Well behaved mesh generation for parameterized surfaces from IGES files

M. Randrianarivony G. Brunnett

Computer Graphics and Visualization
Fakultät für Informatik
Technische Universität Chemnitz
D-09107 Chemnitz
GERMANY

maharavo@informatik.tu-chemnitz.de

March 17, 2003

Abstract: This document gives a survey of parametric representations that are often used in computer aided geometric design. After recalling briefly the description of curves, we will detail the case of surfaces including B-spline, NURBS, tabulated cylinder, trimmed surface and surface of revolution. Secondly, we will provide an overview of CAD interfaces such as VDA-FS, SET, DXF, SAT, STEP, and most importantly IGES. The second part of the paper deals with applications in surface mesh generation. The parameter domain is triangularized using Delaunay-Voronoi meshing and the spatial mesh is refined according to the error provided by the parametric function itself. We need also to merge the resulting meshes so as to obtain a single mesh of the whole object surface. Some corroborative benchmarks from mechanically interesting objects will equally be provided.

Contents

1 Introduction

Some surfaces are already parametric in their nature. That is for instance the case for B-spline and NURBS surfaces. Some other important surfaces can have implicit representation. These surfaces include spheres, cylinders, cones and planes. Using rational functions, the latter ones have also simple parametric representations. In this document, we are exclusively interested in surfaces which are represented parametrically.
This document is structured in two principal parts. The first one consists of surface representation in CAD models and application in surface mesh generation is the second part. In section 2, we will have to review briefly the case of curves because curves are important for the definition of some surfaces. In section 3, we will give some detailed description of important surfaces that are often involved in CAD exchange files. These include B-spline surfaces, NURBS surfaces, trimmed surfaces, surfaces of revolution and tabulated cylinders. These surface types that will be described here are enough to describe most (if not all) mechanically interesting objects. Afterwards, we will consider some well known CAD interfaces. In this document we will focus significantly our attention on IGES CAD interface. In section 5, we will apply the information about CAD parametric surfaces in mesh generation. We will deal only with surface meshes. Extension to solid meshes might be a part of future works. Our mesh generation procedure consists mainly in having an initial mesh of the parameter domain which can be obtained by Delaunay triangulation. Next, this initial mesh is refined according to the error in each triangular element. After having the meshes for all patches, they can all be merged in order to have the final mesh of the whole surface. Care must of course be taken in order to avoid the presence of hanging nodes. In the last section, we will give some interesting benchmarks.

2 Concise survey of curves

Although we are mainly interested in surfaces, we need to review briefly the way curves are represented because they are of overwhelming importance in the definition of such surface types as tabulated cylinder and trimmed surface.

2.1 B-spline curves

A B-spline curve of order k is a piecewise polynomial curve. It can be expressed by the formula:

$$\mathbf{C}(t) := \sum_{i=0}^{n} \mathbf{d}_i N_i^k(t) \ , \tag{1}$$

where the points $\mathbf{d}_i$ $i = 0, ..., n$ are known as control points or de Boor points and N_i^k are the usual B-spline basis functions given by:

$$N_i^1(t) := \begin{cases} 1 & \text{if}\, t \in [\theta_i, \theta_{i+1}) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$N_i^k(t) = \frac{t - \theta_i}{\theta_{i+k-1} - \theta_i} N_i^{k-1}(t) + \frac{\theta_{i+k} - t}{\theta_{i+k} - \theta_{i+1}} N_{i+1}^{k-1}(t) \ . \tag{3}$$

For open curves, the functions N_i^k are defined on a knot sequence $\theta_0 = ... = \theta_{k-1}, \theta_k, ..., \theta_n, \theta_{n+1} = ... = \theta_{n+k}$. The parameter interval is therefore $I = [\theta_0, \theta_{n+k}]$. For closed curves, the knot sequence should have the next relationship:

$$\begin{cases} \theta_{n+1} &= \theta_n + \theta_0 \\ \theta_{n+2} &= \theta_{n+1} + (\theta_1 - \theta_0) \\ \quad \cdots & \\ \theta_{n+k} &= \theta_{n+k-1} + (\theta_{k-1} - \theta_{k-2}) \, . \end{cases} \tag{4}$$

So as to evaluate $\mathbf{C}$ at a point t, we do not use the formulas (1)(2) and (3). We use instead the de Boor algorithm ([2, 6, 9]):

step 0 : Determine r for which $t \in [\theta_r, \theta_{r+1})$.

step 1 : for $s = r - k + 1, \cdots, r$ define $\mathbf{d}_s^{(0)} := \mathbf{d}_s$.

step 2 : for $j = 1, \cdots, k - 1$

for $i = r - (k - j) + 1, \cdots, r$

$$\begin{aligned} \alpha_i^j &:= \frac{t - \theta_i}{\theta_{i+k-j} - \theta_i} \\ \mathbf{d}_i^{(j)} &:= (1 - \alpha_i^j)\mathbf{d}_{i-1}^{(j-1)} + \alpha_i^j \mathbf{d}_i^{(j-1)} \end{aligned}$$

step 3 : $\mathbf{C}(t) = \mathbf{d}_r^{(k-1)}$.

