April 9, 2000
RT0354
Computer Aided Design 12 pages

Research Report

Rough Quadrilateralation of Triangular Meshes via Belt-like Subregioning for Surface Reconstruction

Takayuki ITOH, Atsushi YAMADA, Tomotake FURUHATA, Keisuke INOUE, and Kenji SHIMADA

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

Rough Quadrilateralation of Triangular Meshes via Belt-like Subregioning for Surface Reconstruction

Takayuki Itoh* Atsushi Yamada* Tomotake Furuhata**
Keisuke Inoue* Kenji Shimada***

* IBM Research, Tokyo Research Laboratory, {itot, ayamada, inoue}@trl.ibm.co.jp
** IBM Japan, Yamato Software Laboratory, furuhata@jp.ibm.com
*** Mechanical Engineering, Carnegie Mellon University, shimada@cmu.edu

Abstract

Surface reconstruction is a technique for converting fine geometric models such as triangular meshes or unorganized points into a smaller number of curved surface patches. Here, the topology of surface patches should be taken into consideration, because the smoothness and continuity of the patches depend on their topology. This paper reports a method for forming the topology of surface patches for surface reconstruction methods, given an initial fine triangular mesh. It first clusters the triangular elements into several belt-like subregions along the mesh domain boundary, and then subdivides the subregions into rough elements. The rough elements can be used as surface patches by fitting them to the input triangular mesh. The topology of rough elements is suitable for surface reconstruction, because they are well-aligned along the domain boundary.

Keywords: surface reconstruction, advancing front, quadrilateral meshing.

1. Overview

Surface reconstruction of geometric models [Baj95] [Eck96] [Kri96] is an active research area in computer graphics (CG) and computer aided design (CAD). The technique is useful in the CG area, because it reduces the data size of geometric models. It is also useful in CAD area, since it converts polygonal meshes or unorganized points into surface models, which are generally used in commercial CAD systems.

Figure 1 shows an outline of the surface reconstruction process in our implementation [Yam99b]. Given fine triangular meshes or unorganized points, our implementation first forms the topology of a set of rough surface patches, and then fits the patches to the input mesh or points.

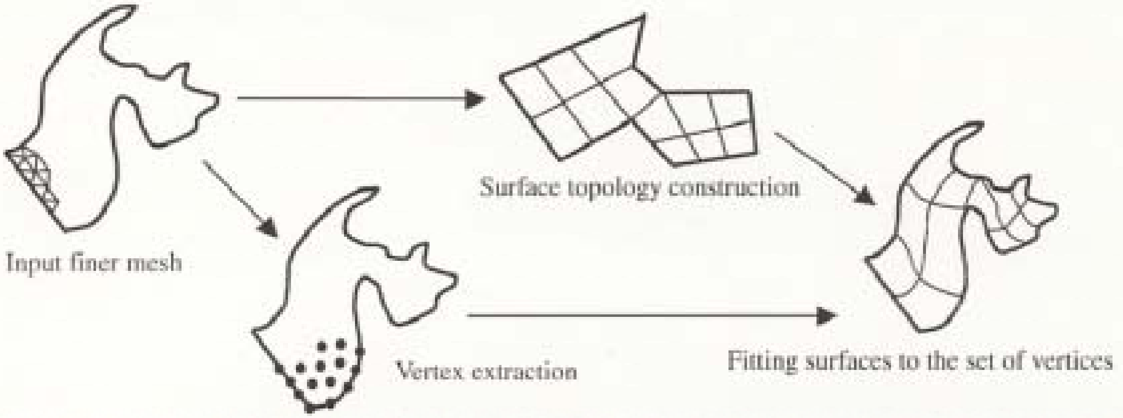


Figure 1. Outline of our surface reconstruction method.

Our previous report [Yam99b] focused on the fitting process of surface patches to unorganized points, but the construction of a topology of surface patches is also important, because the smoothness and continuity of the surface patches strongly depends on their topology. There have been various approaches for constructing the topology of surface patches, however, it is still a vital problem.

We think that the following conditions are desirable for a topology of surface patches:

Condition 1: The patches are quadrilateral.

Condition 2: The patches are significantly larger than the input triangular elements.

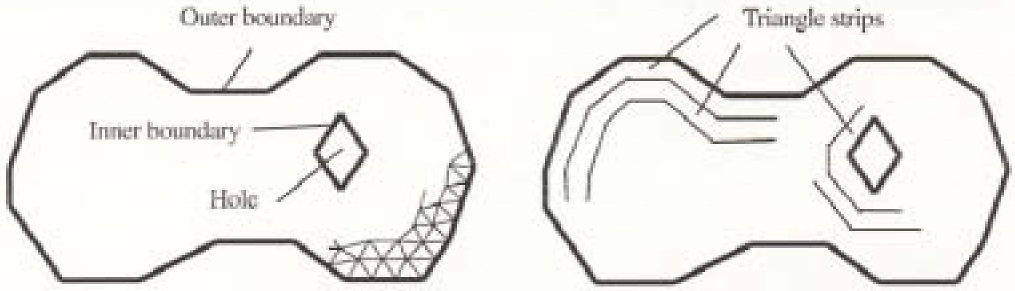
Condition 3: The angles of the vertices of the patches are close to 90 degrees. To satisfy it, it is better that vertices on the domain boundary are adjacent to two patches, if the boundary should be G1 continuous around the vertices. It is also better that vertices inside the geometry are adjacent to four patches. Experimentally we have found that the topology along the domain boundary is more important.

