guide (Li et al. 2011). The second contribution was to discover interesting behaviors of irregular vertices on mesh surfaces, and define a mathematical theory on irregular vertex editing (Li et al. 2010). A computational tool was also implemented and used in a practical architectural BIM competition (Kobayashi and Wonka 2011). The tool was limited for triangle meshes, but the extension to quad mesh has been developed. The most important contribution is that only our developed tool can make the irregular vertices travel on a surface without degrading the mesh quality or generating additional irregular vertices.

Conversely, there are so many off-the-shelf and commercially available computational design tools that support a bottom-up approach. For example, Generative Component is a powerful parametric modeling tool showing each design procedure as a node (Bently 2011). Rhino 3D and Grasshopper, in which the designer can control all components and association rules through the graphical interfaces, are very popular among students in schools of architecture in the US (Grasshopper 2011). Other scripting tools within 3D modeling packages such as MEL in Maya and MaxScript in 3DS Max are also used to generate the algorithmic geometries (Autodesk 2011).

There are several different research projects to subdivide the surface of architectural geometry. Eigensatz and his research group introduce the method to subdivide it into a set of strips (Eigensatz et al. 2010), and into a set of planer faces (Pottmann et al. 2010). Global remeshing research projects (Hoppe et al. 1993; Bommes et al. 2009) are also related our works.

3 Overview

The objective of this paper is to establish and demonstrate a pipeline to generate architectural geometry using a top-down approach. As explained in the Introduction section, it is required to implement a computational tool to edit irregular vertices in order to realize the approach. While the editing tool has already been developed and explained in our previous work (Li et al. 2010), this paper focuses on demonstrating the workflow

for architectural geometry design. Once the mesh topology is defined, the final step is to generate more practical solutions by generating detailed geometry. The generation tools and example designs generated using the tool are shown.

4 Methodology

This section shows two design solutions realized by using our developed tools.

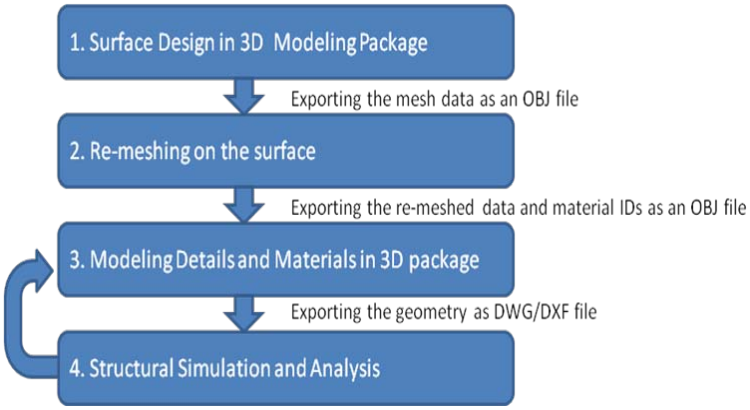

Figure 3. Our Developed Pipeline

One of the most important discoveries in our prior research projects was that a pair of irregular vertices can travel on a mesh surface without degrading the quality of the mesh or destroying the other mesh topology, though it was impossible for a single vertex. The pair traveling is implemented as a combination of basic operations such as flipping and collapsing an edge, and merging and splitting vertices. On behalf of this traveling behavior, we could find the way to cancel /remove the irregular vertex by making a trio of irregular vertices. A trio of irregular vertices can be converted to a single irregular vertex and two regular ones. In other words, it is possible to remove the irregular vertices on a mesh by traveling a pair to an isolated single vertex. We categorized the pair behavior into type-move, move, and remove & generate groups as the result. The detail is explained in (Li et al 2010). Using our irregular editing tool, a top-down approach becomes possible. The diagram in the Figure 3 shows the pipeline.

