EUROGRAPHICS '89
W. Hansmann, F.R.A. Hopgood and W. Strasser (Editors)
Elsevier Science Publishers B.V. (North-Holland)

Deformation of Solids with Trivariate B-Splines

Josef GRIESSMAIR, Werner PURGATHOFER

Institut für Praktische Informatik
Techn. Univ. Vienna, A-1040, Karlsplatz 13/180, Österreich.

Abstract

Solid geometric models can be deformed to free-form solids by the use of trivariate B-Splines. This paper describes the problems of implementing such transformations for shaded rendering.
The surfaces are subdivided into triangles adaptively so that the error in image space is limited. This adaptive triangulation ensures a smooth appearance of the resulting pictures.

Key words: B-Splines, Solid Modelling, Triangulation, Ray Tracing, Deformation.

1.Introduction

Deformation of regular primitives to free form objects is a powerful tool to describe a wide variety of objects. Unfortunately, the rendering process (e.g. ray tracing) of deformed surfaces and deformed solids is not trivial. Many authors have addressed problems of this kind and have proposed mathematical solutions to different cases. Jet, no one really describes the algorithm that was used to render such objects for shaded display, except for some special cases [1] [2] [9] [10] [12] [14] [19] [20].
This paper deals with arbitrary b-rep solids which are deformed by trivariate B-Splines. It describes a concrete implementation of the deforming process, and gives practical solutions to the arising problems. The result is a boundary representation of the deformed object with a given accuracy. In chapter 2 an unusual notation for B-Splines is introduced, chapter 3 describes the theory of trivariate B-Spline deformation. Chapters 4 and 5 give the details of the algorithm, chapter 6 describes the rendering process.

2. B-Spline Notation

The following notation for B-Splines is introduced on the basis of a curve in R^2, which is a univariate parameter curve. The formal extension to multivariate B-Splines can then be done easily.

A B-Spline curve is a parameter curve:

$$B_k(t) = \sum_{i=1}^{n} N^k_{i,k}(t)\, P_i \quad \text{with } t \in [0, n\text{-}k]$$

where $P_i \in R^2$ are the "control points" of the curve and where $N^k_{i,k}(t)$ are the "weighting functions" of these control points.

$N^k_{i,k}(t)$ are defined as follows:

(i) the knot vector $\tau = (t_1, t_2, .. , t_{n+k+1})$ is defined by:

$$t_i = \begin{cases} 0 & \text{for } i = 1, .., k \\ i - (k+1) & \text{for } i = k+1, .., n+1 \\ n - k & \text{for } i = n+2, .., n+k+1 \end{cases}$$

(ii)

$$N^k_{i,0}(t) = \begin{cases} 1 & \text{for } (t_i \le t < t_{i+1} \text{ and } 0 \le t < n-k) \text{ or } (i = n \text{ and } t = n-k) \\ 0 & \text{otherwise} \end{cases}$$

(iii)

$$N^k_{i,m}(t) = \begin{cases} \dfrac{t - t_i}{t_{i+m} - t_i} N^k_{i,m-1}(t) + \dfrac{t_{i+m+1} - t}{t_{i+m+1} - t_{i+1}} N^k_{i+1,m-1}(t) \\ \quad \text{for } t_i \neq t_{i+m+1} \\ \quad \text{(if a N = 0, then the value of the containing expression is zero)} \\ 0 \quad \text{for } t_i = t_{i+m+1} \end{cases}$$

The notation is slightly different from the usual notation [15] [17] [8].
There the parts (ii) and (iii) are only valid for $t \in [0, n-k)$; $N^k_{n,k}(t) = 1$ and $N^k_{i,k}(t) = 0$ $(1 \le i \le k-1)$ are defined separately for $t = n-k$. The advantage of our notation is that the whole curve can be calculated at once and no special cases are necessary. This is extremely important for multidimensional B-Splines.

