GENERATING HIGH QUALITY MESHES FOR INTERACTIVE EXAMINATION OF SURFACE QUALITY ON CAR BODIES

G. Sußner G. Greiner R. Grosso

Computer Graphics Group, University of Erlangen, Germany {sussner,greiner,grosso}@cs.fau.de

ABSTRACT

The use of reflection lines and specular high lights for the quality control of car body surfaces is an important issue in the development process of a car. The interactive examination is based on standard graphics tools such as the SceneViewer of OpenInventor which simulates reflection lines by using striped environment maps. The interpolation of texturing and shading values is critical and requires high quality meshes. Thus, element shape and size are essential. In this paper we present a new technique for the tessellation of trimmed surfaces. The result of the algorithm is a 2-manifold mesh with a low triangle count and a triangulation pattern which is best suited for the visualization.

Keywords:mesh generation, reflection lines, quality control

1. INTRODUCTION

Examining car bodies by reflection lines and specular high lights is an important issue in the development process of a car. It is not only used to discover curvature discontinuities. Such lines also reveal the character of the car body, no matter, whether it is for the overall shape or the details of certain parts. In order to shorten development time, designers and construction engineers meet frequently to examine the car shape and to tune the character of the car.

Nowaday, this quality control process is done by generating triangle meshes from the CAD-models and examining them via standard graphic tools, such as the SceneViewer of OpenInventor. The reflection lines are simulated by environment texturing using an image with horizontal or vertical lines whereas the high lights are simulated with a highly specular material and moving the light source around the part. The standard texturing method requires the RGB-values to be interpolated across the surface elements. Similarly, triangle shading interpolates colors and intensities. In order to obtain high quality images, the interpolation process is crucial. Thus, size and shape of the triangles are of special interest.

There exist many methods for generating triangular meshes out of NURBS patches ([1], [2], [3], [4], [5], [6]). Some of them are optimized for speed whereas others are optimized for lowest triangle count or best approximation. There also exist a large number of techniques for the tessellation of trimmed patches for car body parts. These methods are optimized with respect to the reflection lines, which is crucial for the quality evaluation of the surface.

In this paper we first present a set of common requirements for tessellated surfaces. Since none of the existing methods perfectly fits these requirements, we propose a new algorithm which generates high quality tessellations optimized for the quality control of trimmed patches of car body parts.

Car parts usually consist of several NURBS patches. Tessellating them individually often results in a speckled rendering on the screen. These artefacts are produced by T-vertices (hanging nodes), holes or cracks along adjacent faces. Since the main goal is a high quality visualization, we have to eliminate distortions. Most of the CAD-programs declare the adjacencies between patches, but only at the end of the construction stage. In order to improve the development time, car parts are visualized from the very beginning up to the end of the car design. Therefore, an efficient quality control system has to be able to find out the adjacencies of the single faces independently of the information provided by the CAD-system. We address this problem in the second part of the paper, where we present a new algorithm which automatically finds out all adjacent surfaces and creates a consistent mesh.

2. PRELIMINARIES

The car industry used to have cube like rooms with many parallel light rows on the wall and on the ceiling to examine car bodies for surface errors and design issues. They put a car body with a highly reflective surface into that room and examined the reflected light rows, called *Reflection lines*, on the surface. This is known as *Cubing* process. Since you need a physical body for that, you have to cut the current CAD model out of a block of UREOL. This is a very expensive and time consuming process. Digital examination on a computer screen offers a very attractive method to tune this process. It allows several cubing sessions, even at one day, i.e. you can examine several different designs and compare them to each other. Additionally, errors in surfaces can be corrected and re-examined instantly, i.e. in minutes instead of weeks.

For this, we use standard graphic tools available at modern graphic workstations, like the SceneViewer of OpenInventor. We use a sphere environment texture to simulate the reflection lines and display a triangular mesh in the viewer, where you can zoom in/out and rotate the model having the same effect as walking around the physical car model in the cubing room.

Since we use interactive tools for visualization it is mandatory to keep triangle count low. Our goal is to create meshes with as less triangles as possible but as much as necessary to meet both a graphic quality criterion regarding displaying of reflection lines and specular high lights and a shape quality criterion for proper approximation of the shape within a given tolerance ε.

2.1 Issues on Reflection lines

Reflection lines have exact one order less continuity than the considered surface, i.e. if the line is just G^0-continuous, the surface is G^1-continuous, if it is even G^0-discontinuous, you have got a fold in the surface, Fig. 1.

While environment mapping is on, a reflection vector from the viewer to a triangle's vertex is calculated by the graphics hardware according to the normal at that vertex. That means, we have three texture coordinates for a triangle. The graphics hardware now pastes the bilinearly interpolated texture fragment on the triangle.

Due to the linear interpolation, reflection lines are approximated by linear line segments, which depend on the size of the triangles. This means, if you examine the object from a relatively far distance, the reflection lines are approximated well. But, when zooming in, the quality of the reflection lines deteriorates, because the displayed triangles are larger on the screen, causing coarse approximation of reflection lines. Reducing triangle size will improve image quality, but will also increase graphics load.

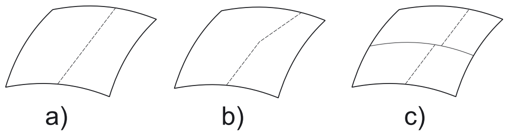


Figure 1: This figure shows the behavior of reflection lines. a) Surface is G^2-continuous b) Surface is G^1-continuous c) Surface is G^0-continuous

In Fig. 2 one clearly sees, that thin triangles or even an improper triangulation, drastically influence the quality of the displayed reflection lines. These artifacts can be avoided by using a regularly triangulated mesh as it is shown in Fig. 2(a,b). This triangulation is based on nearly equidistant iso-parametric lines in 3D.