4.1 Irregular Vertex Editing

It is difficult to edit faces on free surface objects without degrading the quality of mesh in current 3D modeling packages. In most cases, the size and shape of faces are different. The edges are not smoothly lined up. Quad and triangle faces are mixed. There are irregular vertices at unexpected locations on the mesh (Figure 5). Therefore, it is almost impossible to generalize the pipeline to fix all problems described above by editing the original mesh. Therefore, we propose an approach to remesh the original object at the first step.

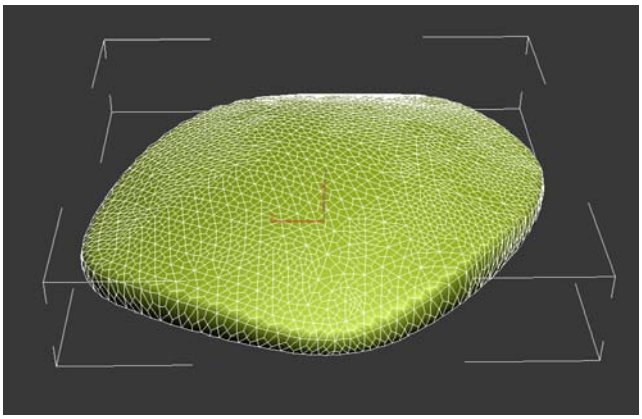

Figure 4. Original Mesh Object

Once the original mesh created in another 3D modeling package is imported into our developed tool, the user can choose either full-automatic remeshing or remeshing step-by-step by growing the faces from an initial unit. Figure 5 shows the result of automatic remeshing of the object in Figure 4. Several irregular vertices are generated at unexpected locations on the mesh as shown in red circles. An irregular vertex with valence 5 (V5), which has five edges instead of 6, is represented as a blue dot. A irregular vertex with valence 7 (V7), which has seven edges, is represented as a orange dot in Figure 5.

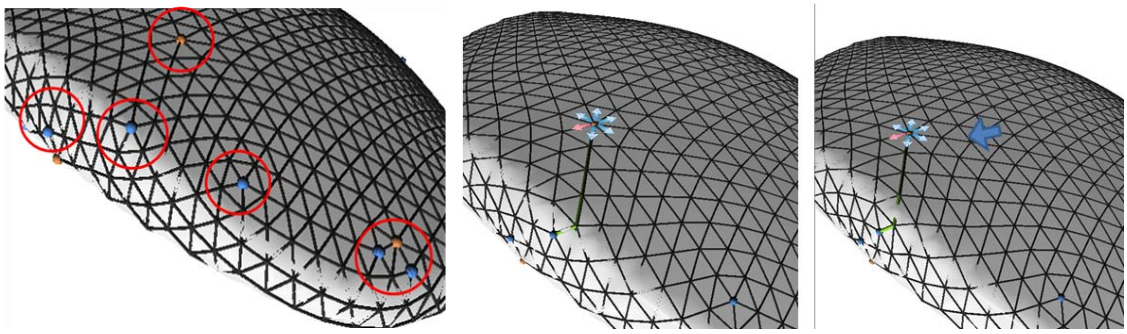

Figure 5. Result of Remeshing and V5-V7 Pair Traveling. The remesh result (left), before the traveling (middle) and after traveling (right)

The next step is to remove the irregular vertices. It would be ideal to remove all of irregular vertices, but it is unfortunately impossible based on the topological invariant known as Euler characteristics (eq.1).

$\chi = V - E + F,$ (eq. 1)

where V is the number of vertices, E the number of edges, and F the number of faces. The Euler characteristic χ was classically defined for the surfaces of polyhedral, and any convex polyhedron's surface has the Euler characteristic as 2. In short, any manifold triangle mesh has at least twelve irregular vertices as a soccer ball has twelve pentagons.

Our solution is to find a pair of V5 and V7 and move the pair to another V5 or V7. As shown in Figure 5, a pair of V5-V7 can travel to any location on the mesh without changing the valence of other vertices.

Some irregular vertices are canceled by converting V5-V7-V5 to V5 with two regular vertices. The others are moved away to the bottom side of the object.