Since k is constant for a given B-Spline, we will denote $N^k_{i,m}$ by $N_{i,m}$ in the rest of the paper. But note that the $N_{i,m}$ are different for different k.

The extension for trivariate B-Splines is simple:

$$B_{k_x k_y k_z}(s,t,u) = \sum_{h=1}^{m} N_{h,k_x}(s) \left(\sum_{i=1}^{n} N_{i,k_y}(t) \left(\sum_{j=1}^{p} N_{j,k_z}(u)\, P_{h,i,j} \right) \right)$$

where $s \in [0, m-k_x]$, $t \in [0, n-k_y]$, $u \in [0, p-k_z]$ and $P_{h,i,j} \in R^3$.

3. Deformation of a Box and of an Enclosed Solid with Trivariate B-Splines

This chapter gives a short comprehension of the deformation of a solid enclosed in a box as described in [19].

3.1 Definition of a Box ($P^{1,1,1}$, S, T, U) in R^3

Let E_x, E_y, E_z be the unit vectors with $E_x = (1\ 0\ 0)^T$, $E_y = (0\ 1\ 0)^T$, $E_z = (0\ 0\ 1)^T$, and $S = (S_x S_y S_z)^T = aE_x$, $T = (T_x T_y T_z)^T = bE_y$, $U = (U_x U_y U_z)^T = cE_z$ with $a, b, c \in R^+$, and $P^{1,1,1} \in R^3$.

The vectors S, T, U placed on the top of $P^{1,1,1}$ define the box $Q = (P^{1,1,1}, S, T, U)$ (fig.1).

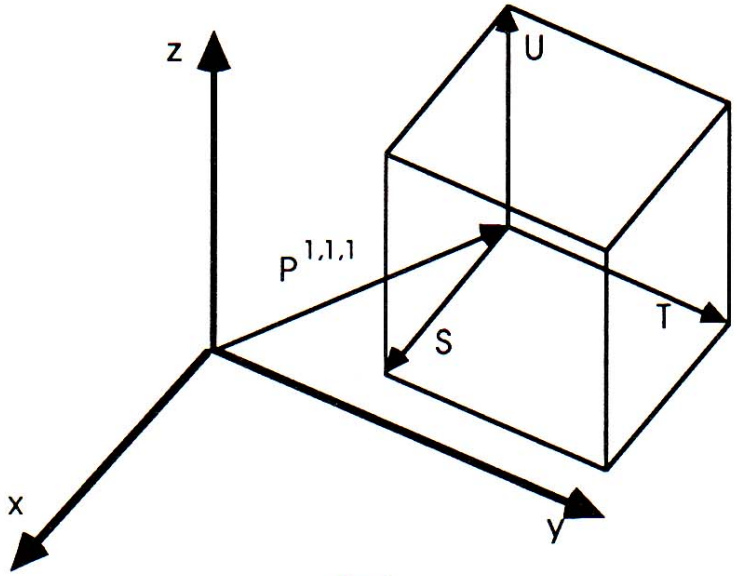

fig.1

Definition: A point $P = (P_x, P_y, P_z) \in R^3$ lies within the box

$Q = (P^{1,1,1}, S, T, U)$, iff

$$P_x^{1,1,1} \le P_x \le P_x^{1,1,1} + S_x \text{ and}$$
$$P_y^{1,1,1} \le P_y \le P_y^{1,1,1} + T_y \text{ and}$$
$$P_z^{1,1,1} \le P_z \le P_z^{1,1,1} + U_z.$$

3.2 Grid of Control Points

The box Q is filled with a grid of control points:

$$P^{h,i,j} = P^{1,1,1} + \frac{h-1}{m-1} S + \frac{i-1}{n-1} T + \frac{j-1}{p-1} U$$

with $m, n, p \in N \setminus \{0, 1\}$, $\quad 1 \le h \le m,\ 1 \le i \le n,\ 1 \le j \le p.$

The grid is defined by m planes that are parallel with the UT-plane, n planes that are parallel with the SU-plane and p planes that are parallel with the ST-plane.