Conditions 1 and 3 are very similar to the requirements for quadrilateral meshes for numerical simulations such as the finite element method (FEM), so we think it is worthwhile to adapt quadrilateral meshing methods to develop surface patches topology construction methods.

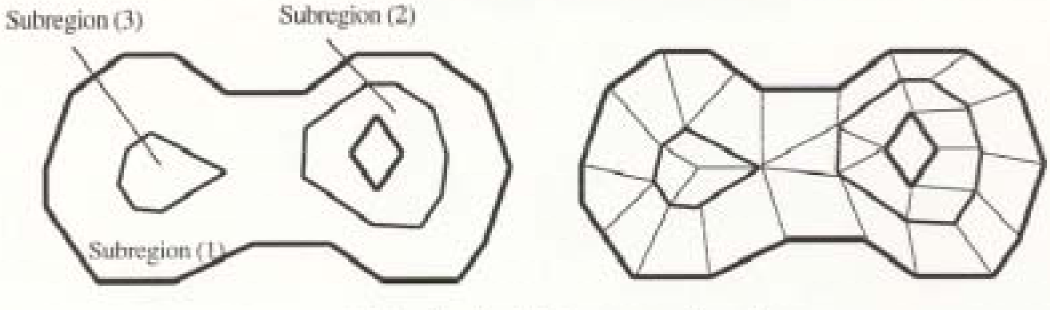
The advancing front method [Bla91] [Cas96] [Ras97] [Whi97] [Zhu91], a popular quadrilateral meshing method for numerical simulations, is one of the most worthwhile methods to follow. It first generates elements along the domain boundary, and then continues generating elements along the boundary of the empty region in order, until the elements fill the whole domain. Quadrilateral meshes generated by the method are usually well-aligned along the boundary, and therefore the resulting topology is usually preferable for numerical simulations.

This paper reports a method for generating "rough" meshes from "fine" triangular meshes. The rough meshes generated by the method are well-aligned along the domain boundary, similar to the meshes generated by advancing front method.

Figure 2 shows the outline of the method. Given a fine triangular mesh shown in Figure 2(a), the method first generates triangle strips as shown in Figure 2(b). It then clusters them into several belt-like subregions with user-specified thickness, as shown in Figure 2(c). It finally generates rough elements by subdividing the subregions, as shown in Figure 2(d). Our method does not guarantee that all patches and vertices satisfy the above-mentioned conditions, but in this paper our experimental tests show that most of the patches and vertices will satisfy them.



(a) L: input triangular mesh. (b) R: triangular strips.



(c) L: subregion. (b) R: output rough mesh.

Figure 2. Outline of our topology construction method.

2. Implementation

This paper assumes that the data structure of an input triangular mesh, $M_I = (V, T, \Delta T, B, S, R, L)$, and an output quadrilateral mesh , $M_q = (W, E, Q)$, consist of the following ordered lists:

(1) vertices of M_t , $V = (v_1, ..., v_N)$,

(2) triangular elements of M_t , $T = (t_1, ..., t_M)$,

(3) adjacent elements of T , $\Delta T = (\Delta t_1, ..., \Delta t_M)$, which give at most three adjacent triangles for each triangle,

(4) mesh domain boundary loops, $B = (b_1, ..., b_L)$, which give a set of continuous triangular element edges on the mesh domain boundary,

(5) triangle strips, $S = ((s_{1,1}, s_{2,1}, ...), (s_{1,2}, s_{2,2}, ...), ..., (s_{1,L}, s_{2,L}, ...))$, which give a set of adjacent triangles aligned along B ,

(6) subregions, $R = (r_1, ..., r_K)$, which give a set of adjacent triangle strips,

(7) outer and inner loops of R , $L = ((l_{1,O}, l_{1,I}), (l_{2,O}, l_{2,I}), ..., (l_{L,O}, l_{L,I}))$, which give a set of vertices of M_t on the outer and inner boundaries of subregions,

(8) vertices of M_q , $W = (w_1, ..., w_P)$,

(9) edges of M_q , $E = (e_1, ..., e_Q)$, and

(10) quadrilateral elements, $Q = (q_1, ..., q_R)$.

2.1 Triangle strip generation

This process first forms the mesh domain boundary loops B . The process first extracts edges of M_t , which are adjacent to only one triangular element. It then traverses the extracted edges through their adjacency. The traversal necessarily arrives at the starting edge, and the visited edges during the traversal forms a loop. The process registers the set of visited edges into a boundary loop b_i and forms B by similarly traversing all the extracted edges through the adjacency of edges.

The process then extracts triangular elements adjacent to B . It extracts those triangles where at least one vertex lies on the boundary loop b_i , and registers the set of extracted triangles into a triangle strip $s_{1,i}$. It repeats the process until it extracts all those triangles where at least one vertex lies on B , and registers them into triangle strips. The process then extracts unregistered those triangles where at least one vertex lies on the triangle strip $s_{1,i}$, and registers the triangles into a triangle strip $s_{2,i}$. Similarly it extracts unregistered those triangles where at least one vertex lies on a triangle strip $s_{j,i}$, and registeres the triangles into a triangle strip $s_{(j+1),i}$. The process is repeated until all triangles are registered into a triangle strip.