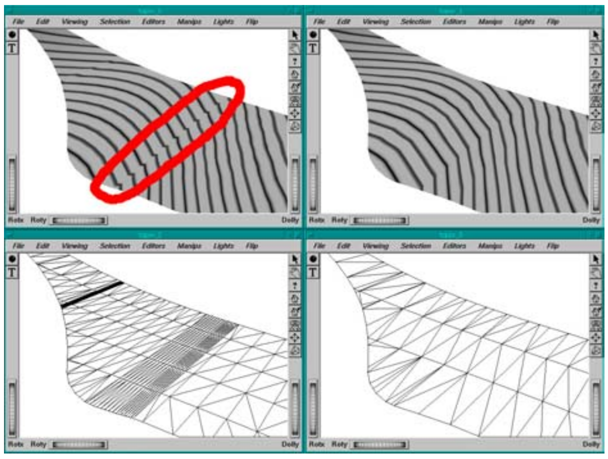
2.2 Issues on Specular High Lights

For specular high lights a regular pattern of triangles is at least as important as for reflection lines. With specular high lights you can consider certain design themes. For example, two parts should be constructed and placed in such a way, that there is no discontinuity of the specular reflection across part boundaries, Fig. 3.

3. RELATED WORK

The tessellation of trimmed surfaces is not a trivial task, especially in the case of automatically created surfaces. In order to keep a high frame rate for rendering, we want to have as few triangles as possible. We are looking for a method which minimizes interpolation artifacts and generates a triangle mesh with regular patterns by providing parameters of max. approximation error and max. grid size. Grid size means the distance between two vertices on iso-parametric lines.

In this section we analyze different approaches and compare their results. We show, that none of these approaches actually fit our demands, i.e. the generation of high quality meshes for the visualization of reflection lines with low triangle count.


(a) Artifacts caused by thin triangles

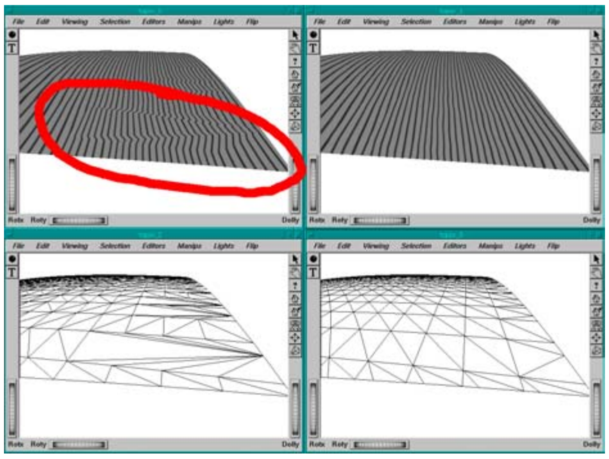

(b) Artifacts caused by improper triangulation

Figure 2: Examples of artifacts on reflection lines pretending an error in the surface representation

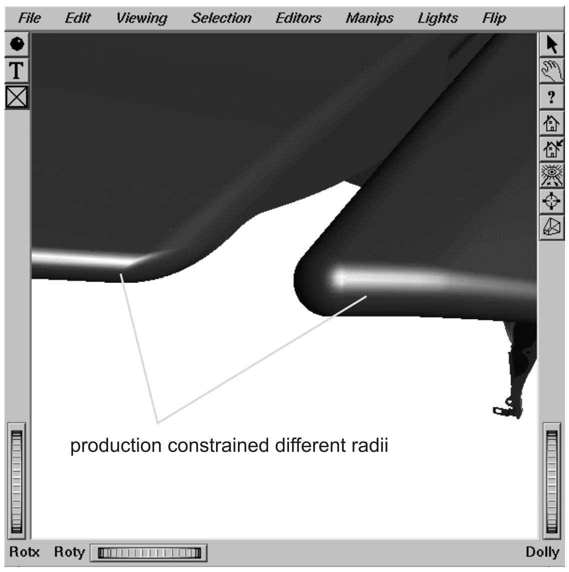


Figure 3: Course of high lights across parts. The radii of both parts cannot be changed. The task is to arrange both parts to get a high light course that is as smooth as possible.

3.1 CATIA OpenInventor Export

CATIA has an integrated OpenInventor export. This module produces excellent meshes with regard to visualization quality. However triangle count could be lower. In this paper we present a method which produces meshes with equivalent visual quality, same approximation error, but use only about 60% of the triangles.

3.2 GLU-Tessellator

The GLU-Tessellator comes with OpenGL 1.2 as a standard, with OpenGL 1.1 it may be available as an extension. We used it for this paper on an IRIX 6.5.8 system. Since it is implementation dependent, result may differ from system to system. For the GLU-Tessellator, you have to provide the surface as NURBS patch. For the trim curves, you have the choice of providing the trim curves also as NURBS curves or as already tessellated trim polygons. Providing NURBS curves, often causes the tessellator to abort, because the given trim curves are self-intersecting. So we decided to tessellate the trim curves by our own algorithm. Additionally, it is only possible to give one desired tessellation parameter, either ε approximation error or maximal grid size.

As you can see in Fig. 4(a), the tessellator produces a high triangle count and may also produce artifacts like in Fig. 2.