3.3 Local Box Coordinates of a Point

Let k_x, k_y, k_z be the orders of three B-Splines with $k_x < m$, $k_y < n$, $k_z < p$.

Definition: The local box coordinates (s, t, u) of a point $P \varepsilon Q$ are given by:

$$s = ((P_x - P_x^{1,1,1}) / a) (m - k_x),$$
$$t = ((P_y - P_y^{1,1,1}) / b) (n - k_y),$$
$$u = ((P_z - P_z^{1,1,1}) / c) (p - k_z).$$

From this definition can be immediately concluded:

$$s \varepsilon [0, m - k_x], t \varepsilon [0, n - k_y], u \varepsilon [0, p - k_z].$$

3.4 Deformation of a Box

The deformation of a box is performed in the following way: First the grid points $P^{h,i,j}$ are displaced arbitrarily to form the control points $P^{*h,i,j}$; then the local box coordinates of every point $P(s, t, u) \varepsilon Q$ are mapped to $P^*(s, t, u)$ with the following function:

$$(1)\quad P^*(s,t,u) = \sum_{h=1}^{m} N_{h,k_x}(s) \left(\sum_{i=1}^{n} N_{i,k_y}(t) \left(\sum_{j=1}^{p} N_{j,k_z}(u)\, P^{*h,i,j} \right) \right)$$

The result is a B-Spline-Box.

Remarks

a) The B-Spline-Box lies within the convex hull of the $P^{*h,i,j}$, because the points P^* are a convex combination of the control points $P^{*h,i,j}$.
b) The bounding surfaces of the box are transformed to B-Spline surfaces.
c) The computational expense to evaluate function (1) is of order $O(k_x k_y k_z)$ and therefore independent of the number of control points.

3.5 Deformation of an Object inside a Box

An object K of arbitrary shape can be deformed in the following way:
Let $Q = (P^{1,1,1}, S, T, U)$ be a box and K an object lying inside Q. A deformation of Q causes a transformation of all interior points of Q and therefore of all points of K.

There are many different object description schemes in computer graphics. They include b-reps, CSG trees, sweeps, mathematical equations or simple rules (as for fractals). The theory of B-Spline deformation of objects is independent of these schemes.
Nevertheless, for a concrete implementation one has to concentrate on one representation. The most general description structure would certainly have been the use of CSG: Unfortunately it turns out that a ray-object intersection requires the solution of an equation of at least degree 6 [19] and so it is difficult to use deformed CSG-objects for ray-tracing. We therefore chose the b-rep method, since all other concepts are not general enough. Also, it is possible to transform different representation schemes to a b-rep approximation (see [13] for CSG objects). The result of deforming a b-rep object with 3D-B-Splines will again be of b-rep kind.

The outlines of the algorithm will be as follows. First all polygons are subdivided into triangles. These triangles are then deformed resulting in non planar triangular patches. These will then be approximated by plane triangles to a given level of accuracy. The resulting b-rep objects can be rendered with standard methods.

4. Deformation of a B-rep Object

Let K be a b-rep object bounded by convex polygons. Then every polygon of K can be subdivided into triangles, i.e. the surface of K is triangulated.

This triangulation is arbitrary. E.g., an efficient solution of the problem is the following algorithm:

Let $ax + by + cz + d = 0$ be the equation of the plane σ of the polygon, then $n = (a\ b\ c)^T$ is a perpendicular vector on σ.

The polygon is projected
onto the xy-plane, if $c \geq a$ and $c \geq b$,
onto the xz-plane, if $b \geq a$ and $b \geq c$,
onto the yz-plane, if $a \geq b$ and $a \geq c$.

In this way the polygon is projected onto that plane which produces the largest mapping of it.