2.2 NURBS curves

A NURBS (nonuniform rational B-spline) curve is a generalization of a B-spline curve in which we use weights $w_i \in \mathbf{R}_+$:

$$\mathbf{C}(t) := \frac{\sum_{i=0}^{n} w_i \mathbf{d}_i N_i^k(t)}{\sum_{i=0}^{n} w_i N_i^k(t)} \ . \tag{5}$$

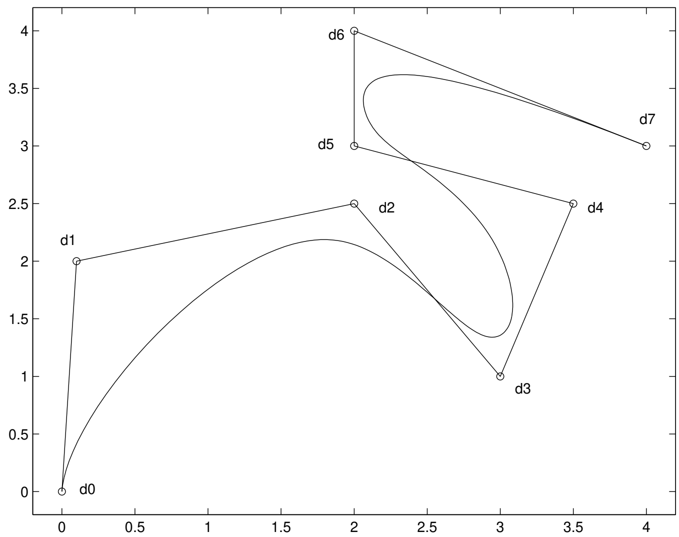

Figure 1: NURBS curve with $n = 7$, $k = 4$ and weights $(1.4, 0.5, 1.6, 1.8, 0.7, 1.9, 1.5, 0.9)$.

When all the weights are equal to unity, we have the B-spline curve in (1) because $\sum_{i=0}^{n} N_i^k = 1$. In Fig. 1 one can see a graphical illustration of NURBS curve together with its control polygon. Usefulness of NURBS in geometric modeling can be found in [16, 14, 15].

2.3 Lines

By a line we mean a straight bounded curve joining two distinct points $\mathbf{A}$ and $\mathbf{B}$ in the space. Although it is the most frequently used geometric curve, let us recall its most obvious parameterization:

$$\mathbf{C}(t) = \mathbf{A} + t(\mathbf{B} - \mathbf{A}), \tag{6}$$

and the parameter interval is $I = [0, 1]$.

2.4 Circular arcs

It is a portion of a circle which can be defined by giving its starting point $\mathbf{A} = (a_x, a_y, a_z)$, its terminate point $\mathbf{B} = (b_x, b_y, b_z)$ and the center $\mathbf{\Omega} = (\omega_x, \omega_y, \omega_z)$. A convention is made to choose from the two circular arcs joining $\mathbf{A}$ and $\mathbf{B}$. The counterclockwise direction is often appreciated. If we use the premise that the circular arc lies in a horizontal plane $z = z_0$, a possible parametric representation is:

$$
\mathbf{C}(t) = \begin{cases} \omega_x + r\cos(t) \\ \omega_y + r\cos(t) \\ z_0 \ , \end{cases} \tag{7}
$$

where r is the radius whose value can be deduced simply from the distance of $\mathbf{\Omega}$ and $\mathbf{B}$. The parameter interval is $I = [\theta_a, \theta_b]$ where θ_a and θ_b are the angular positions of the points $\mathbf{A}$ and $\mathbf{B}$ from the center Ω.
In the case of circular arc which does not lie in a horizontal plane, the rigid motion (8) can be used to transform it and then we can use (7) to have the parameterization of the transformed circular arc. That idea is often applied in IGES CAD format.

Remark 1 A rigid motion is given by the transformation:

$$
\tilde{\mathbf{x}} = \mathbf{M}\mathbf{x} + \mathbf{s}\,, \tag{8}
$$

where $\mathbf{M}$ is a (3×3) matrix defining a rotational operator and $\mathbf{s}$ a shift vector.

2.5 Composite curves

A composite curve $\mathbf{C}$ is a chaining of n curves $\mathbf{C}_i$, $i = 1, ..., n$. Suppose the parameter intervals of the n curves are respectively $I_i = [a_i, b_i]$. Let us introduce

$$
L_i := \sum_{j=1}^{i} (b_j - a_j).
$$

The parameter interval of the composite curve $\mathbf{C}$ is then $I = [0, L_n]$ and the parametric expression of composite curve is:

$$
\mathbf{C}(t) = \mathbf{C}_i(t - L_{i-1} + a_i)\ , \quad \text{if } t \in [L_{i-1}, L_i]\,.
$$

3 Parametric representation of surfaces in CAD

3.1 B-spline surfaces

A B-spline surface is a tensor product surface which generalizes (1). In other words, a B-spline surface can be defined by:

$$
\mathbf{S}(u,v) := \sum_{i=0}^{n_u} \sum_{j=0}^{n_v} \mathbf{d}_{i,j} N_i^{k_u}(u) N_j^{k_v}(v)\ . \tag{9}
$$

3.2 NURBS surfaces

A nonuniform rational B-spline (NURBS) surface with weights $w_{i,j} \in \mathbf{R}_+$ and control points $\mathbf{d}_{i,j} \in \mathbf{R}^3$ $i = 0, \cdots, n_u$ $j = 0, \cdots, n_v$ is given by:

$$\mathbf{S}(u, v) := \frac{\sum_{i=0}^{n_u} \sum_{j=0}^{n_v} w_{i,j} \mathbf{d}_{i,j} N_i^{k_u}(u) N_j^{k_v}(v)}{\sum_{i=0}^{n_u} \sum_{j=0}^{n_v} w_{i,j} N_i^{k_u}(u) N_j^{k_v}(v)} \,. \tag{10}$$