Finally, the process generates a *triangle strip graph*, as shown in Figure 3, by mapping triangle strips to graph-nodes, and their adjacency to graph-arcs.

2.2 Subregioning by the simplification of a triangle strip graph

This process forms subregions R by simplifying the triangle strip graph, as shown in Figure 4. It assumes that each subregion r_i has the following values:

(1) triangle strips that r_i contains,

(2) triangles that r_i contains, and

(3) adjacent subregions.

The process initially defines that one subregion is mapped to each graph-node of the triangle strip graph. It then merges adjacent subregions into larger subregions, until all subregions become thick enough to generate rough elements inside them. Many of subregions form belt-like shapes along outer or inner boundaries of the mesh domain. The merge process consists of the following two steps.

2.2.1 Merger of subregions which have only one adjacent subregion

This step first extracts subregions that have only one adjacent subregion, and sort them according to the number of triangle strips that the subregions contain. The process then checks the numbers of triangle strips of the extracted subregions and their adjacent subregions in the sorted order. If the total of the number of triangle strips of an extracted subregion and its adjacent subregion is not too large, the process merges them into a larger subregion.

2.2.2 Merger of other small subregions

This step first extracts subregions that contain small numbers of triangle strips or triangles, and sorts them according to the number of triangle strips or triangles that the subregions contain. The process then checks the numbers of triangle strips or triangles of the extracted subregions and their adjacent subregions in the sorted order. If the sum of the number of triangle strips or triangles of an extracted subregion and its adjacent subregion is not too large, the process merges them into a larger subregion.

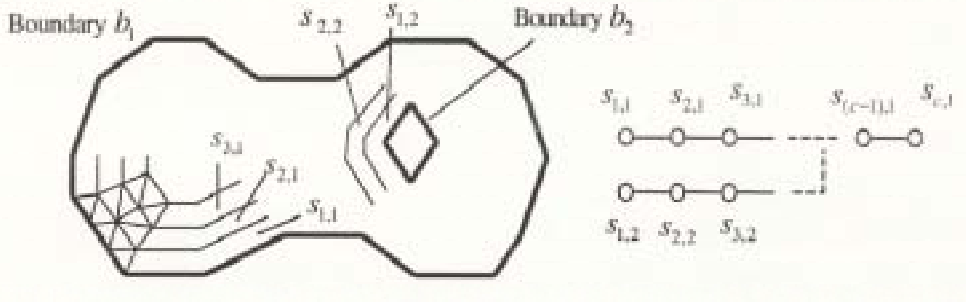


Figure 3. Triangle strip graph.

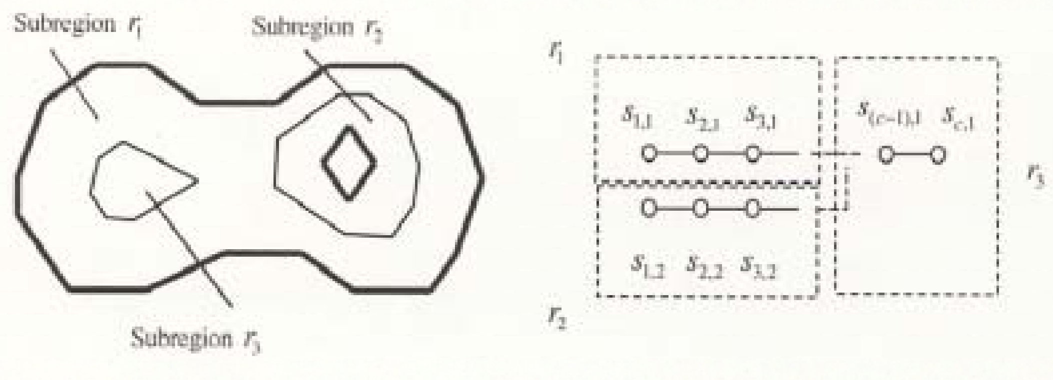


Figure 4. Simplification of the triangle strip graph.

2.3 Rough element generation by subdividing subregions

This process generates rough elements by subdividing subregions. In this process each subregion is regarded as a donut-like domain which topologically has one hole, as shown in Figure 5(a). If the subregion has no holes, or more than one hole, it is topologically modified before subdividing. The detail of the topology modification process is described in Section 2.4.

The rough element generation process consists of the following four steps:

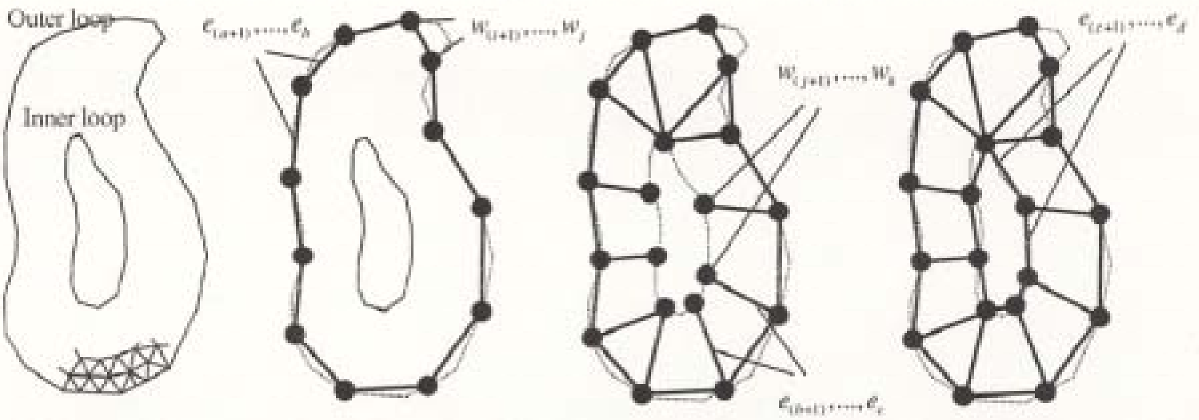
[Step 1] outer and inner loop generation,

[Step 2] edge generation on the outer loop,

[Step 3] edge generation connecting the outer and inner loops, and

[Step 4] edge generation on the inner loop.