Once the mesh is edited, the data is sent to a 3D modeling package again such as 3ds Max. We developed a plug-in tool to generate patterns and assign a material ID for each face on a mesh. The following are output samples using the tool to generate a pattern.

4.2 Shape Grammar on Mesh

Making design patterns on meshes is different from making them on the infinite 2D plane and 3D world space. A mesh is a discrete field. A regular shape grammar stops on the mesh when there is no space to grow. This is a well-known problem called singularity points in tensor field design, and it is still a challenging problem to control the points in computer science on graphics and geometry. Our proposal is to develop shape grammar rules that generate patterns without degrading the design.

So far we implemented two rules: edge growing and vertex growing. Edge growing is a rule to assign different colors to faces sharing the edges of current selected faces. This can generate a check pattern by selecting a single face as an initial face. The top four images in the Figure 7 show the check patterns in quad and triangle mesh.

On the other hand, vertex growing is a rule to assign different colors to faces sharing the vertices in the current selected faces. This can generate a ring pattern by selecting a single face as an initial selected face. Then below four images in Figure 7 show the ring patterns in a triangle and quad mesh sphere. The pseudo codes are described in Table 1.

Figure 6 shows the graphical notation of edge growing and vertex growing rules on quad and triangle meshes.

Table 1 Pseudo Codes of Vertex and Edge Growing

Vertex-Grow Pattern

selectedFaceList
for each step
  vList = get the list of vertices used in the selectedFaceList
  fList = get the list of faces using the vList without the faces that have materials already
  assign a new material to fList
  selectedFaceList = fList

Edge-Grow Pattern

selectedFaceList
for each step
  eList = get the list of edges used in the selectedFaceList
  fList = get the list of faces using the eList without the faces that have materials already
  assign a new material to fList
  selectedFaceList = fList

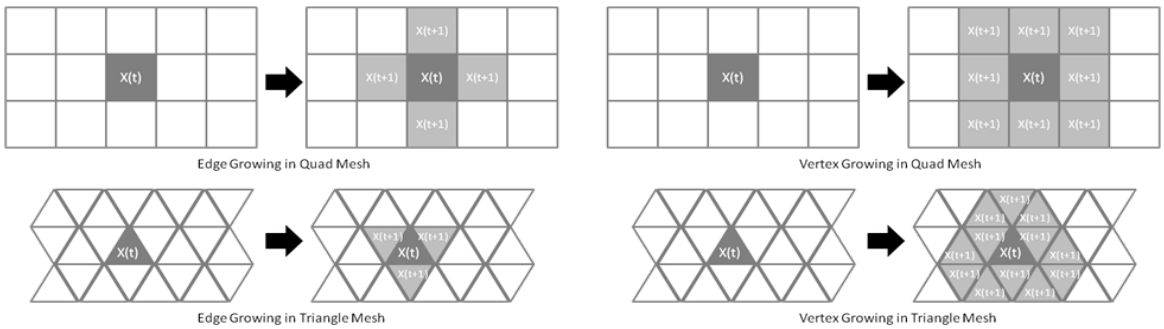

Figure 6. Edge and Vertex Growing Rules

Figure 7. Examples of Edge and Vertex Growing on Sphere

Figure 8. Samples of Generated Design Pattern: Original Mesh with Initial Selected Face(s) (left), the results of growing with colors (middle) and black and white (right)

The first tool to edit the irregular vertices is developed in C++ and OpenGL. The other tool to generate the pattern and geometry on the mesh is developed in MaxScript as a plug-in for 3ds Max. Both tools are operated in regular Windows-based PCs.

5 Case Study

The following are some examples in which several irregular vertices are generated on purpose using our tool.

