The Algorithm of the Consistent Triangulation for the Set of Independently Triangulated Surfaces

A. Kryachko, A. Malashkina, J. Fedorova, T. Firsova
Nizhny Novgorod Software Technologies Lab
Nizhny Novgorod, Russia

Abstract

This paper presents the algorithm intended to create the consistent triangulation for the press-manufactured detail surface.

The source surface is uniquely projected on the OXY plane, excluding vertical domains, and represented as a set of triangulated patches. This representation is created by many CAD systems. The authors use the approach, which is different from the typical one in CAD systems, when the common patch boundary is created for the parametric surface representation during CAD repair. This approach can be used when the information on the boundary surface representation is unavailable.

Between the patches there are gaps and overlaps appeared while creating the model in CAD system, so the source triangulation has a bad quality. The algorithm is intended to create the source surface triangulation, approximating the surface not worse than the source triangulation. The obtained triangulation has a good quality, i.e. it does not contain degenerated triangles and coincident vertices and the angle between the normals is minimized.

Along with the algorithm, the authors developed the tools for surface visualization to debug the algorithm and evaluate the surface quality.

Keywords: *triangulation, patch, consistency.*

1. INTRODUCTION

Present engineering practice enables developing and processing geometry data with the help of CAD-systems. Quite often the CAD-system represents a surface as a set of *patches*, i.e. separate parts of the surface. This representation is called *trimmed surface*. Every patch is represented in the parametric space analytically, so 2D triangulation in the parametric space is typically used to triangulate such a surface. The surface mesh is obtained as a domain triangulation image in 2D. In this case, each patch is triangulated in its own parametric space regardless of adjacent patch triangulation that can cause gaps and overlaps between patches. But to use such a surface model, it has to represent an integrated consistent mesh. This mesh is comfortable for image visualization and surface geometry description to solve tasks on building a finite element mesh for the surface.

One of the common ways to solve this problem is to make adjacent patch boundaries consistent in the parametric space to delete gaps and overlaps before starting triangulation – CAD-repair task [1]. This research is based on a different approach, i.e. to obtain an integrated consistent triangulation based on "good" initial triangulation of every patch.

There is a CAD task targeted to build an elaborate surface for press-manufactured details. These surfaces have a predominant direction defined by the linear print pressing, or, in mathematical terms, these surfaces are uniquely projected on the plane perpendicular to the press motion direction. Without loss of generality, let this plane be OXY.

The article is arranged as follows, Section 2 contains the algorithm description, Section 3 offers analysis of timing and algorithm implementation, and, finally, Section 4 discusses surface visualization with the support of figures demonstrating the algorithm execution on several models.

2. ALGORITHM DESCRIPTION

Input algorithm data represent a set of vertices, and triangles built on the vertices. The triangles are resulted from the triangulation of CAD-surface patches. The surface can be uniquely projected on the OXY plane, but this property is not valid for the possible vertical domains. Gaps and overlaps between patches don't exceed the specified threshold, and triangulations of separate patches are not consistent. But every patch surface is "well-triangulated", i.e. the number of triangles is minimized with the specified triangulated surface deviation from the CAD-surface, and the angle between normals of the adjacent triangles is minimized.

There are five algorithm steps to creating a consistent mesh :

1. Pre-processing.
2. Build "quasi-consistent" mesh on the cloud of source points.
3. Enhance mesh quality with the help of source triangulation of separate patches.
4. Restore vertical domains in the triangulation.
5. Remove surface fragments located outside the source area boundary.

See the detailed description of every stage.

2.1 Preprocessing

Preprocessing is intended to verify source information, remove redundant information, and build data structures necessary for the algorithm execution. Preprocessing submits verifying source triangulation of every patch, building triangle adjacency structure for every patch, and collecting information on the vertical domains. The source vertices located closer than threshold distance to each other are merged.