3.3 Minimizing triangle count

A possibility to reduce triangle count is to do mesh reduction. We reduced a mesh, created by the GLU-Tessellator, with the algorithm of Campagna [7]. The results are shown in Fig. 4(b,c). The resulting mesh has a much lower triangle count but the reflection lines are disrupted pretending an error in the surface.

A method presented by Klein [8] is based on a different approach. He takes a large set of domain values and inserts incrementally those values, which minimize the resulting approximation error until a certain threshold for the error is reached. The local area of the domain around the inserted vertex is re-triangulated to fulfill 2D-Delaunay criteria. This results in a very low triangle count but may lead to disrupted reflection lines like in Fig. 2(b) or Fig. 4(c).

Steinbrenner et al. [9] introduced an algorithm which produces meshes with a similar irregular structure like reduced meshes. Thus you have similar results, if triangle count is low.

3.4 Equidistant domain values

Here, we use equidistant parameterization of $n_u \times n_v$ so, that n_u and n_v are the lowest integer to fulfill the given 3D criteria. As you can see in Fig. 5(a), equidistant iso-parametric lines does not necessarily mean 3D equidistance and often produce much more triangle count and may lead to same artifacts as the GLU-Tessellator.

4. THE TESSELLATION ALGORITHM

In this section we present a new approach for the generation of meshes with a low triangle count and a regular inner structure, which are well suited for surface visualization, avoiding any visual artifacts as described above.

4.1 Automatic basic CAD Repair

In this paper we only consider data sets, which are exported from CATIA, a very popular CAD-program in the car industry, in the VDAFS file format. VDAFS is a standard interface defined by the German car maker association (VDA). Since this data is very complex and difficult to handle, even for sophisticated commercial packages, the exported data is in many cases inconsistent. Thus, the first processing step is CAD repair.

We only handle a subset of the VDAFS interface. We restrict to trimmed and untrimmed surfaces and 2D domain trim curves which are needed by trimmed surfaces. These are:

- A SURFACE consists of $m \times n$ Bézier patches which share a common parameter domain $[u_0, u_1] \times [v_0, v_1]$.
- A domain trim curve CONS is defined by k 2D-Bézier curves which also share a common parameter domain $[w_0, w_1]$.
- A FACE references a SURFACE and can have several trim curves. The first one defines the outer shape of the FACE, whereas the following cut holes out of the SURFACE. Each trim curve consists of at least one CONS element and has to be a closed curve.

In order to be general and accept any type of input, $n \times m$ Bézier patches could be converted into a single NURBS patch. Nevertheless, the CAD system creates Bézier patches of order 20, e.g. for blending surfaces. Creating a single NURBS patch means, that all $n \times m$ sub patches have the same highest order. Since performance is also an issue, we decided just to force conditions of adjacent continuous Bézier patches, i.e. all patches of one row have to have the same v-order and all patches of one column have to have the same u-order. Therefore, there is no restriction for NURBS input data, because all sub patches already have the same order.

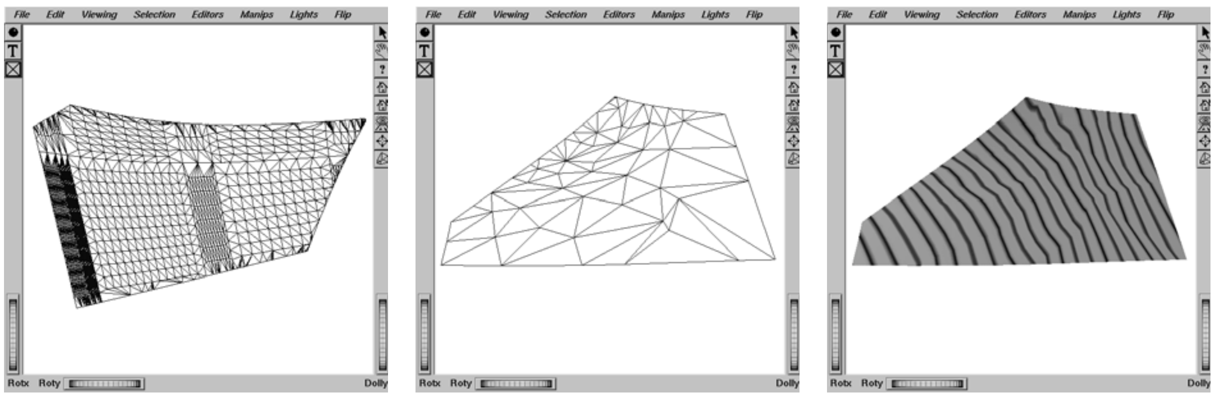
First, all rows and columns are inspected and checked, if the conditions for adjacent Bézier patches are fulfilled, i.e. u- and v-orders, the right domain values and common control points. Different orders are automatically corrected by increasing the degree. If there are any errors, except for different orders, the face will not be accepted.

Then, G^1-continuity in the trimmed area along adjacent patches is controlled. If a surface is detected, which does not satisfy this condition, it is automatically detected and triangulated. During the visualization, these surfaces are colored to indicate the problem. Usually, SURFACES which are not G^1-continuous are created with the help of blending functions or are modeled individually by hand.

Finally, the trim curves are checked. For the same reasons as above, we use the Bézier form. This step is much more critical, because a CONS often consists of many Bézier curves of lower and higher order. The resulting NURBS curves consist of sub curves all of the same order, resulting in a performance penalty.

The trim curves consist of several CONS curves and have to be closed. Thus, each CONS have to be checked for consistence, i.e. common control points and domain values. Finally, the CONS curves are connected to form a closed curve, Fig. 7.