Figure 5 shows the four steps the in rough element generation process. The detail of each step is described in the following sections.



(a) Step1: Outer and inner loops. (b) Step2: Edges on the outer loop.

(c) Step 3: Edges connecting outer and inner loops. (d) Step 4: Edges on the inner loop.

Figure 5. Rough element generation.

2.3.1 Outer and inner loop generation

This process first selects an unprocessed subregion r_s, and extracts triangle edges that are on the mesh boundary or adjacent to a different subregion. It then traverses the extracted edges through their adjacency. The traversal necessarily arrives at the starting edge, and the visited edges during the traversal forms a loop. If r_s contains a hole, two loops are to be generated. The larger loop is registered as an outer loop $l_{s,O}$, and the other is registered as an inner loop $l_{s,I}$.

2.3.2 Edge generation connecting the outer and inner loops

Next, the process selects some vertices of M_t on the outer boundary $l_{s,O}$, with user-specified interval. It then generates vertices of M_q, $w_{(i+1)},\dots,w_j (i < j)$, at the selected vertices of M_t. It then connects the vertices of M_q to form edges of M_q, $e_{(a+1)},\dots,e_b (a < b)$, as shown in Figure 5(b). If a part of $l_{s,O}$ is adjacent to another subregion, some edges of M_q are shared with the adjacent subregion.

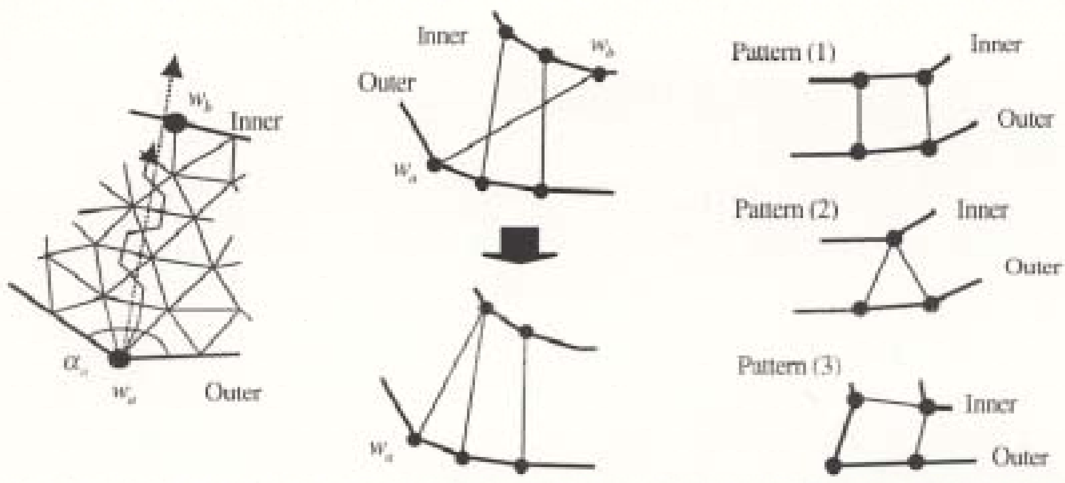


Figure 6. (a) Traversal of triangles. (b) Replace of a vertex on the inner loop. (c) Three patterns of rough elements.

2.3.3 Edge generation connecting outer and inner loops

Next, the process generates edges of M_q, $e_{(b+1)},\dots,e_c\ (b < c)$, which connect $l_{s,O}$ and $l_{s,I}$. It generates the edges starting from the vertices on $l_{s,O}$, $w_i, w_{(i+1)},\dots,w_j$, toward $l_{s,I}$. As shown in Figure 5(c), it generates vertices $w_{(j+1)},\dots,w_k\ (j < k)$ on $l_{s,I}$ while it generates the edges of M_q.

To generate an edge which connects $l_{s,O}$ and $l_{s,I}$, it traverses triangles in the subregion starting from a vertex w_a on $l_{s,O}$, along a vector that equally-divides the angle between two edges adjacent to w_a. When the traversal arrives at a triangle adjacent to $l_{s,I}$, a vertex of the triangle on $l_{s,I}$ is selected, and a vertex w_b is generated at the selected vertex. An edge of M_q, which connects w_a and w_b, is then generated. Figure 6(a) represents an example of the triangle traversal process. The dotted arrow line denotes the vector, and the polygonal arrow line denotes the traversal of triangles.

It is possible that the traversal does not arrive at $l_{s,I}$. In this case the process selects a vertex of M_t on $l_{s,I}$, which is the closest from w_a, and then generates an edge of M_q. It is also possible that the edge generated by above-mentioned process intersects previously generated edges. In this case, w_b on $l_{s,I}$ is replaced by a vertex of the intersected edge, as shown in Figure 6(b).