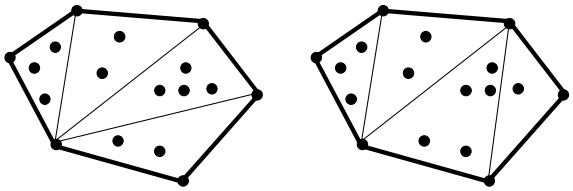
2.2 Building "Quasi-Consistent" Mesh on the Cloud of Source Points

The algorithm of building the initial "quasi-consistent" (except vertical domains) mesh represents a modification of the "greedy insertion" algorithm (M.Garland и P.Heckbert [2]). Unlike M.Garland and P.Heckbert's algorithm, this algorithm works on scattered data. Quick performance of this algorithm is based on the priority queue, which makes inserted point localization in the current triangulation unnecessary. The information on all source points belonging to the current triangles is stored that enables

quick re-counting deviations for points involved into re-organized triangles. All vertices, which are not internal for vertical patches and domains, are inserted into the mesh. *Vertical domain* is a connected set of vertical triangles that belong to one patch. So, the source patch including vertical triangles splits to several domains, a part of them is vertical. Further, such domains are also referred as patches.

At first, for the selected vertex set, a convex hull is built in the OXY plane and is arbitrarily triangulated. With the help of the local optimization method (Lawson [3]), the arbitrary triangulation is transferred to Delauney triangulation (Fig. 1). Every time when points are inserted into triangulation, choose the source point with the max deviation from the current approximated surface, and insert it into the triangulation. The deviation in a point is calculated as follows: $|z(x, y) - h(x, y)|$, where $z(x, y)$ is z-coordinate value in the source point (x, y) and $h(x, y)$ is z-coordinate value on the approximated surface (on a triangle) in the point (x, y).

Then, by triangulation edges, the inserted point is connected with the vertices of the triangle (or quadrangle if the point is on the edge) it belongs to (Fig. 2). The resulted triangulation is converted to the *shape-dependent* triangulation. The decision to replace the diagonal could only be made depending on the triangle shape. The triangulation conversion is specific because the edges incident to the inserted vertex are not replaced. It insures algorithm convergence. The shape quality of two triangles t and q, adjacent by the edge, is estimated as $area(t) \cdot area(q)/(diam(t) \cdot diam(q))$, where $area(t)$ is the area of the triangle t and $diam(t)$ is the diameter of the triangle t.



a) Arbitrary triangulation *b*) Delauney triangulation

Fig. 1. Building and convex hull triangulation.

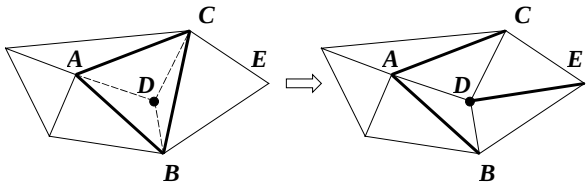


Fig. 2. Inserting the vertex into the triangulation.

Vertex D is inserted. Diagonal BC is changed to DE when building *shape-dependent* triangulation.

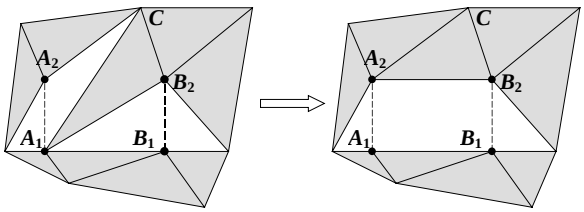


Fig. 3. Choosing the z-coordinate for the duplicate vertex.

Vertex couples A_1, A_2 and B_1, B_2 are duplicated, and vertices A_2 and B_2 provide the min error of the triangle A_2B_2C.

Let us define two triangulation vertices as duplicated if they have identical x- and y- coordinates when z-coordinates are different. Duplicated vertices result from vertical patches and often represent the ends of one of the vertical edges in a vertical patch. The duplicated vertices are crucial to describe the vertical patch triangulation. They require special processing in the insertion algorithm, otherwise they can be taken for the same vertex on the plane OXY. While inserting vertices into the current triangulation, duplicated vertices are traced, and the array *NextDublVerts* is created. In this array, *NextDublVerts*[i] is an index of the double-vertex of the i-vertex that follows it in the ordered sequence of i-vertex doubles on incrementing z-coordinate.