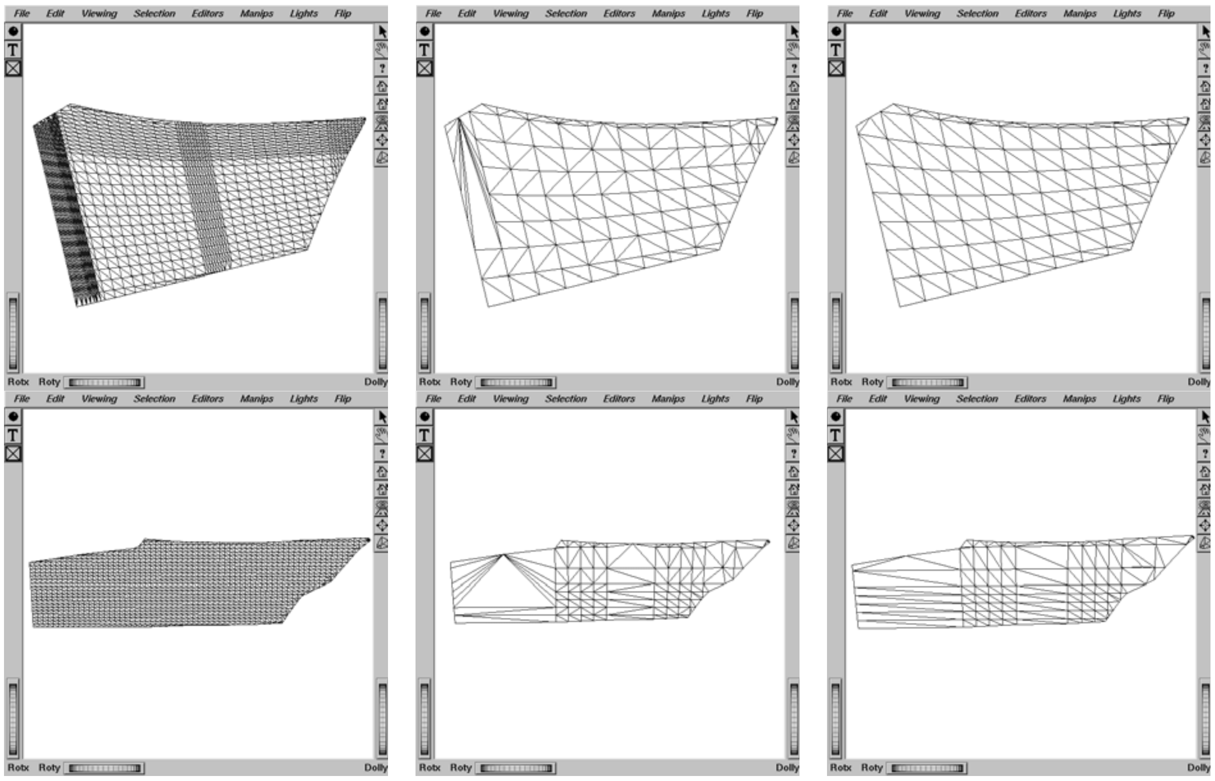
The experience shows that real CAD data sets often contain non-closed trim curves. For example, CONS



(a) GLU-Tessellator of IRIX 6.5.8 produces lots of triangles so that reflection lines are rendered correctly

(b) Reflection lines are rendered ambigously. The characteristics of the reflection lines is wavy due to the low triangle count and irregular structure of the mesh. So you cannot be sure that there is no design flaw in the surface.

Figure 4: Meshes with high and low triangle count.



(a) equidistant domain values

(b) Delaunay triangulation

(c) Preserving iso-parametric lines

Figure 5: 5(a) shows equidistant iso-parametric lines in 2D. 5(b) and 5(c) show equidistant iso lines in 3D but different triangulation

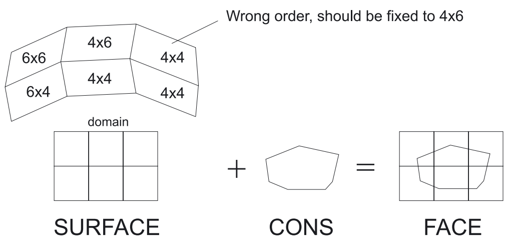


Figure 6: Handled types of the VDAFS standard.

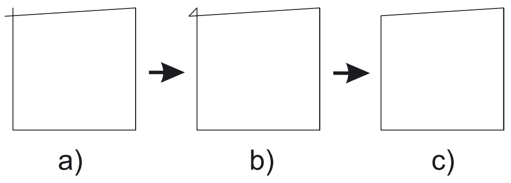


Figure 7: CATIA often exports trim segments which are not connected properly.

curves which actually should be adjacent, slightly differ, or in the case that degenerated SURFACES are used, e.g. "suit-case-corner", one CONS element is missing, i.e. the one which has a 3D length of 0. Since we use domain based trim curves, we have to fill the missing segments and force the trim curve to be continuous and closed, see Fig. 8.

4.2 Adaptive domain values

In order to obtain 3D equidistant iso-parametric lines and a regular grid, it is necessary to adapt each domain direction (u,v) individually. Then, we obtain the grid by the tensor product. Each u-value can be found by first approximating the arc length to the desired grid size, and then by satisfying the imposed geometric constraints. In Fig. 5 the difference between an equidistant parameterization in 2D, Fig. 5(a), and 3D, Fig. 5(b,c) is shown.

4.3 Inserting Trim Curves

The trim curves, which are applied in domain space, are tessellated in a similar manner so that they also fit the given criteria, i.e. maximal arc length in 3D and maximal geometric deviation.