The above-mentioned process is skipped if the angle α_a between two edges on $l_{s,O}$ adjacent to w_a is not enough large.

Consequently, these four steps generate the following three patterns of rough elements, as shown in Figure 6(c):

[Element pattern 1] a quadrilateral element formed by two vertices on $l_{s,O}$ and two vertices on $l_{s,I}$.

[Element pattern 2] a triangular element formed by two vertices on $l_{s,O}$ and one vertex on $l_{s,I}$. It is generated if an edge which connects $l_{s,O}$ and $l_{s,I}$ shares a vertex on $l_{s,I}$ with previously generated edges.

[Element pattern 3] a quadrilateral element formed by three vertices on $l_{s,O}$ and one vertex on $l_{s,I}$. It is generated if Step 3 is skipped.

2.3.4 Edge generation on the inner loop

Next, the process generates edges of M_q, $e_{(c+1)},\dots,e_d\,(c<d)$, on $l_{s,I}$. It generates the edges by connecting $w_{(j+1)},\dots,w_k$, which are generated in Step 3. The process finally subdivides r_s into several rough elements, as shown in Figure 5(d).

2.4 Topology modification for non-donut-like subregions

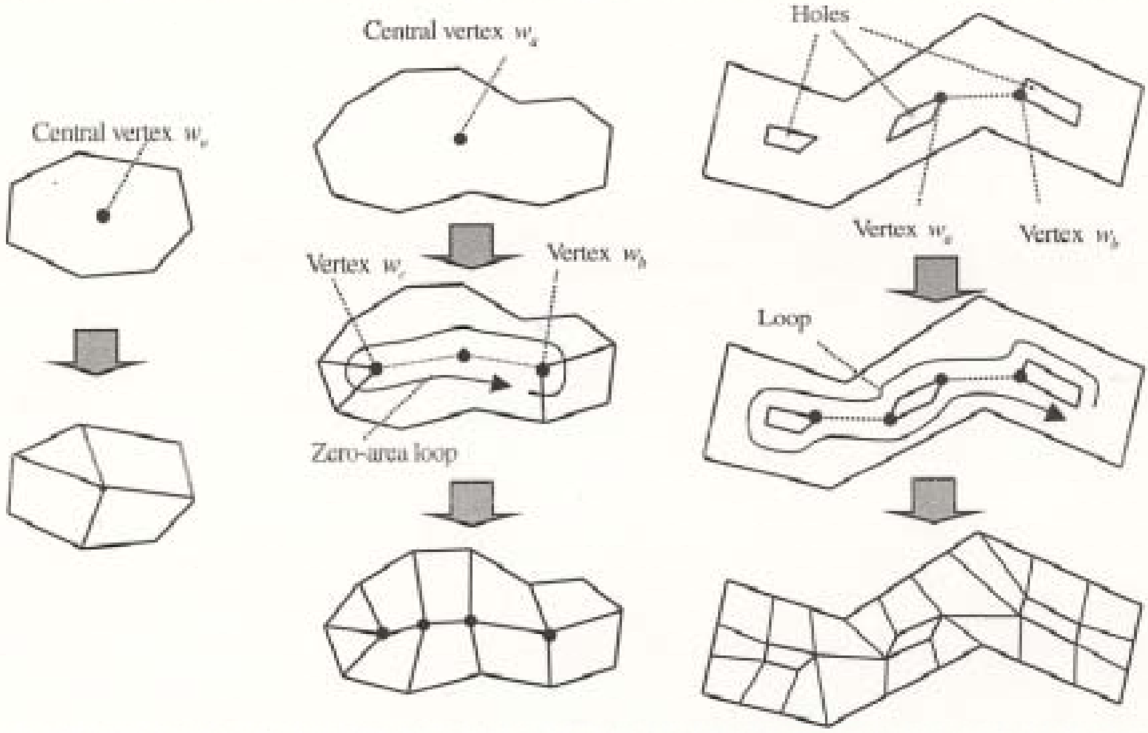


Figure 7. Topology modification. (a) Rough elements with a central vertex. (b) Rough elements with some interior vertices. (c) Rough elements with multiple holes.

Above-mentioned rough element generation method supposes that every subregion has one inner loop. Otherwise, the method modifies the topology of a subregion using the following algorithm.

If there is no hole in a subregion r_s, the process generates a zero-area hole in r_s. After generating vertices of M_q on the outer loop $l_{s,O}$, the process first extracts a central vertex of M_t in a subregion, and then generates a vertex of M_q, w_a, at the central vertex. It then calculates distances between w_a and each vertex on $l_{s,O}$. If all of the calculated distances are small enough, the process generates edges connecting w_a and each vertex on $l_{s,O}$, as shown in Figure 7(a). If some vertices on $l_{s,O}$ are too distant from w_a, the process first generates edges from the distant vertices on $l_{s,O}$ by using a triangle traversing process, similar to the traversal process shown in Figure 6(a). It then generates vertices $w_b, w_c, \dots$, at the ends of the generated edges, and they are connected, as shown in Figure 7(b). The set of connections of inner vertices is treated as a zero-area loop, $l_{s,I}$. The process then generates edges connecting $l_{s,O}$ and $l_{s,I}$ using the algorithm described in Section 2.3.