The error in the triangle is calculated as the max deviation in z-coordinate of the source points, whose projections are inside the triangle, from the triangle plane. If among the triangle vertices there are duplicated vertices, choose the z-coordinate (from the current duplicates) to provides the smallest error in the triangle. The corresponding field of the triangle structure refers to the index of this vertex (Fig. 3).

2.3 Enhancing Mesh Quality with the Help of Source Triangulation of Separate Patches

The source surface triangulation of every patch was created in 2D parametric space. Then, it was reflected into 3D physical space. Generally, the triangulation resulted from the integrated cloud of points according to the *shape-dependen*t algorithm differs from the source triangulation (on points of separate patches). To provide the same triangulation quality inside every patch, try to replicate the source triangulation inside patches. To achieve it, insert the source triangulation edges of separate patches to the consistent mesh that represents the triangulation of the whole area, except the vertical domains. This procedure provides the integrated consistent mesh and optimal triangulation inside every patch.

Edge insertion is an iterative process. All edges are inserted into the mesh, except vertical patch edges that don't belong to the vertical patch boundary. In the mesh there are vertices of every inserted edge because they are included into the cloud of points. For every inserted edge *e,* a set of triangles intersecting this edge is specified. Merge all triangles intersecting the edge *e*, and delete all internal edges from the obtained area. This results in a simple polygon divided by the edge *e* into two polygons. If every polygon is triangulated, and triangulations of these polygons are merged with the triangulation of the rest of the area, the resulted triangulation will have the inserted edge (Fig. 4). When inserting the edge, only edges intersecting the inserted one are deleted from the triangulation. So, when there are no overlaps of the source patches, none of the inserted edges will be deleted from the triangulation because, when projected on the plane OXY, the source edges don't intersect. However, if there are overlaps of two adjacent patches, their source edges projected on the plane OXY can intersect. In this case, the resultant triangulation will have only the latest of the inserted edges.

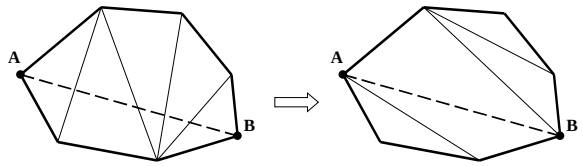


Fig. 4. Inserting the edge *AB* into the triangulation.

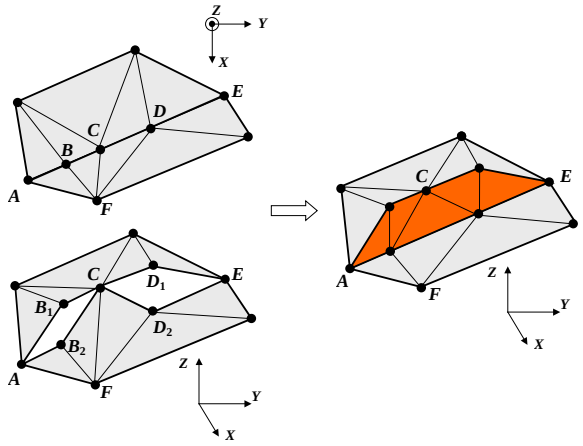


Fig. 5. Inserting the vertical patch (in distinct color) into triangulation. On the left, triangulation area before inserting the patch, on the right – after inserting.

The triangulation of each of two polygons obtained while inserting the edge is performed with "ear clipping" method [4]. To "clip", choose the "ear" of the best shape for the moment . To obtain a set of triangles intersecting the inserted edge, the *Anchors* array is used. In the array, there is an index to the triangle incident to this vertex. With the *Anchors* array and adjacency structure, the triangles intersecting the inserted edge are searched for $O(1)$.