3.3 Surfaces of revolution

Let us take a surface of revolution which is obtained by turning the directrix about an axis or revolution. The starting angle of rotation is α_0 and the ending is α_1. Suppose that the directrix is expressed parametrically by $\mathbf{C}(t) = [C_x(t), C_y(t), C_z(t)]$ where $t \in [a, b]$. A way to define a parameterization of the surface of revolution is

$$\mathbf{S}(t, \alpha) = R_\alpha[\mathbf{C}(t)],$$

where R_α is a transformation which describes the rotation about the axis of revolution. R_α can be expressed easily in polar coordinates by a combination of rotations and shifts. The parameter domain is then $[a, b] \times [\alpha_0, \alpha_1]$.

3.4 Trimmed surfaces

Let us consider a surface whose parameterization $\mathbf{S}$ is defined on the parameter domain $\mathbf{I} = [a, b] \times [c, d]$. Let us take a parametric curve $\mathbf{C}$ whose parameter interval is $[e, f]$. Suppose that the image of $\mathbf{C}$ lies in $\mathbf{I}$ i.e.

$$\forall \ t \in [e, f], \qquad \mathbf{C}(t) \in \mathbf{I} \ .$$

Additionally, we assume that $\mathbf{C}$ does not have a double point, i.e. there do not exist two different parameters t_a, $t_b \in [e, f]$ for which $\mathbf{C}(t_a) = \mathbf{C}(t_b)$. A trimmed surface $\mathbf{G}$ whose base surface is $\mathbf{S}$ and whose boundary curve is defined by $\mathbf{C}$ is the set of those points $\mathbf{S}(u, v)$ such that (u, v) lies inside the image of $\mathbf{C}$ (See Fig. 2).

Remark 2 There are two ways to test whether a parameter (u, v) lies inside a curve $\mathbf{C}$ or not. First, one can approximate the image of the curve by a polygon and make a simple test if (u, v) lies inside the polygon. The second way is to evaluate the integral

$$\frac{1}{2\pi} \int_{t_1}^{t_2} \frac{C_2'(t)[C_1(t) - u] - C_1'(t)[C_2(t) - v]}{[C_1(t) - u]^2 + [C_2(t) - v]^2} dt \tag{11}$$

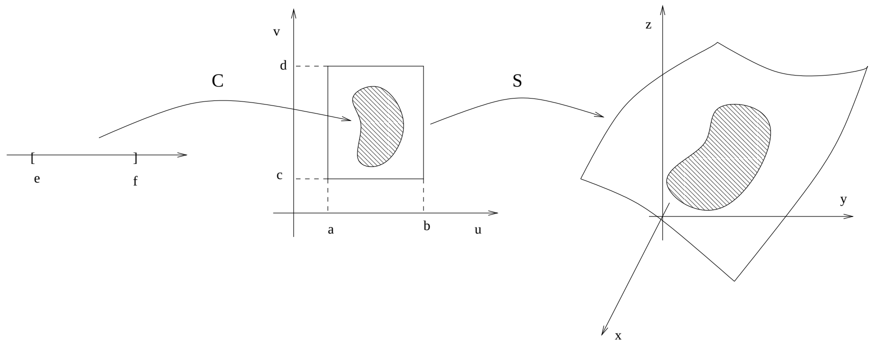

Figure 2: Trimmed surface

which equals to 1 if (u, v) lies inside the curve $\mathbf{C} = [C_1, C_2]$ and it takes a zero value otherwise. Let us note that (11) is a singular integral and if the boundary is easily approximated by a polygon then the first method is more advisable.

Remark 3 It is also possible to consider trimmed surfaces with inner boundaries (see Fig. 3). The inner curves should not either have double points. Moreover, one must consider the assumption that the inner curves do not intersect with one another.

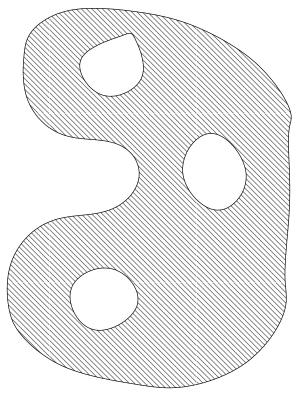

Figure 3: Trimmed surface with four inner boundaries

3.5 Tabulated cylinders

A tabulated cylinder is the surface which is drawn when we move a line along a curve which is called directrix. One of the endpoints of the line and the starting point of the directrix are supposed to coincide. We will denote it by $\Omega = (\omega_x, \omega_y, \omega_z)$. The usual cylinder is a particular case in which the directrix is a circle. Let us suppose that the directrix has $\mathbf{C}(t) = [C_x(t), C_y(t), C_z(t)]$, $t \in [a, b]$ as a parametric expression. A possible parameterization of a tabulated cylinder is therefore:

$$\mathbf{S}(u, v) = \begin{cases} C_x(a + u(b-a)) + v(w_x - C_x(0)) \\ C_y(a + u(b-a)) + v(w_y - C_y(0)) \\ C_z(a + u(b-a)) + v(w_z - C_z(0)) \end{cases} \tag{12}$$

The parameter domain is $[0, 1]^2$.