If there is more than one hole in r_s, the process generates a large inside loop that connects all holes. The process first extracts closest two vertices, w_a and w_b, on the boundaries of the two holes. The process then generates a segment that connects w_a and w_b, using a triangle traversal process similar to the process shown in Figure 6(a). This process is repeated until all holes are connected, and finally the process generates a loop by traversing all the segments and the inner boundaries of all of the holes, as shown in Figure 7(c). The process then generates rough elements using the algorithm described in Section 2.3.

2.5 Post-processing

Some post-processesing can improve the topology of a rough mesh. We implemented a function that converts two adjacent triangles into a quadrilateral, as shown in Figure 8(a). We also implemented a function that converts a triangle, whose only vertex is on the domain boundary, into a quadrilateral by adding a vertex on the boundary, as shown in Figure 8(b).

Figure 8. (a) Two triangles coupling. (b) Triangle to quadrilateral conversion.

3. Related works

Our surface patch construction method receives a fine triangular mesh as input and outputs a rough quadrilateral mesh. Various methods for generating rough triangular meshes or fine quadrilateral meshes have recently been proposed, however, our method is different. This section discusses some of them and compares them with our method.

3.1 Triangular to Quadrilateral Mesh Conversion

Mesh conversion is one way to generate a quadrilateral mesh, given a triangular mesh. Most of the mesh conversion methods [Bor96] [Hei83] [Ito98] [Joh91] [Lo89] [Shi95] first couple adjacent triangular elements and then convert the couples into quadrilateral elements. However, they cannot generate "rough" quadrilateral elements from "finer" triangular elements. Though it is possible to use the quadrilateral elements generated by the mesh conversion method as surface patches, this does not satisfy the second condition, because they are not larger than the input triangular elements.

3.2 Simplification of Triangular Meshes

Mesh simplification [Gar97] [Hop93] [Sch92] [Tur92] is another way to generate a rough mesh, given a fine triangular mesh, and is often applied to surface reconstruction applications. Most of them generate a coarse triangular mesh by culling off the input mesh by using various algorithms, such as vertex removal, edge and face collapse, vertex re-allocation, and so on. Though it is possible to use the coarse elements as triangular surface patches, most applications work better with quadrilateral surface patches rather than triangular patches. It is also possible to generate coarse quadrilateral surface patches by using triangular to quadrilateral mesh conversion methods after the simplification process, but it appears difficult to optimize the topology of the quadrilateral patches in such a conversion process.

3.3 Quadrilateral meshing for numerical simulations

Quadrilateral meshes are preferred for numerical simulation methods such as FEM, and various quadrilateral meshing algorithms have therefore been developed. The advancing method [Bla91] [Cas96] [Ras97] [Whi97] [Zhu91] is a famous algorithm for generating quadrilateral meshes well-aligned along the domain boundary. It is preferred in FEM since the domain boundary is often numerically important. It also seems that quadrilateral meshes generated by the method satisfy the above-mentioned conditions for surface patches. However, most of the advancing front methods generate meshes only from wireframe or surface models. It is possible to generate wireframe models from mesh domain boundaries; however, the method does not always generate proper meshes from geometrically complicated triangular meshes.

3.4 Topology construction of surface patches in conventional surface reconstruction methods

Various approaches of topology construction have been proposed for surface patches in conventional surface reconstruction methods. Krishnamurthy et al developed a user interface for manually constructing surface patches [Kri96]. Eck et al reported an automated implementation by using a combination of mesh simplification and triangle to quadrilateral conversion methods [Eck96]. Bajaj et al reported an automated implementation that constructs surface patches by using cross sections of a tetrahedral mesh generated from a set of given unorganized points [Bajaj95].

4. Experimental results

The method reported in this paper has been implemented in C and C++ on UNIX (mainly IBM AIX 4.2 and SGI IRIX 6.5), and Windows95/98/NT. We have applied the method to our automated blending surface generation system. Figure 9(a) shows an example of an input geometric model for the system. The model consists of three large planar surface patches, and three thin curved surface patches smoothly connecting the planar surfaces. It still contains a gap between the six surfaces; however, it is often difficult to manually create surface patches filling the gap and smoothly connecting the given surface patches with conventional commercial CAD systems. The target of our system is automated generation of such surface patches, so called fillets. The system first generates a triangular mesh filling the gap, and then deforms the triangular mesh by the mesh fairing method [Yam99a]. The method proposed in this paper then generates the topology of surface patches, and finally the system fits the patches to the triangular meshes by a least-square-based fitting method [Yam99b]. Figure 9(b) shows an example of surfaces generated by the system.

The left-hand images of Figures 10, 11, and 12 show the topology of surface patches generated by our method with input triangular meshes. The right-hand images of Figures 10, 11, and 12 show the result of fitting the patches to triangular meshes. Table 1 shows measurement metrics of the quality of the resulting models. Here, we calculated the value $\varepsilon_t = N_{tr} / N_p$ to measure the ratio of the number of triangular patches to the final patches, where N_{tr} denotes the number of triangular patches, and N_p denotes the total number of patches. We also calculated the value $\varepsilon_v = N_{iv} / N_v$, where N_{iv} denotes the number of vertices that do not satisfy the third condition as described in Section 1, and N_v denotes the total number of vertices. These results show that our method satisfies the first and third conditions. Also the figures show that our method satisfies the second condition, because the sizes of the surface patches are much larger than the sizes of the input triangular elements.