2.4 Restoring Vertical Domains in the Triangulation

Every vertical domain is inserted into the triangulation independently. To insert the next vertical domain, define the consistency of its normal with the normals of the surface triangles, with which its triangulation is going to merge. If normal directions are not consistent, all triangles of the vertical domain are re-oriented. The triangle array of the vertical patch is added to the triangle array of the created triangulation. Then, choose an edge of the vertical domain boundary to be initial. Consequently follow the edges of the domain boundary to provide the merge process on each of the edges. Define the current mesh edges that compose the edge *e* projected on the plane OXY. Split the edges into two vertically, and insert corresponding vertical triangles between them (Fig. 5). Merge on the edge *e* includes defining mesh edges, whose projections belong to the edge *e* projection. If there are several such edges (see Fig. 5: edges B_2C and CD_2 , the edge B_2D_2 as the edge *e*), each of these edge vertices, except the *e* edge ends, is a subject to consider and check if a vertex is located on the edge *e* in 3D space. If it is located on the edge *e* on the plane OXY only (see Fig.5: vertex *C* for the edge B_2D_2), remove it from the surface triangulation from the corresponding edge side. To achieve it, remove from the triangulation the triangles (see Fig.5: B_2FC and FD_2C) incident to the point from the corresponding side, and triangulate the obtained polygon with the "ear clipping" method on the plane OXY. Each of the remained triangulation vertices located between the *e* edge ends is connected with the vertices of the only vertical patch triangle (see Fig. 5: CB_2D_2), which the edge *e* belongs to, with the help of triangulation edges. Then, cross-references are set between the obtained vertical patch triangles and corresponding triangulation triangles, adjacent to the *e* edge parts (or to the whole edge *e*).

2.5 Removing Patches Located Outside the Source Area Boundary

Since we are working with the convex hull of the projection on the plane OXY of the initial point cloud, the obtained triangulation can have triangles located outside the boundary. At the stage of quality enhancement, when source patches are inserted into the mesh, markers are assigned to their edges. Use these markers to remove triangles located outside the boundary. Consequently remove boundary triangles with unmarked boundary edges from the obtained triangulation.

3. PROGRAM IMPLEMENTATION AND TIMING ANALYSIS

This algorithm was implemented as a C++ class with the help of Microsoft® Visual C++ 6.0 compiler. See Fig.6 to define algorithm time dependency on the number of vertices in the source mesh. Used equipment: Pentium II 400MHz, 192Mb RAM.

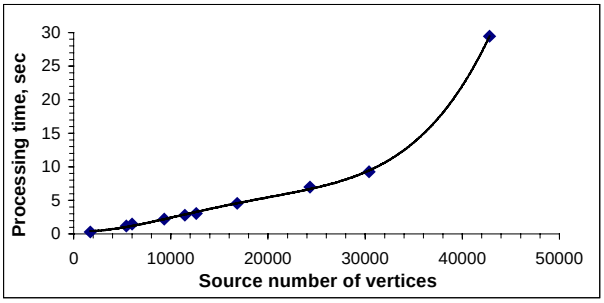


Fig. 6. Algorithm timing analysis.

4. VISUALIZATION PECULIARITIES

Figures 7 and 8 illustrate algorithm execution results on two models. The first figure (Fig. 7) illustrates the processing of several patches meeting area when gaps and overlaps don't exceed the specified threshold. The resultant triangulation is consistent and of good quality. It's clear that the algorithm execution results in the merge of several hither vertices. The second figure (Fig. 8) shows the gap between patches exceeding the threshold value (source surface assignment error). However, when vertices were inserted to create triangulation, that domain was triangulated to avoid gaps.

A specialized Windows MDI application using OpenGL library was developed to visualize surfaces.

If the mode of materials with metallic glance is used while rendering, and triangle normals are not weighed, it turns out comfortable to search for defects necessary in algorithm debugging. If such rendering method is used, there is a visual light gradient in triangulation areas with big angles between normals and degenerated triangles. This gradient enables easy diagnosing triangulation areas of bad quality (Fig. 9).

Extra application options provide algorithm debugging and include the following:

- Different modes of triangulation rendering: the whole triangulation, patch rendering, and triangle rendering. These modes enable selecting necessary objects with the mouse to get information on them, and to save the selected objects in a separate file.
- Triangulation area marking system. Options to mark triangulation areas (e.g., of bad quality), and to save marks in the source file with the corresponding comments.