4 CAD interfaces

4.1 Overview of standard CAD formats

4.1.1 DXF

DXF stands for **D**rawing e**X**change **F**ormat ([1]). This format was used originally to store 3D models created by the CAD-system autoCAD ([3, 11]). DXF format represents information in tagged data mode. That is, an integer preceeds every data piece. There are nowadays interesting tools for loading a DXF file and use the data from it in OpenGL applications. DXF files are available in both ASCII and binary formats.

4.1.2 VDA-FS

VDA-FS or **V**erband **d**er **A**utomobilindustrie-**F**lächen **S**chnittstelle is a file format that has its origin from the German car manufacturers association VDA. VDA-FS format is well known in its capability to manipulate free form 3D surfaces which are very frequently involved in car designs. A VDA-FS file can be recognized by its suffix ".vda".

4.1.3 SET

The file format SET (**S**tandard d'**É**change et de **T**ransfert) started its development in 1983 and it became a French standard for CAD data exchange in 1985. This format has originated from the aircraft industry Airbus.

4.1.4 IGES

IGES stands for **I**nitial **G**raphics **E**xchange **S**pecification and it is the US standard for CAD data exchange (see [19]). This format is more than just a storage of geometric entities like curves and surfaces. It can also contain graphical properties like colors in RGB mode (which can be easily converted in other color modes like CMY or HLS modes) and material properties. Another variant of IGES is described in [5] where the author explains the NINO format. That is an IGES file which contains exclusively NURBS entities. Since thid is a subset of IGES file, every application which accepts an IGES file will also accpet a NINO file. Due to the fact that the majority of (if not all) curve and surface entities in IGES can be expressed in terms of NURBS curves and surfaces (see [16]), a convertion program from IGES to NINO is depicted in [5]. A convertion from NINO to IGES is certainly of no use because a NINO file is already an IGES file. Since all the numerical tests done in this paper have been performed with IGES files, we are going to see more description of this format in the next section.

4.1.5 STEP

The International Organization for Standardization (ISO) has introduced the STEP (**ST**andard of the **E**xchange of **P**roduct model data) format in order to release a CAD exchange model which should be accepted as standard worldwide. STEP is nowadays the most modern CAD interface.

4.1.6 SAT and SAB

SAT (**S**tandard **A**CIS **T**ext) and SAB (**S**tandard **A**CIS **B**inary) are file formats that are used for storing and transferring data in ACIS (see [4]) which is a very flexible C++ and SCHEME library produced by Spatial Technologies for geometry manipulation purposes. These two formats are also known as save files. Although SAT files are introduced by ACIS, they can be loaded by some other CAD-systems.

4.2 Brief IGES description

Although STEP proposes a very promising and modern international standardization, IGES is apparently the most frequently used CAD interface for now. That is probably due to its chronological introduction. IGES was interestingly developed in 1979 by the US National Bureau of Standards whereas the first release of STEP took only place in 1994. Between the year frame 1979-1994, a lot of applications have been already done in many domains which need CAD data such as finite element method or mesh generation. On the other hand, in order for a standard to gain popularity, a minimum amount of time is required. In this section, we give further description of IGES file format. An IGES file is structured in five main sections: start section, global section, directory entry, parameter data and terminate section.

- The start section is a human-readable section in which you can write anything like the directory location of the file. It should have at least one line.

- In the global section, we find general information such as how to read the current file, where, when and by whom was the file generated. It can also specify the parameter delimiter as well as the record delimiter.

- The directory entry gives in general an overview of the different components of the IGES file and it points as well to a record in the Parameter data which contains the complete information about all parameters. For instance, in the directory entry we can see that we have to deal with NURBS surfaces. The information like control points, knot sequence and weights cannot be found in the Directory entry. Every entity has its own identification number which can range from 100 to as many as 514. In Table 1 we summarize the entity number of a few important geometries.

Entity	ID number
Circular arc	100
Polynomial/rational B-spline curve	126
Composite curve	102
Line	110
Surface of revolution	120
Tabulated cylinder	122
Polynomial/rational B-spline surface	128
Trimmed parametric surface	144

Table 1: Interesting curve and surface entities

- It is in the parameter data that we can find the values of all parameters. This section varies from one entity to another but it has in general the following forms, if we suppose that the parameter delimiter is a comma (,) and record delimiter is a semicolon (;).

  entity number,parameter1,parameter2,...,parameterN;

  After the record delimiter, any comment can be added.

- The terminate section consists only of a single line. Columns 33-72 are not used in this section. We find here four fields corresponding to the former four sections.

Field	Columns	Section
1	1-8	Start
2	9-16	Global
3	17-24	Directory entry
4	25-32	Parameter data

Table 2: four fields in terminate section

5 Application in grid generation

In this section we want to solve a problem of generating a mesh on an object whose boundary surface is constituted of patches given in parametric representation. Such a problem can be found in [17, 18], but the author has only used NURBS surfaces in those papers.

5.1 Problem formulation and definitions

We suppose that we have parametric functions

$$\mathbf{S}_i : D_i \to \mathbf{R}^3 \quad , \tag{13}$$

where D_i is supposed to be a subset of $\mathbf{R}^2$. The explicit expression of $\mathbf{S}_i$ is supposed to be known exactly. That have been extracted from an IGES file in our case. Our goal is to triangularize the image domaine of $\mathbf{S}_i(D_i)$ by a mesh $\mathcal{T}_i$. We merge then the resulting meshes $\mathcal{T}_i$ in order to have an overall mesh $\mathcal{T}$. For the sake of simplicity of notation, we will just drop the indices when no confusion can happen.