After this projection the polygon is surrounded by an orthonormal 2D-box (rectangle). The center of this rectangle always lies inside the polygon, since the polygon is convex. This centre is transformed back into σ and connected with all vertices of the polygon. The computational effort is not much more than the intersection of a straight line with the plane σ, caused by projecting the center back into σ (fig.2). Alternately, the mean value of the vertices could be used as centre.

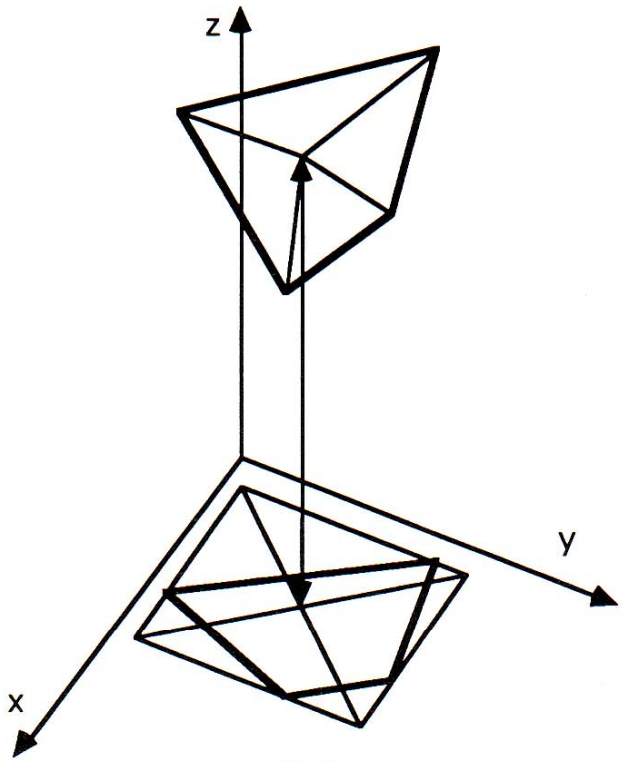

fig.2

Next, the object, which is now made up of triangles, is surrounded by an orthonormal 3D-box as described in section 3.1. This box will be deformed as described in the previous chapter and therewith will cause the deformation of our object.
By applying (1) to the triangles, deformed triangular patches are produced (continuous parameter surfaces). The aim is to find a representation of the deformed object, that is made up of plane triangles. The following concept allows an approximation of continuous parameter surfaces with plane triangles.

5. Adaptive Approximation of a Continuous Parameter Surface with Triangles

Let F be a continuous mapping from R^2 into R^3:

$$(u, v) \longmapsto F(u, v) = (F_x(u, v)\ F_y(u, v)\ F_z(u, v))^T,$$

where we can assume u,v ε [0, 1] without restricting generality. All other cases can be produced by applying a linear transformation.

F(u, v) is the surface patch in R^3 that shall be approximated with triangles.

A trivial solution would be a regular triangulation of the parameter space. The triangle vertices are transformed by F(u, v) and connected by straight lines. In this way triangles are created that are an approximation of the surface. It is obvious that the subdivision degree corresponds tightly to the quality of this approximation, but also to computational expense in terms of memory and time. (fig.3).