A natural way to insert trim polygons would be to insert the polygon segments and generate a constrained Delaunay triangulation ([10]). Nevertheless, as one can see in Fig. 5(b), this approach does not lead to the desired results. A 2D-Delaunay criteria does not necessarily fit into 3D space, in particular along trim curves. This fact may produce artifacts when display-

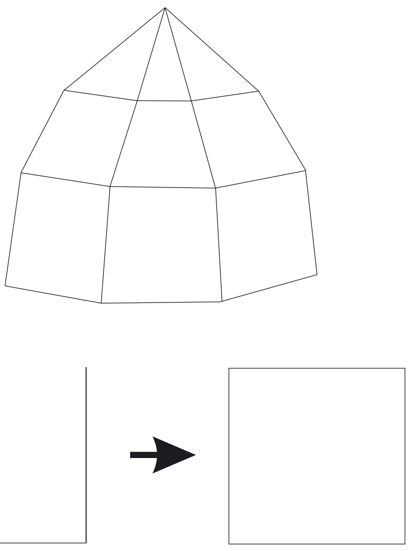


Figure 8: If a row of control points coincides, CATIA omits the corresponding trim segment, resulting in a non-closed trim curve.

ing reflection lines.

The solution is to preserve the grid structure of the iso-parametric lines. First, the parameter domain is subdivided into quadrilaterals as it was shown above. Then, the polygons resulting from the trim curves are inserted into the grid by clipping each polygon segment against each rectangle with the Cohen-Sutherland clipping algorithm. If a grid rectangle is intersected, a constrained Delaunay triangulation of the rectangle and the clipped segments is done. Those rectangles which are not intersected by the trim polygons can be handled in two different ways. First, the rectangle is used for rendering with the help of an OpenGL **GL_QUAD** object, Fig. 9. Second, the rectangle is split into two triangles (chosen so that they best fit for surface approximation). This approach preserves the grid structure as shown in Fig. 10(b).

4.4 Stitching Adjacent Meshes

Car parts usually consist of many trimmed surfaces. The adjacencies of those faces are usually determined semi automatically by the constructor in the CAD program. This is a very time consuming task and is usually done at the final construction stage. If this information is missing, adjacent faces are tessellated individually. Thus, along common boundaries, T-cracks arise (see Fig. 11), which result in speckled images. The adjacencies are automatically determined within a given tolerance ε by comparing each mesh with all other meshes. The search is carried out with the help of a global BSP-tree, which results in a algorithm of

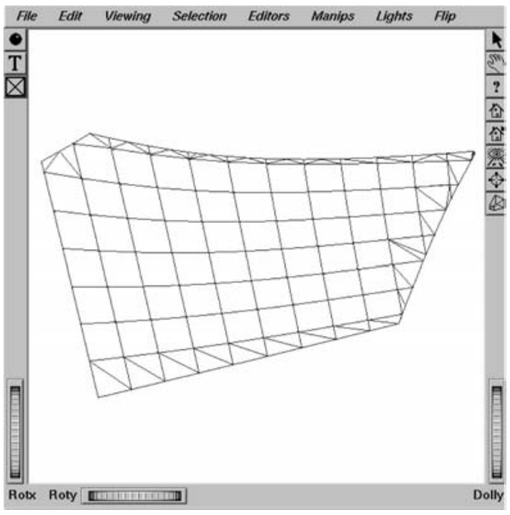


Figure 9: Grid rectangles are left as GL_QUADs

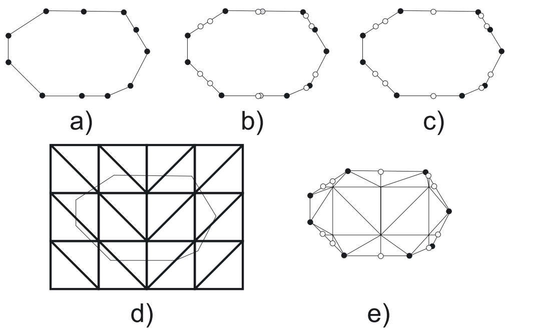


Figure 10: Generation of the domain triangulation. a) Tessellation of trim curve b) inserting iso-values c) polygon reduction d) generating triangulation for untrimmed surface e) inserting trim polygon and retriangulation of concerned quads

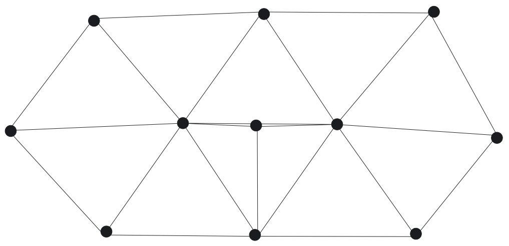


Figure 11: T-Junctions along common face boundaries.

complexity $n \log n$.

4.5 Fixing Orientation

Adjacent meshes may be differently oriented, see Fig. 14(a). In order to obtain a 2-manifold single mesh, this problem has to be fixed.

For this purpose we use the common boundary segment information obtained during the stitching process. If we have found a common boundary edge, we decide, if the neighbor meshes have same orientation, Fig. 14(b). If this is not the case, the orientation of one mesh is changed. The mesh is marked and will not be touched again, see 5.3 for implementation details.

4.6 Reducing Stitched Boundaries

Since adjacent patches usually do not have common vertices, the stitching step introduces many additional triangles, because nearly every boundary edge is split.

At the final step of our algorithm all meshes are merged into one single mesh. Then, the number of edges is reduced by edge collapsing along the former patch boundaries, which reduces the triangle count to an amount which is roughly the same as before stitching.