Definition 1 Let $T \subset D$ be a triangle whose vertices are a, b, c. The respective images by $\mathbf{S}$ of the latter are $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$. We define the polygonal map $\mathbf{M}$ on T as the function which transforms T into the 3D triangle having $\mathbf{A}, \mathbf{B}, \mathbf{C}$ as vertices. More precisely, let $(u, v) \in T$ be given. Consider the barycentric coordinates $\lambda_a, \lambda_b, \lambda_c$ of (u, v) with respect to T. The image of (u, v) by $\mathbf{M}$ is then:

$$\mathbf{M}(u, v) := \lambda_a \mathbf{A} + \lambda_b \mathbf{B} + \lambda_c \mathbf{C} \ .$$

Definition 2 Let T be a triangle in the parameter domain D. The error $\varepsilon(T)$ within T with respect to $\mathbf{S}$ is defined to be the energy norm of the difference of S and the polygonal map $\mathbf{M}$ divided by the area μ of $\mathbf{M}(T)$. That is:

$$\varepsilon(T)^2 := \frac{1}{\mu^2} \int_T [\mathbf{S}(u,v) - \mathbf{M}(u,v)]^2 du\, dv \ . \tag{14}$$

Definition 3 A mesh $\mathcal{T}$ is a set of topologically open triangles T_i $i = 1, ..., n$ for which we have:

1. The elements of $\mathcal{T}$ are disjoint:

$$T_i \cap T_j = \emptyset \qquad \forall\, i \neq j\, .$$

2. Every edge of any element $T_i \in \mathcal{T}$ is either a boundary edge or an edge of another element $T_j \in \mathcal{T}$.

5.2 Triangulation

The algorithm that we adopt here is to take an initial mesh $\mathcal{T}_0$ of the domain D. Afterwards, we refine all elements T of $\mathcal{T}_0$ for which $\varepsilon(T)$ is greater than some prescribed accuracy ε_0. That process can be repeated recursively. The process of creating an initial mesh $\mathcal{T}_0$ of the domain D is done in two steps

- We approximate the outer boundary as well as the inner boundaries (if they exist) of D by polygons. That can be done arbitrarily accurate because in practice, the explicit expressions of the boundaries are given in the IGES file. The key issue is here to choose as few edges as possible in the polygonal approximation while achieving the highest accuracy. That means at positions where the boundary of D varies considerably, we need edges which are short. Whereas at those which are almost linear, we can have long edges. In our program, that process has also been done iteratively by computing at each step the difference between the boundary curve and the polygon and inserting new nodes when the error is too large.

- We triangulate the resulting 2D domain having polygonal boundaries. Two main algorithms can be used for that purpose: advancing front technique (see [7, 13]) or Delaunay-Voronoi triangulation (see [8]). For the subsequent numerical tests, Delaunay-Voronoi triangulation has been used.

5.3 Refinement procedure

We want to describe briefly a method that can be used to achieve a refinement strategy. An element T which has an error $\varepsilon(T)$ too large will subdivided into four triangles (see Fig. 4). That process can lead to the existence of hanging nodes which are not conform to the definition of a mesh.

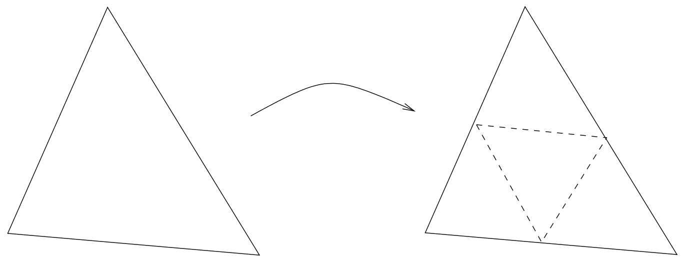

Figure 4: Subdivision of a triangle into 4 sub-triangles

In order to remove the resulting hanging nodes, two cases should be considered. First, if only one adjacent triangle T' to a triangle T has been refined (see Fig. 5), then we subdivide the element T by joining the hanging node with the opposite vertex.

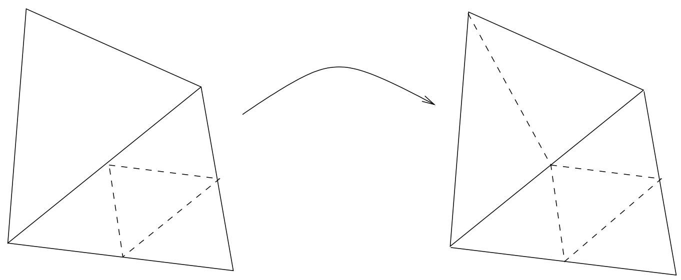

Figure 5: Hanging node removal

On the other hand, if more than one neighboring elements have undergone a refinement process like in Fig. 6, then the element will also be refined into four subelements.