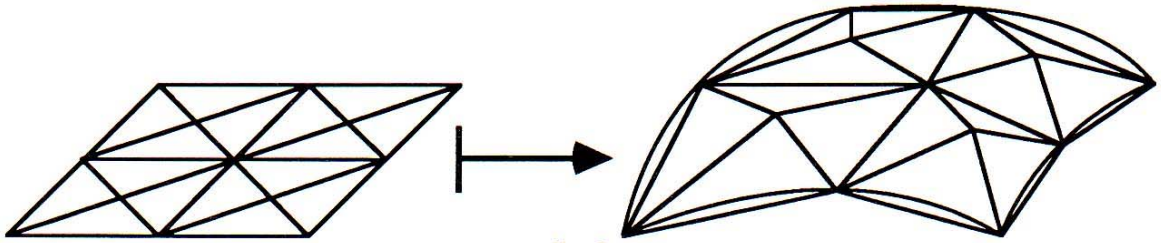

fig.3

5.1 Triangulation Algorithm

Beside computational expense, there is another reason for a better subdivision scheme. The regular triangulation does not take into account the different complexities at different areas of the surface at all. Large, almost plane areas are subdivided to a much too high degree, whereas parts of strong tension are not handled in enough detail.
The following algorithm takes these facts into account and leeds to a very fine subdivision where it is needed. First of all, a basic triangulation in parameter space is necessary, as in fig.4. Of course, any other basic triangulation would do as well.

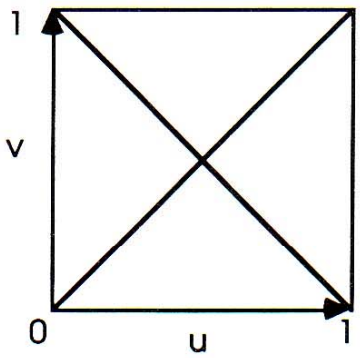

fig.4

Now the vertices of the parameter space triangles are mapped into object space by F(u, v). Every edge in parameter space has its partner in object space if the points are connected in the same way. In the general case, the mapping of an edge will not be equivalent with its partner (fig.5).

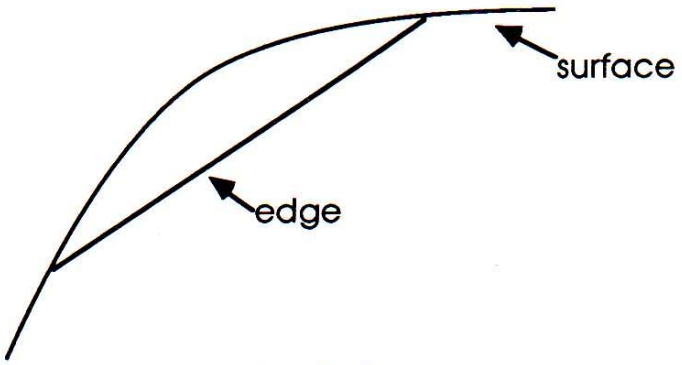

fig.5

The difference between the mapping of an edge and the straight connection of the mapping of its corners will be the basis for a measure to be defined, which classifies how good an edge is. Bad edges will be replaced by four others in parameter space so that the resulting edges will be "better". A sequence in which these edges shall be eliminated will be defined, so that there is always a "worst" edge.

To describe one iteration step, during which the worst edge is replaced by four others, the following notation is used (see also fig.6):

in object space:

k^* ... worst edge

δ_1, δ_2 ... triangles to which k^* belongs

A^*, B^* ... end points (corners) of k^*

C^*, D^* ... third corners of δ_1, δ_2

P ... midpoint of k^*

n_1, n_2 ... normal vectors on δ_1, δ_2

in parameter space:

A, B, C, D ... inverse maps of A^*, B^*, C^*, D^*

k ... edge connecting A,B

M ... midpoint of k

and again in object space:

M^* ... F(M) (is in general not equivalent to P !)

a ... vector from P to M^*

(2) and (3) describe one iteration step of the algorithm (fig.6):

(2) in parameter space the triangle ABC is substituted by the two triangles ACM, BCM and the triangle ABD is substituted by the two triangles ADM, BDM.

In other words, (2) replaces the edge k by the four new edges AM, BM, CM, DM.

(3) in object space the triangle $A^*B^*C^*$ is substituted by the two triangles $A^*C^*M^*, B^*C^*M^*$ and the triangle $A^*B^*D^*$ is substituted by the two triangles $A^*D^*M^*, B^*D^*M^*$.

Again, (3) replaces the edge k^* by the four new edges $A^*M^*, B^*M^*, C^*M^*, D^*M^*$.

(2) and (3) can be repeated until there are no bad edges left.