5. Conclusion

This paper reported a method to generate rough meshes from fine triangular meshes for the purpose of surface reconstruction. It generates surface patches well-aligned along the domain boundary, since it first generates belt-like subregions along the boundary and then subdivides them.

One of the possible future works is the analysis of the smoothness and continuity of the surface patches generated by our system. Such analysis may suggest other conditions besides the method and criteria proposed in this paper.

Another future work is an application of the method to closed triangular meshes. Our current target was the conversion of open triangular meshes, because that method has been applied to filling gaps in geometric models. Other consideration for closed triangular meshes may be necessary so that the method can be applied in the general computer graphics area.

References

[Baj95] Bajab C. L., Bernardini F., and Xu G., Automatic Reconstruction of Surfaces and Scalar Fields from 3D Scans, Proceedings of SIGGRAPH '95, pp. 109-118 (1995).

[Bla91] Blacker T. D., and Stephenson M. B., Paving: A New Approach to Automated Quadrilateral Mesh Generation, International Journal for Numerical Methods in Engineering, Vol. 32, pp. 811-847 (1991).

[Bor96] Borouchaki H., Frey P. J., and George P. L., Unstructured Triangle-Quadrilateral Mesh Generation, Application to Surface Meshing, Proceedings of 5th International Meshing Roundtable, pp. 229-242 (1996).

[Cas96] Cass R. J., Benzley S. E., Meyers R. J., and Blacker T. D., Generating 3-D Paving: An Automated Quadrilateral Surface Mesh Generation Algorithm, International Journal of Numerical Methodism Engineering, Vol. 39, pp. 1475-1489 (1996).

[Eck96] Eck M., and Hoppe H., Automatic Reconstruction of B-Spline Surfaces of Arbitrary Topological Type, Proceedings of SIGGRAPH '96, pp. 325-334 (1996).

[Gar97] Garland M., and Heckbert P. S, Surface Simplification Using Quadric Error Metrics, Proceedings of SIGGRAPH '97, pp. 209-216 (1997).

[Hei83] Heighway E. A., A Mesh Generator for Automatically Subdividing Irregular Polygons into Quadrilaterals, IEEE Transactions on Magnetics, Vol. Mag-19, pp. 2535-2538 (1983).

[Hop93] Hoppe H., DeRose T., Duchamp T., McDonald J., and Stuetzle W., Mesh Optimization, Proceedings of SIGGRAPH '93, pp. 19-26 (1993).

[Ito98] Itoh T., Shimada K., Inoue K., Yamada A., and Furuhata T., Automated Conversion of 2D Triangular Mesh into Quadrilateral Mesh with Directionality Control, Proceedings of 7th International Meshing Roundtable, pp. 77-86 (1998).

[Joh91] Johnston B. P., Sullivan J. M., and Kwasnik A., Automatic Conversion of Triangular Finite Element Meshes to Quadrilateral Elements, International Journal for Numerical Methods in Engineering, Vol. 31, pp. 67-84 (1991).

[Kri96] Krishnamurthy V., and Levoy M., Fitting Smooth Surfaces to Dense Polygon Meshes, Proceedings of SIGGRAPH '96, pp. 313-324 (1996).

[Lo89] Lo S. H., Generating Quadrilateral Elements on Plane and over Curved Surfaces, Computer and Structures, Vol. 31, No. 3, pp. 421-426 (1989).

[Ree97] Rees M., Combining Quadrilateral and Triangular Meshing Using the Advancing Front Approach, Proceedings of 6th International Meshing Roundtable, pp. 337-348 (1997).

[Sch92] Schroeder W. J., Zarge J. A., and Lorensen W. E., Decimation of Triangle Meshes, Proceedings of SIGGRAPH '92, pp.65-70 (1992).

[Shi95] Shimada K., and Itoh T., Automated Conversion of 2D Triangular Mesh into Quadrilateral Mesh, Proceedings of International Conference on Computational Engineering Science '95, pp. 350-355 (1995).

[Tur92] Turk G., Re-tiling Polygonal Surfaces, Proceedings of SIGGRAPH '92, pp. 55-64 (1992).

[Whi97] White D. R., and Kinney P., Redesign of the Paving Algorithm: Robustness Enhancements through Element by Element Meshing, Proceedings of 6th International Meshing Roundtable, pp. 323-335 (1997).

[Yam99a] Yamada A., Shimada K., Furuhata T., and Hou K., A Discrete Spring Model for Generating Fair Curves and Surfaces, Proceedings of Pacific Graphics '99 (1999).

[Yam99b] Yamada A., Furuhata T., Itoh T., and Shimada K., Curve and Surface Fitting to Scattered Points Based on a Discrete Spring Model, IPSJ Graphics & CAD technical report, 99-CG-96, pp. 31-36, in Japanese (1999).

[Zhu91] Zhu J. Z., Zienkiewicz O. C., Hinton E., and Wu J., A New Approach to the Development of Automatic Quadrilateral Mesh Generation, International Journal for Numerical Methods in Engineering, Vol. 32, pp. 849-866 (1991).