5. IMPLEMENTATION DETAILS

The key issue of the implementation is the extensive use of BSP-trees for vertices and control points. This allows for a fast identification of common vertices and vertices within a given distance. We use one global BSP-tree for all meshes and one global BSP-tree for the control points of all Bézier patches. While tesselating each FACE, we use a common BSP-tree for the domain values and trim polygons. For each vertex, we store a back reference to each mesh and each triangle. Furthermore, for each control point, we store a back reference to each face.

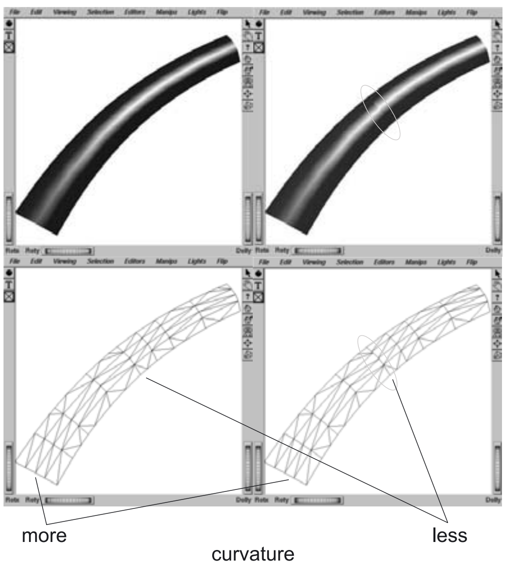


Figure 12: Highlight discontinuity across adjacent faces. Here, two faces have common control points but have different curvature. a) Parameterization is done for both faces simultaneously. b) Each face is parameterized individually, resulting in a different parameterization for each face, disrupting the course of the high light.

5.1 Tessellating Trimmed Surfaces

Due to different curvatures, adjacent FACES may have different parameterizations at the common boundary. This produces artifacts as it is shown in Fig. 12. This can be avoided, if both FACEs have identical control points at that boundary, which can be achieved by using a common parameterization for both FACEs.

For tessellating a trimmed surface, we first get the parameterization in the u- and v-direction, according to the algorithm presented below. A 3D approximation error and grid size have to be specified. Each trim curve is polygonized according to the given 3D criteria in a similar manner as the domain parameterization. Afterwards, we have to do CAD repair and remove loops. Then, the $u \times v$-parameterization is inserted into the trim polygon, i.e. the polygon is intersected with each iso-parametric line. This is necessary to preserve iso-parametric lines. Since the polygon may be now oversampled, we apply a polygon reduction, with respect to the given 3D criteria, where the points on an iso-parametric line must remain in the polygon.

The following code example shows the principal algorithm for domain interval $[u_0, u_1]$.

```
for(i=0; i<numParams; ++i) {
  x0=u[i]
  x1=u[i+1]

  // first, check grid size
  while (max_arclen(u[i],x1) >= gridsize) {
    x2=(x0+x1)/2
    if (max_arclen(u[i],x2) >= gridsize) {
      x1=x2
    } else if (gridsize-max_arclen(u[i],x2) >= eps) {
      x0=x2
    } else {
      x1=x2 // reached enough iterations
    }
  }

  iter_ctr=0
  x0=u[i]
  // now check approximation error
  if (max_dist(u[i],x1) >= eps) {
    while (max_dist(u[i],x1) >= eps
           && iter_ctr < max_iter) {
      if (max_dist(u[i],x2) >= eps) {
        x2=x1
        x1=(x0+x1)/2.
      } else {
        // yes, we are below eps, but we want to be
        // as close as possible to the max. geometric
        // deviation
        x0=x1
        x1=x2
        iter_ctr++
      }
    }
  }

  insertParameter(u,x1)
}
```

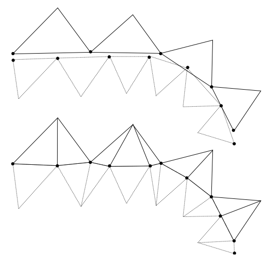
The next step is to create a regular grid from the iso-parametric lines. One still has the freedom to subdivide the quadrilaterals into triangles. In our implementation, we use triangles. For the subdivision, we use the diagonal which gives the best approximation to the surface.

Now we insert the trim polygons into the 2D grid. The quads, which are affected by inserting polygon segments are retriangulated according to Delaunay criteria. Just subdividing the quad's triangles results in a bad shaped triangulation and may lead to numerical problems. Afterwards, triangles, which lie in the exterior of the trimmed area are removed.

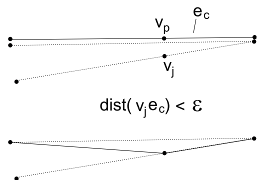
Finally, we lift the 2D mesh into 3D space. Thin and small triangles are eliminated via mesh reduction on the mesh boundary according to the given 3D geometric error criteria.

5.2 Stitching Adjacent Meshes

In contrast to [9], who first find common boundary curves of the surfaces and then use this information to mesh the FACEs afterwards, we mesh each FACE individually and stitch the resulting meshes together. Since we do not know anything about the location and orientation of each face, we potentially have to compare each mesh with all other meshes. To avoid this $O(n^2)$ problem, we find the longest boundary edge



(a) Stitching meshes along their common boundaries



(b) If we do not check back that e_c is closest edge to v_p this situation can occur

Figure 13: Stitching meshes

r_{max} of all meshes $M_i, i \in I$. Then, for each boundary vertex v_j of M_i, we find all boundary vertices V_J of $M_k, k \in I, k \neq i$, within a radius of r_{max} to v_j. Since we use a common BSP-tree for all mesh vertices we have reduced the complexity to $O(n \log n)$. Now, from all boundary edges, adjacent to $v' \in V_J$, we determine the closest edge e_c, which has a distance lower than ε, and project v_i onto e_c, obtaining v_p. If no edge e_c was found, we continue with next vertex v_i.