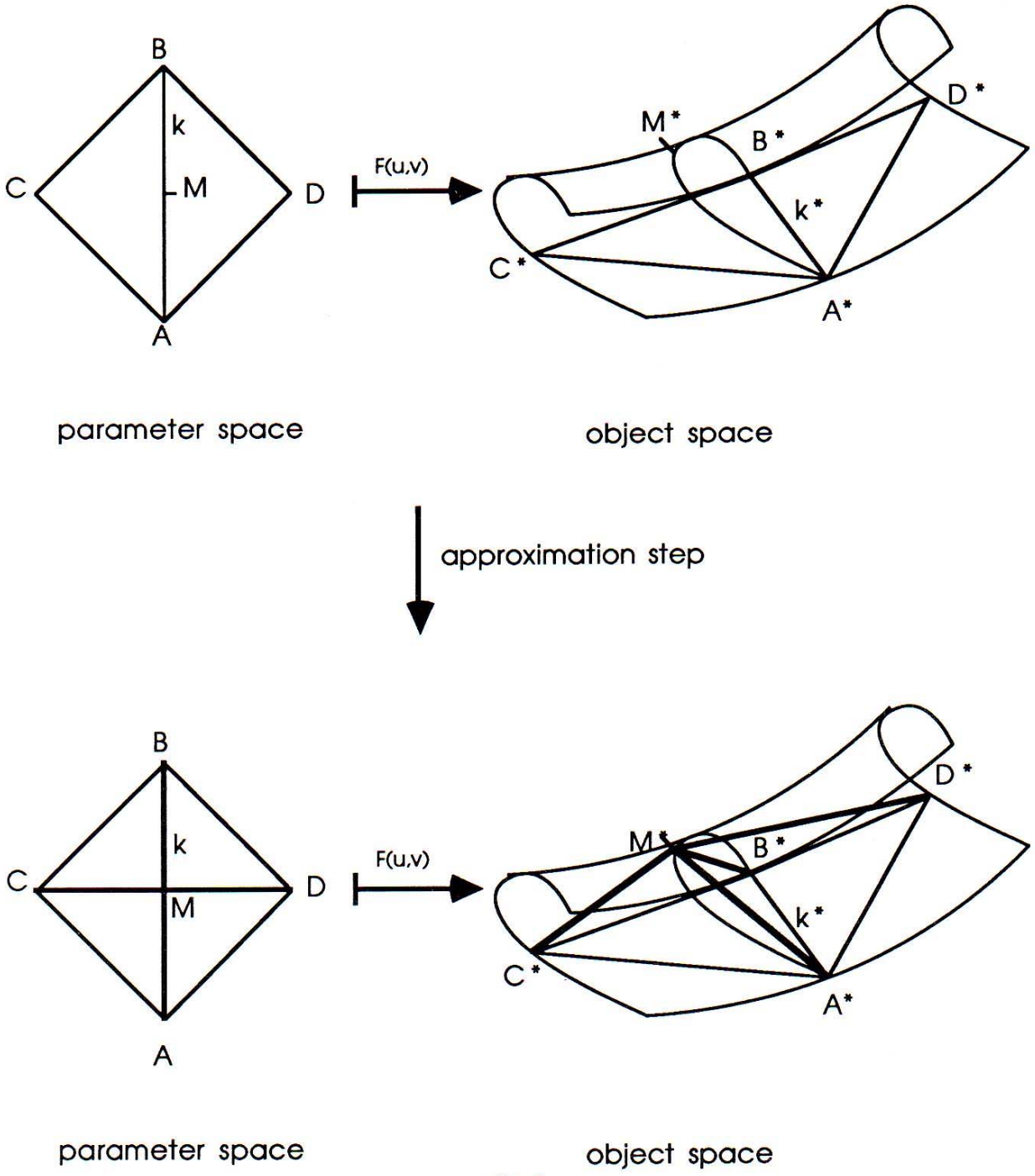

fig.6

5.2 A Measure for Edges

This chapter describes a measure to determine the "bad" edges. A heuristic view will tell us that an edge is not good, if its object space representation is not close enough to the triangular surface patch.