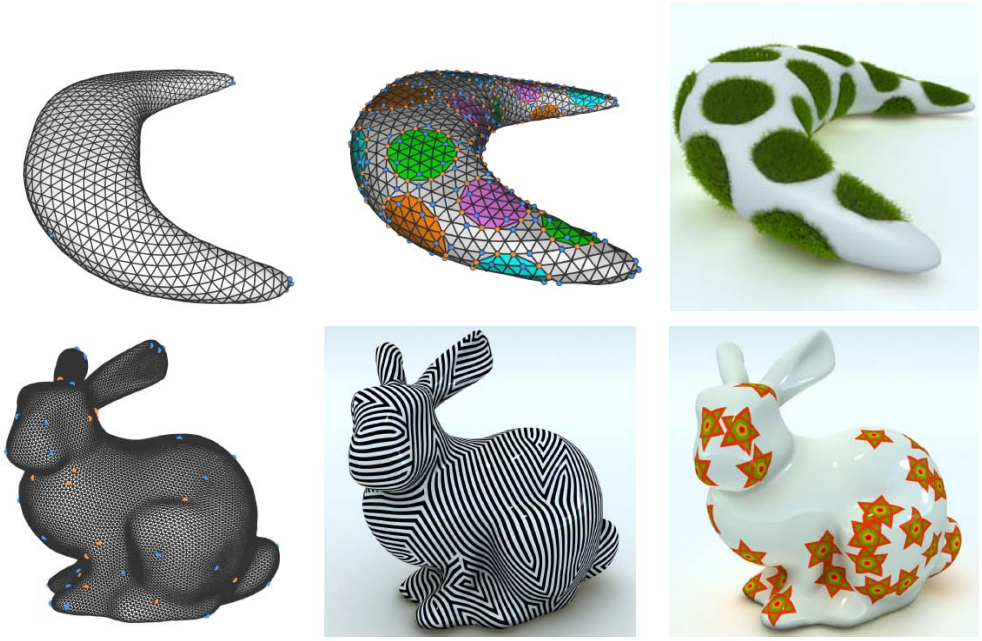

Figure 9. Other Example Designs

The top three images in Figure 9 show the original mesh (left), generated irregular vertices on purpose (middle), and output by assigning materials (right). The bottom three images are the image of a bunny model after editing irregular vertices (left), output with stripe pattern (middle), and output with star pattern (right).

This tool and pipeline was used in a practical architectural design competition, Build Live Tokyo 2010, organized by the International Alliance for Interoperability Japan Associate (IAI) Japan (IAI Japan 2010). It is a competition to design a media center within 48 hours. We participated in the competition as a team with Forum8 Co. Ltd, which is a civil-engineering software developing company (Forum8 2011). The images in Figure 10 are the output design. The top roof was created by using our developed tool, and simulated by practical architectural structural tools (Kobayashi and Wonka 2011)

Figure 10. Architectural Geometry Design in a Competition

5.1 Discussion

The advantages and disadvantages of our developed pipeline are detailed in this section. The advantages are:

- Reducing the time to generate architectural geometry designs from a free surface object.
- Enabling a top-down digital design process
- Generating alternative designs with a few input changes.

Once a mesh object is defined, it is possible to remesh in a few seconds using a standard PC. After the remeshing process, the irregular vertices need to be edited by removing as many as possible without degrading the geometry and moving the pairs to another area on the given mesh. Though it is up to the experience and skill set, we could get an expected result in 10 minutes. The next step is to select initial face/faces and to choose the growing rule from edge-growing or vertex-growing. It is possible to generate a pattern in a few seconds. 10 minutes is enough time to test 5 to 10 different patterns. The next step to generate detailed geometry requires several trial and error iterations in order to get a satisfied design. We spent 5 to 10 minutes in order to get the proper size of frames and glass panels using the graphical interfaces. If a preset is used in this process, it is possible to complete it within a minute. The final step is to put the geometry in a rendering environment, and render an image. We used a rendering tool, V-ray 1.5, in 3ds Max, and it took about 10-15 minutes to render the 1000 by 1000 pixels image of Figure 1.

In short, it is possible to generate an architectural geometry similar to Figure 1 in roughly an hour. In fact, once the growing rules and the parameters for detailed geometry are defined, it was possible to get another design within 15 minutes for any kind of mesh object.