5.4 Mesh merging

In practice, the 3D surfaces which are the images of D_i by $\mathbf{S}_i$ in formula (13) describe the boundary faces of an object. Therefore, some of them are certainly adjacent. In this section, we suppose that we have meshes $\mathcal{T}_i$ which

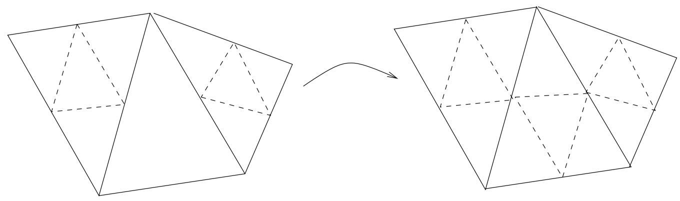

Figure 6: Hanging node removal

are triangulation of the surface patches and we want to have a single mesh $\mathcal{T}$. For two non-adjacent meshes $\mathcal{T}_i$ and $\mathcal{T}_j$, that does not pose any problem because none of their boundary elements will interfere and we need therefore to remove no hanging node. The only problem that we must solve is the case of two adjacent meshes. For that latter case, we run through all boundary nodes of the mesh $\mathcal{T}_i$ and we determine whether they should belong to the mesh $\mathcal{T}_j$ as well. If so, then we determine to which boundary element T of $\mathcal{T}_j$ should the node belong. Afterwards we insert a new node inside $T \in \mathcal{T}_j$ and subdivide T in order to avoid hanging nodes. Finally, the resulting mesh $\mathcal{T}$ is simply the reunion of all meshes $\mathcal{T}_i$.

6 Benchmarks

The program that has been used to obtain the following results was written in C and it uses openGL ([12]) and GLUT ([10]) for the final graphical display of the mesh. The program can load automatically IGES files which were in our numerical experiments all generated with the help of pro Engineer ([20]).

6.1 Mechanical part

Our first benchmark consists of a simple mechanical part. It is displayed graphically in Figure 7. The CAD representation in the IGES file of this object has 30 surface patches. The mesh is constituted of 44,794 triangular elements. In our numerical results, we investigate also the aspect ratio which is by the way the ratio between the longest edge and the shortest edge of an element. The inverse of the aspect ratio of the elements of the mesh is displayed in the bar chart of Figure 8. The majority of the elements have small aspect ratio. In fact, 73.86 percent of the elements have an aspect ratio between 1 and 2.51.

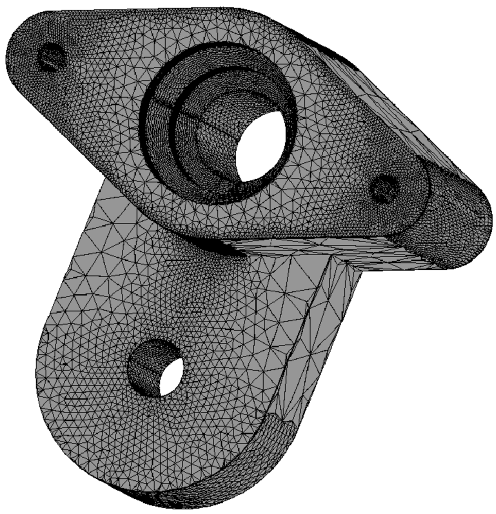

Figure 7: Mechanical part

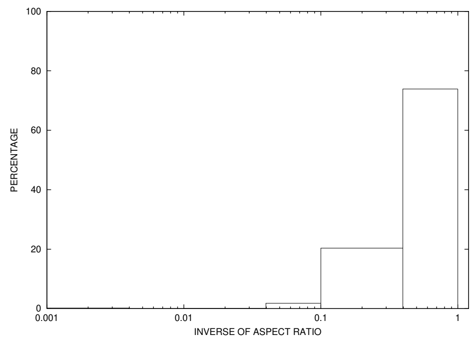

Figure 8: Result for mechanical part

6.2 Faucet

The second object is a faucet whose bottom part is shown in Fig. 9. The IGES file storing this object has 29 surface patches which are either NURBS surfaces or surfaces of revolution. The mesh has 87,488 triangular elements whose aspect ratio are shown in Fig. 10. The histogram shows that the elements have mainly low aspect ratio. 67.78 percent of the elements have aspect ratio between 1.0 and 2.51. Elements having aspect ratio greater than 10 are not more than four percent.