A trivial solution to the problem is to take lal as a measure.

Although this measure is very good in most cases, there are cases where it produces unnecessary subdivisions. Let w be the angle between the planes of the two triangles δ_1, δ_2. Fig.7 and fig.8 indicate two cases in which the trivial measure will lead to a subdivision although it makes no sense.

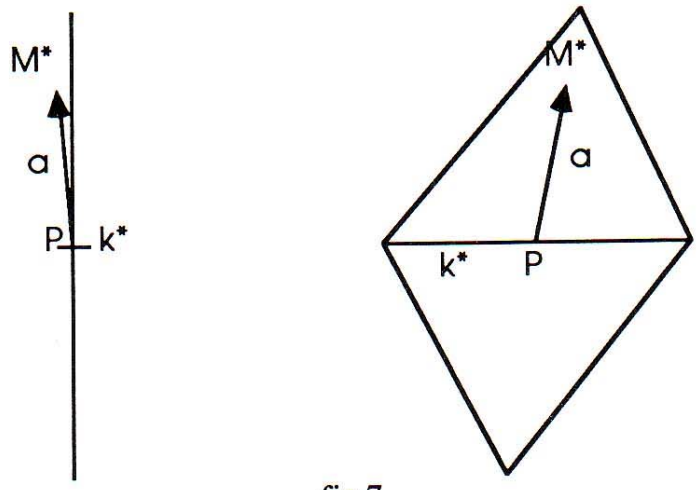

fig.7

$w \approx 180°$ and the vector a is almost parallel to one of the triangle planes.

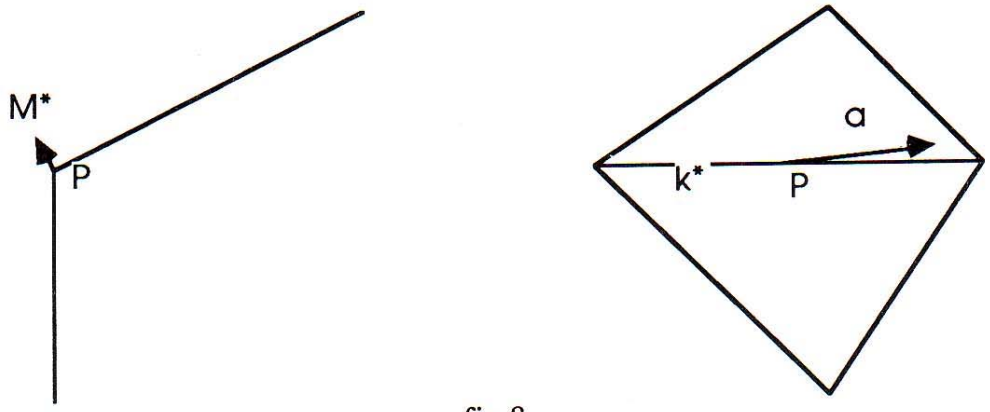

fig.8

$w << 180°$ or $w >> 180°$ and the angle between k^* and a is very small.

To avoid these errors, we will introduce a better measure.

Let d_1 be the absolute distance of M^* from the plane of δ_1, and let d_2 be the absolute distance of M^* from the plane of δ_2 (fig.9).

The improved measure is simply $d = d_1 + d_2$. It can easily be seen that the cases of fig.7 and fig.8 cause no harm anymore.