The disadvantages are:

- Difficulty on moving the irregular vertices
- Difficulty on editing the irregular vertices around sharp edges
- Difficulty on defining the size of face

It is mathematically impossible to cancel/remove all irregular vertices on mesh. Therefore, it is very important to define where the vertices should be on mesh. However, the irregular vertices can travel only as a pair, so it is not easy to control the locations intuitively. It is more difficult when the object has sharp edges. We have developed some solutions by specifying the movable area, but it does not work well in some cases. In addition to the location of irregular vertices, it is also difficult to find the proper size of faces while keeping the mesh quality intact. If the size of the face is too big for a given mesh, the features of the geometry are lost. If the size is too small, it takes more time to develop and manage the elements. Therefore, in order to find the proper size, trial and error is essential.

6 Conclusion and Future Work

- We introduced our implemented design tool to re-mesh a free surface object and edit the irregular vertices without degrading the mesh quality.
- We demonstrated our developed top-down approach in computational architectural design and the pipeline to design architectural geometries using the developed tool.
- The workflow to re-mesh the surface, edit the irregular vertices, assign the design pattern on the mesh, and generate the details was explained.
- Several sample outputs designed by using our pipeline were illustrated in the case study.

In terms of the irregular vertex editing, our solution did not work well when the mesh has sharp edges. Our future work is to improve the tool on this issue using the curvature and distance field. In the pattern generation, we would like to develop more rules and generalize the pattern generation on a mesh surface.

Acknowledgement

We would like to acknowledge the help of Eugene Zhang (Oregon State University), Peter Wonka and Yuanyuan Li (Arizona State University) for the tool implementation, Christopher Grasso and Michael McDearmon for rendering figures, and the support of Forum8 Co Ltd on using VR and structural analysis tools. This project was funded by NSF contracts IIS-0915990, CCF-0643822, CCF-083080, and IIS-0917308.

References

AUTODESK 2011. Online available on 4.1.2011 at http://www.autodesk.com

BENTLEY 2011. Online available on 4.1.2011 at http://www.bentley.com

Bommes, D., Zimmer, H., and Kobbelt,L., 2009. Mixed-integer quadrangulation. ACM Transaction on Graphics (SIGGRAPH 2009)

Eigensatz, M., Kilian, M., Schiftner, A., Mitra, N. J., Pottmann, H., and Pauly, M., 2010. Paneling Architectural Free Surface, ACM Transaction on Graphics (SIGGRAPH 2010)

Forum8 2011. Online available on 4.1.2011 at http://www.forum8.co.jp

GRASSHOPPER 2011. Online available on 4.1.2011 at http://www.grasshopper3d.com

Hoppe, H., Derose, T., Duchamp, T., McDonald, J., and Stuetzle, W. 1993. Mesh optimization. IN SIGGRAPH 1993, ACM, NEW YORK, NY, USA, 19-26

IAI Japan 2010. Online available on 4.1.2011 at http://www.building-smart.jp/

Kobayashi, Y. and Wonka, P. 2011. Irregular Vertex Editing for Architectural Geometry Design. Proceeding of Simulation of Architecture and Urban Design (SIMAUD 2011)

Li, Y., Bao, F., Zhang, E., Kobayashi, Y., and Wonka, P. 2011. Geometry Synthesis on Surfaces Using Field Guided Shape Grammar, IEEE Transactions on Visualization and Computer Graphics, Feb. 2011

Li. Y., ZHANG, E., Kobayashi, Y., and Wonka, P. 2010. Editing Operation for Irregular Vertices in Triangle Mesh. ACM Transaction on Graphics (SIGGRAPH ASIA 2010)

Pottmann, H., Huang, Q., Deng, B., Schiftner, A., Kilian, M., Guibas, L., and Wallner, J., 2010. Geodesic Patterns. ACM Transaction on Graphics, Vol. 29, Nr.3, 2010