5. CONCLUSION

This paper represents the algorithm to create surface consistent triangulation specified as a set of triangulated patches. The resultant triangulation represents an integrated patch and has a good quality, i.e. doesn't include degenerated triangles and big angles between normals.

To improve the mesh quality, another step is going to be added to the algorithm. Usually there is a high vertex density between patches due to a big number of patches converging to one point. Therefore, it's important to delete redundant vertices that not to provide the information on the surface shape. The model boundary is to be restored following not the information on edge insertion but the boundary shape specified outside.

6. REFERENCES

[1] A.Mezentsev and T.Woehler, *Methods and algorithms of automated CAD repair for incremental surface meshing.*

[2] M.Garland and P.S.Heckbert, *Fast triangular approximation of terrains and height fields,* 1997.

[3] C.L. Lawson, *Software for* C^1 *surface interpolation*, In Math. Software III, J.R. Rice, ed., Academic Press 1977, pp. 161-164.

[4] X.Kong, H.Everett, G.Toussaint, *The Graham Scan triangulates simple polygons*, Pattern Recognition Letters 11 (1990), pp.713-716.

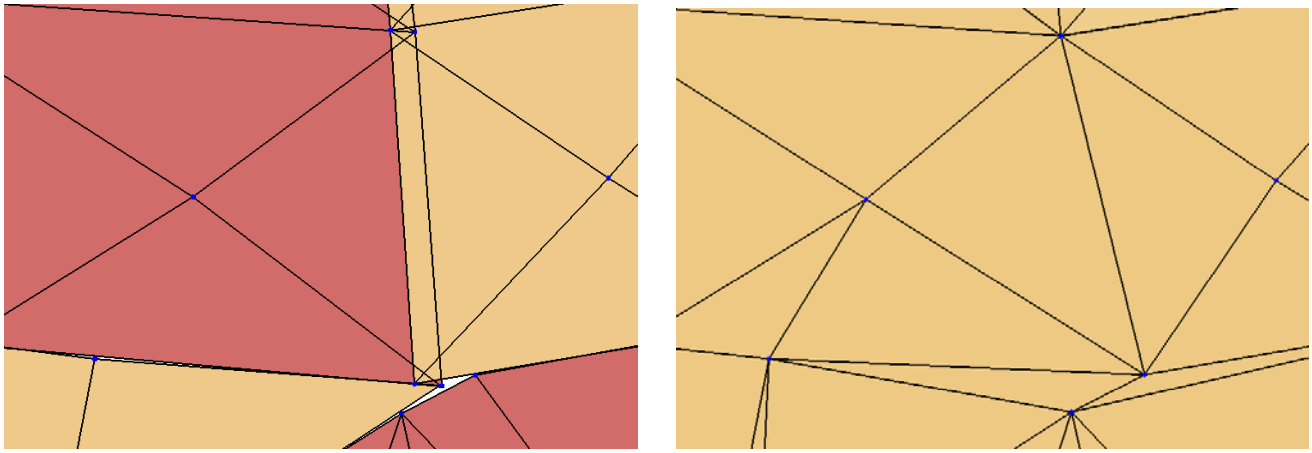
About Authors

Andrew Kryachko, Nizhny Novgorod State University, student.
E-mail: **kray@nstl.nnov.ru**, phone: (8312) 31-90-10

Anna Malashkina, Nizhny Novgorod State University, postgraduate student.
E-mail: **anna@nstl.nnov.ru**, phone: (8312) 31-90-10