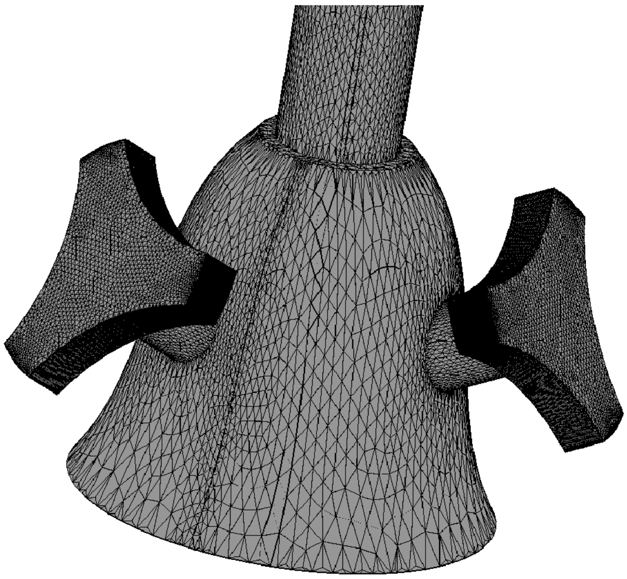

Figure 9: Faucet

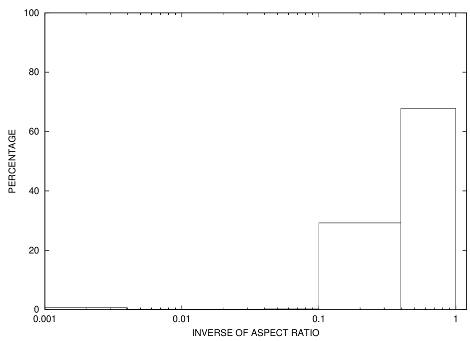

Figure 10: Result for faucet

6.3 Teacup

Our next object is a simple teacup which consists of 23 surface patches. The resulting mesh which is shown in Fig. 11 has 40,646 triangular elements. If the teacup did not have a handle, then the triangulation of the object would be a cakewalk. We would have only to discretize the directrix of the surface of revolution and rotate the results around the axis of revolution. The presence of the handle of the teacup makes the meshing of this object not obvious because NURBS curves are involved in the handle. Furthermore, we have to deal with the hanging node removal between the handle and the main part of the teacup. The numerical outcomes about the aspect ratio of the elements are depicted in Fig. 12. The results show that the mesh can still be categorized as having low aspect ratio.

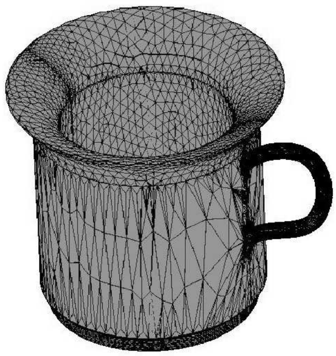

Figure 11: Teacup

6.4 Mill cutters

We present in this benchmark two mill cutters which are shown in Fig. 13 and Fig. 14. The first one has oblique teeth and the second one has straight teeth. These objects demonstrate a situation in which we have simple objects with a lot of surface patches. They consist in fact of 38 and 39 surface patches. One tooth of the straight mill cutter has been generated with Pro/Engineer ([20]) as an extrude object in which the section is a combination of NURBS

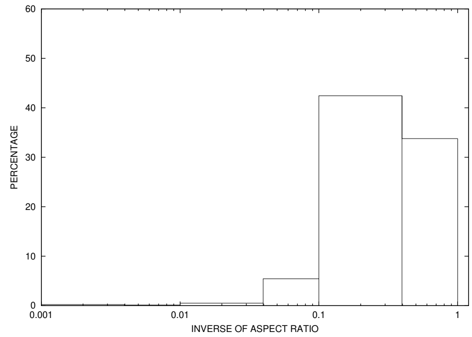

Figure 12: Result for teacup

curves and circular arcs. As for the oblique mill cutter, one tooth is a variable section sweep object with helical path as X-trajectory. The resulting meshes have 132,084 and 92,900 triangular elements. The distribution of the aspect ratio of the elements can be found in figure 15 and 16. Those mill cutters show tangibly that the treatment of trimmed surfaces are unavoidable if one has to deal with practically interesting surfaces. The right section of those objects are in fact trimmed surfaces having both inner as well as outer boundary curves.

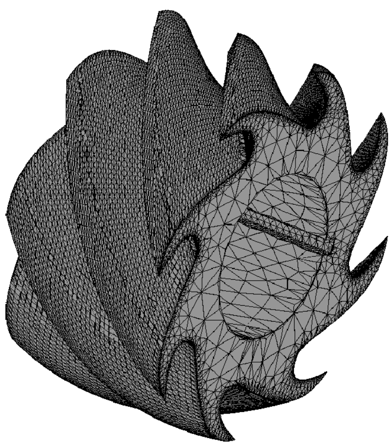

Figure 13: Mill cutter with oblique teeth

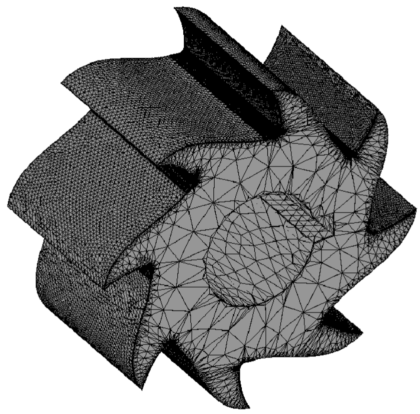

Figure 14: Mill cutter with straight teeth

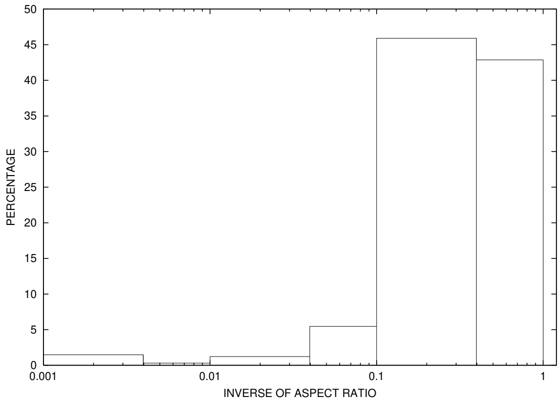

Figure 15: Result for oblique mill cutter

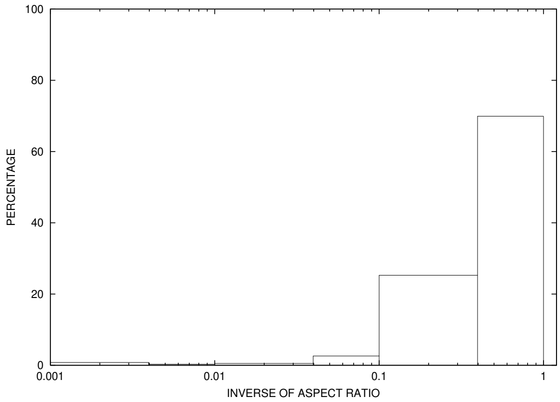

Figure 16: Result for straight mill cutter

6.5 Tilt rotor

Our last example is a tilt rotor having three airfoils. It is drawn in Fig. 17. The surface is composed of 25 patches. The airfoils were created with Pro Engineer as blend objects whose end sections contain two NURBS curves. The surface mesh of this tilt rotor is constituted of 47,344 triangular elements. The numerical results in the bar chart of Fig. 18 show equally that most elements have low aspect ratio.