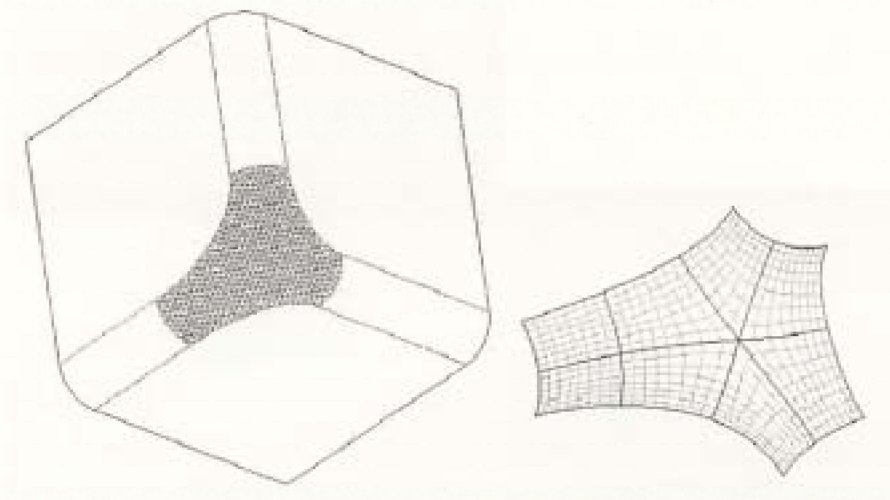


Figure 9. Example of an input geometry and output surface patches in our system.

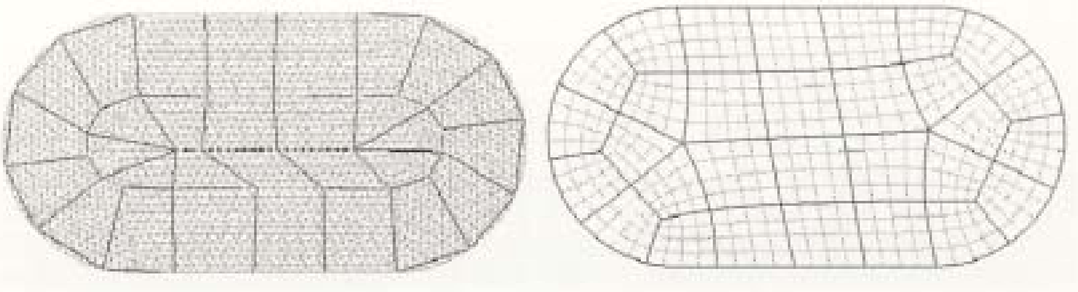


Figure 10. Result (1).

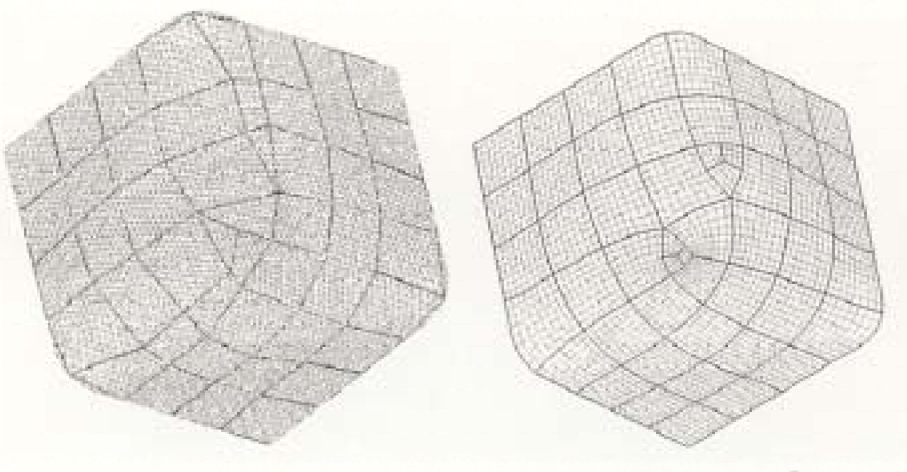


Figure 11. Result (2).

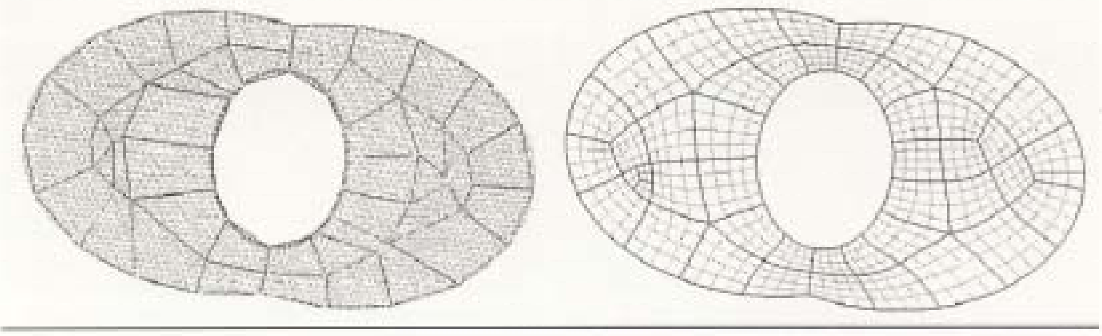


Figure 12. Result (3).

Table 1. Measurement values of surface patches.

value	ε_f	ε_v
Result (1)	0.0000	0.2000
Result (2)	0.0417	0.0500
Result (3)	0.0273	0.1321