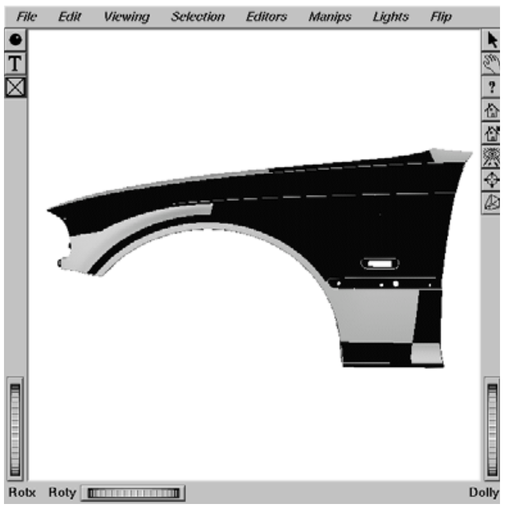
In Fig. 13(b) we present a case for which, we have to check back, whether the boundary edges, containing v_i, are really the closest boundary edges to v_p. This is done in a similar way as above. If the check fails, we ignore the split and proceed with the next vertex v_i. If the distance between v_p and the end points of e_c is lower than ε, we move the corresponding end point into v_i, otherwise we split e_c at v_i.

5.3 *Fixing Orientation*

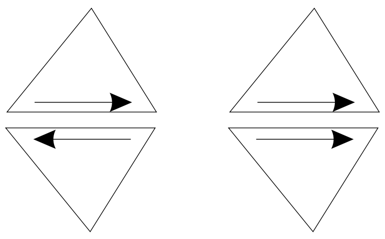
After stitching all meshes together, we give all adjacent meshes the same orientation. For this, we put all meshes in a list L_{remain}. We then remove the first element of L_{remain} M_i and find all meshes $M_I \in L_{remain}$, which have a common edge with M_i. For each $M_j, j \in I$, the orientation is changed, if necessary (see Fig. 14(c)) and M_j is removed from L_{remain}. Now, we repeat this procedure with each $M_j, j \in I$. If L_{remain} is empty, all adjacent meshes have the same orientation. Finally, all meshes are merged into one single 2-manifold triangular mesh.

5.4 *Edge Reducing*

The stitching step requires that many triangles are inserted to close cracks. In order to reduce the triangle count down to its original size, we eliminates triangles along the former patch boundaries. This considerably



(a) FACEs with bright color have counterclockwise orientation and those rendered dark are oriented clockwise



(b) Adjacent triangles with same and different orientation

Figure 14: CAD models often consist of different oriented meshes

reduces the number of triangles roughly to unstitched mesh (see Fig. 15).

6. RESULTS

In Table 1 we compare our method against the CATIA-OpenInventor export and the GLU tessellator. For each data set, each patch is tessellated individually. Then for our approach and for the GLU tessellator the merging step is done. The CATIA export is also capable to produce single meshes if the data sets were "skinned" in CATIA. The GLU tessellator produces about five times more triangles than the 3D grid method, which considerably deteriorates the rendering performance. Compared to meshes exported from CATIA we produce meshes with a triangle count of about 60%. Considering merged meshes the rate is even better due to our reduction step along the patch boundaries.

Fig. 15(c) shows a zoomed area of a car part with all its FACEs individually tessellated, then stitched and finally reduced along their FACE boundaries. As you can see, the stitching step with following reduction along patch boundaries increases the triangle count only marginally, about three percent. The skinned meshes form CATIA export have about a 15% higher triangle count than its individually tessellated export.

As a result, our approach allows to load more car parts at once for interactive visualization or it delivers better visualization quality for the same number of parts.

7. CONCLUSION AND FUTURE WORK

In this paper we discussed the effects of different tessellations strategies for trimmed surfaces. These tessellated surfaces are used for the evaluation of the surface quality with the help of reflection lines. A fully automatic algorithm to detect adjacencies among tessellated trimmed surfaces was presented. These surfaces are then merged into a single 2-manifold mesh, This high quality mesh is suitable for interactive reflection line examination.

The visualization quality can be further improved by using environmental dot-product bump mapping and normal maps which is supported by recent hardware. Using this technique we need a parameterized mesh which is tessellated just to an ε tolerance. A regular triangulation pattern is no longer necessary. Instead one has to create normal maps while tessellating surfaces.