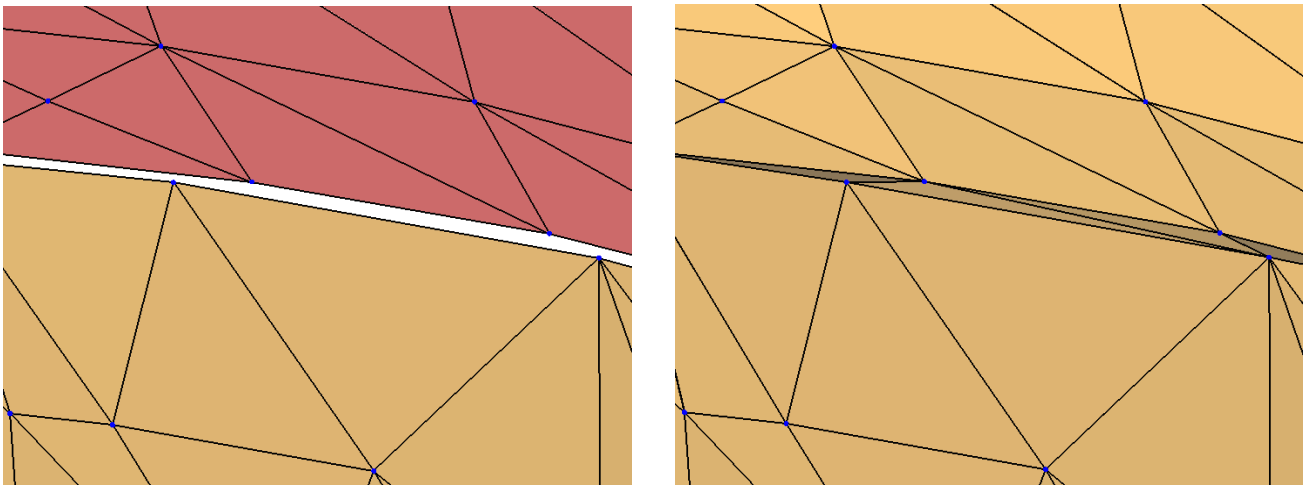
Julia Fedorova,
E-mail: **juf@nstl.nnov.ru**, phone: (8312) 31-90-10

Tatiana Firsova,
E-mail: **tafir@nstl.nnov.ru**, phone: (8312) 31-90-10



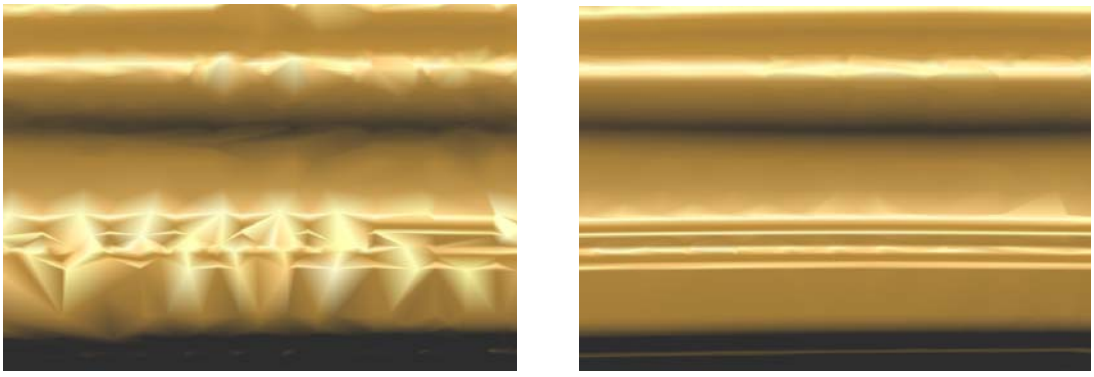
a) *b*)

Fig. 7. Result of the algorithm execution when patches (in distinct colors) are overlapped. *a*) an area of the source triangulation, *b*) the corresponding area after the algorithm execution.



a) *b*)

Fig. 8. Result of the algorithm execution when there is a large gap between two patches. *a*) an area of the source triangulation, *b*) the corresponding area after the algorithm execution.



a) *b*)

Fig.9. Two identical areas of the same surface: *a*) with degenerated triangles and big angles between normals, *b*) without degenerated triangles. In the first figure, there are distinct flecks corresponding to the bad triangulation quality. In the second figure, practically there are no flecks on the surface.