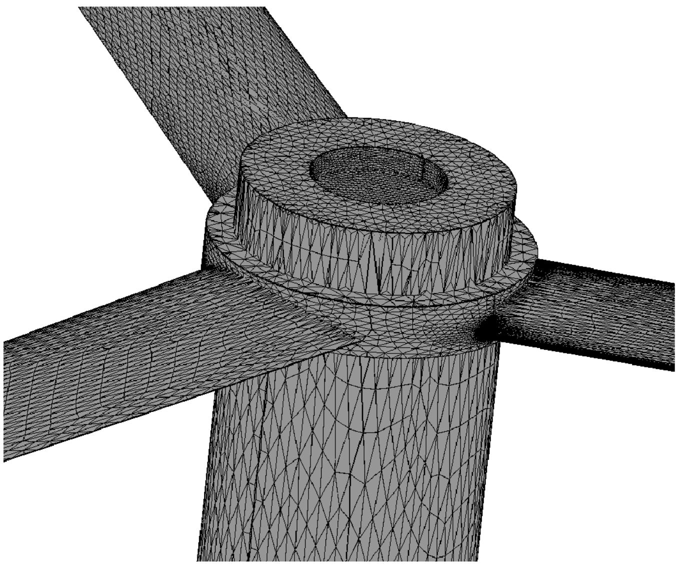

Figure 17: Tiltrotor

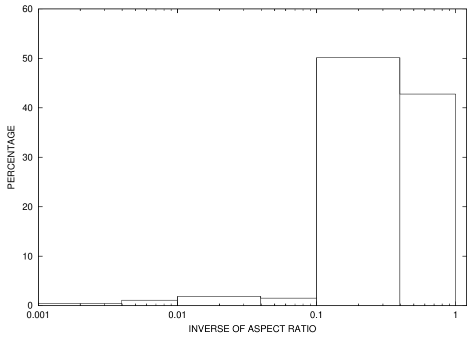

Figure 18: Result for tilt rotor

References

[1] Autodesk, DXF Reference Guide, www.autodesk.com/techpubs/autocad/dxf, (2002).

[2] C. de Boor, A practical guide to splines, Springer (New York, 1978).

[3] M. Bousquet and J. Hester, 3D-Konstruktion und Präsentation mit AutoCAD, IWT-Verl. (München, 1993).

[4] J. Corney, 3D modeling with ACIS kernel and toolkit, John Wiley & sons (Chichester, 1997).

[5] A. Evans and D. Miller, NASA IGES and NASA-IGES-NURBS-Only Standard, Handbook of grid generation, CRC Press, Chapter 31 (1999).

[6] G. Farin, Curves and surfaces for computer aided geometric design, Academic Press, 2nd edition (Boston, 1990).

[7] P. George, Automatic mesh generation, John Wiley & sons (Chichester, 1991).

[8] P. George and H. Borouchaki, Delaunay triangulation and meshing, Hermes (Paris, 1998).

[9] J. Hoschek and D. Lasser, Grundlagen der geometrischen Datenverarbeitung, Teubner (Stuttgart, 1989).

[10] M. Kilgard, The OpenGL Utility Toolkit (GLUT) Programming Interface API Version 3, Silicon Graphics, Inc. (1994)

[11] R. Knight and W. Valaski, AutoCAD-Referenz, IWT-Verl., (München, 1993),

[12] J. Neider, T. Davis and M. Woo, OpenGL programming guide, Addison-Wesley publishing company (Reading, 1994).

[13] J. Peraire, J. Peiro and K. Morgan, Advancing front grid generation, Handbook of grid generation, CRC Press, Chapter 17 (1999).

[14] L. Piegl and W. Tiller, A menagerie of rational B-spline circles, Computer Graphics & Applications, vol. 9, no. 5, pp. 48–56 (1989).

[15] L. Piegl, Infinite control points-a method for representing surfaces of revolution using boundary data, Computer Graphics & Application, vol. 7, no. 3, pp. 45-55 (1987).

[16] L. Piegl, On NURBS: A Survey, Computer Graphics & Application, vol. 11, no 1, pp. 55-71 (1991).

[17] J. Samareh-Abolhassani, Unstructured Grids On Nurbs Surfaces, AIAA-93-3454-CP, pp.435-445 (1993).

[18] J. Samareh-Abolhassani, Triangulation of NURBS surfaces, in Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, pp. 377-388, Melksham, Wiltshire (1994).

[19] Initial Graphics Exchange Specification IGES 5.3, U.S. Product Data Association, (South Carolina, 1996).

[20] M. Vogel and P. Bunte, Pro/ENGINEER und Pro/MECHANICA, Carl Hanser Verlag, (München, 2001).