References

[1] Abi-Ezzi S.S., Subramaniam S. “Fast Dynamic Tessellation of Trimmed NURBS Surfaces.” *Eurographics*, pp. 107–126, 1994

[2] Chew L. “Guaranteed-Quality Mesh Generation for Curved Surfaces.” *Proceedings of Ninth Annual ACM Symposium on Computational Geometry*, pp. 274–280, 1993

[3] Klein R. “Linear Approximation of Trimmed Surfaces.” *The Mathematics Of Surfaces VI*, 1994

[4] Schumaker L.L. “Computing optimal triangulations using simulated annealing.” *Computer Aided Geometric Design, 10(3)*, pp. 329–346, 1993

[5] Schumaker L.L. “Triangulations in CAGD.” *IEEE Computer Graphics and Applications, 13(1)*, pp. 47–52, 1993

[6] Sheng X., Hirsch B.E. “Triangulation of trimmed surfaces in parametric space.” *Computer Aided Design, 24*8)*, pp. 437–444, 1992

[7] Campagna S. *Polygonreduktion zur effizienten Speicherung, Übertragung und Darstellung komplexer polygonaler Modelle.* Dissertation, Universität Erlangen-Nürnberg, 1998

[8] Klein R. *Netzgenerierung impliziter und parametrisierter Kurven und Flächen in einem objektorientierten System.* Dissertation, Universität Tübingen, 1995

[9] John P. Steinbrenner N.J.W., Chawner J.R. “Fast Surface Meshing on Imperfect CAD Models.”, 2000. Proceedings of the 9th International Meshing Roundtable

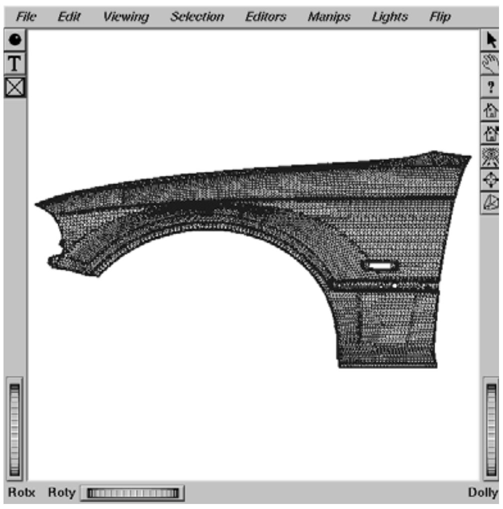
[10] Shewchuk J.R. “Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator.” 1996

[11] Woo M., Neider J., Davis T., Shreiner D. *OpenGL Programming Guide.* Addison-Wesley, 3rd edn., 1999

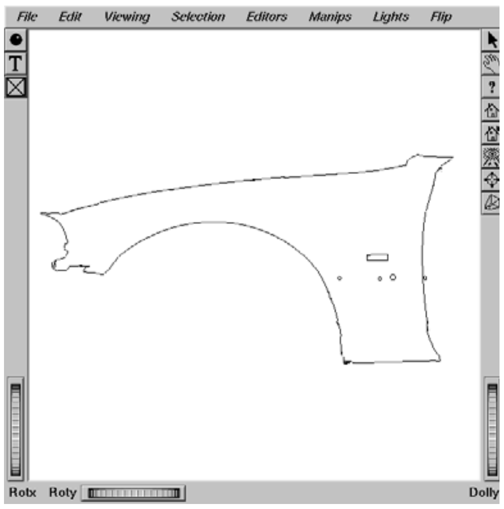
[12] Farin G.E. *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, 4th edn., 1997

[13] Mezentsev A.A., Woehler T. “Methods and Algorithms of Automated CAD Repair for Incremental Surface Meshing.”, 2000. Proceedings of the 9th International Meshing Roundtable

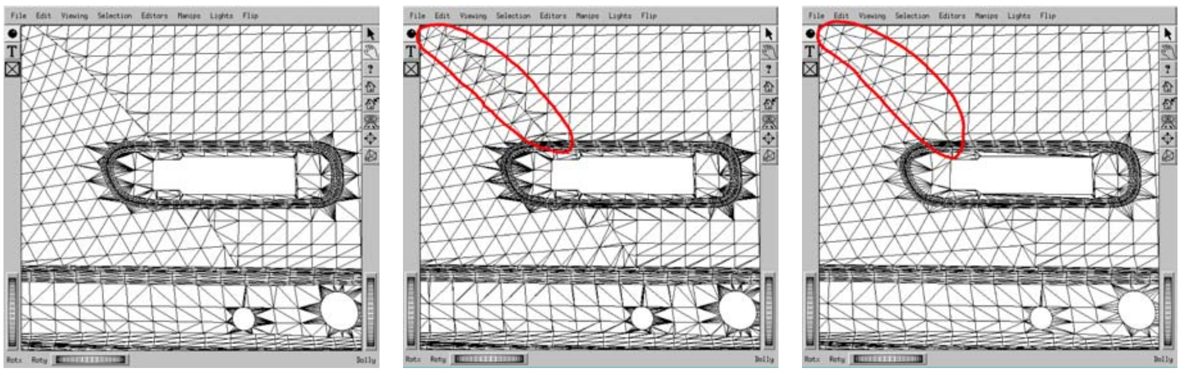
[14] Petersson N.A., Chand K.K. “Detecting Translation Errors in CAD Surfaces and Preparing Geometries for Mesh Generation.”, 2001. Proceedings of the 9th International Meshing Roundtable



(a) merged mesh



(b) boundary of merged mesh



(c) single faces, stitched and reduced along seams and boundaries

Figure 15: (a) shows a fender where its single patches were merged into a single mesh. (b) is the extracted boundary of the single mesh. (c) shows how the single patches were stitched, introducing many triangles, and a final reduction step along the face boundaries (pointed out red)

$\varepsilon = 0.05$mm maxlen=10mm	fender		hood		side frame	
	individual	merged	individual	merged	individual	merged
Our Approach	32149	33084	40287	41348	124474	126982
CATIA-OIV-Export	50471	58569	61828	70150	234765	278557
GLU-Tess.	159407	165245	197183	200581	713504	723874

Table 1: Three car parts tessellated with $\varepsilon = 0.05$ and a grid size of 10. Note, that the GLU-Tessellator can handle only ε.