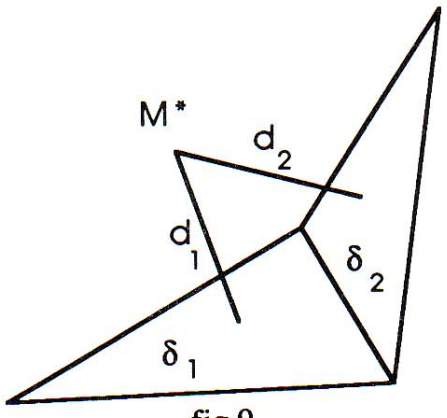

fig.9

Calculation of d:

If p_1, p_2, p_3 are the vectors to the corners of δ_1, then

$n_1 = (p_2 - p_1) \times (p_3 - p_1)$ (x = vector product)

is a normal vector on the plane of δ_1 (fig.10).

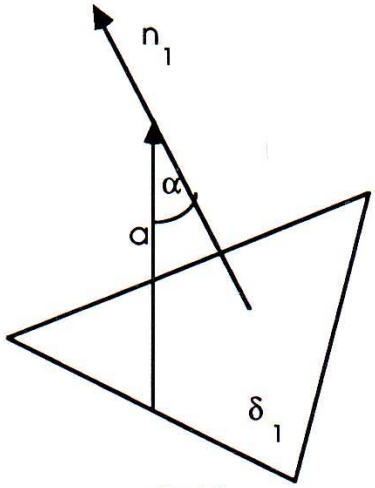

fig.10

Since

$$\cos(\alpha) = \frac{n_1 * a}{|n_1|\,|a|} \qquad (* = \text{dot product})$$

d_1 can be expressed as

$$(4)\quad d_1 = \left|\cos(\alpha)\,|a|\right| = \left|\frac{n_1 * a}{|n_1|}\right|$$

If the equation of the plane of δ_1 is given by

$$k_1x + k_2y + k_3z + d = 0,$$

a normal vector on the plane is $n_1 = (k_1\ k_2\ k_3)^T$.

d_2 is calculated with an analogous formula to (4).

According to Shannon's Sampling Theorem, there will always be the probability to make mistakes with any measure, because there is only a discrete set of testing points. This probability can, of course, be reduced by increasing the set of test points. The following example will demonstrate this (fig.11).

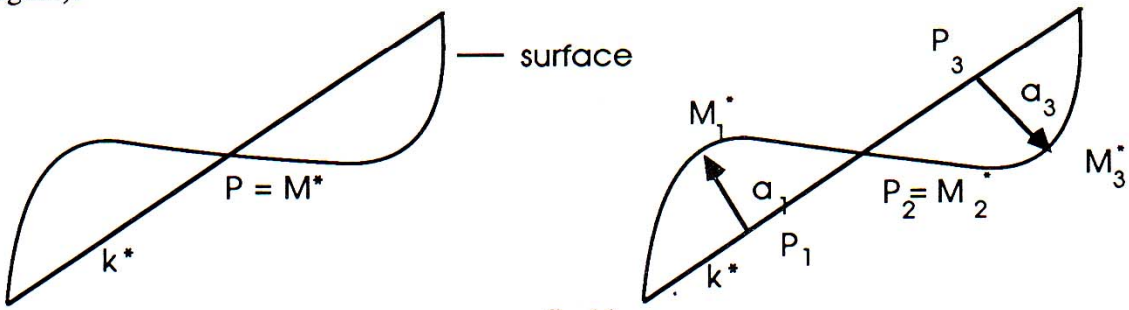

fig.11

Nevertheless, there is a significant tradeoff with efficiency.

5.3 The Worst Edge

From all bad edges we still have to define which of them is the worst. It seems obvious to take the edge with the highest measure d. But to actually ensure that the error is no more than a given eps, it is not essential in which order the bad edges are handled, since all bad edges have to be replaced. The quality of the visual appearance, though, seems to correlate with the shape of the triangles, such that regular triangles increase it.

Therefore it is better to define the worst edge slightly different.
Let E be the set of all edges with d > eps for a given constant eps. Then the edge e ε E is worst, if for all f ε E: |e| ≥ |f|. In other words, the longest bad edge is the worst.

This definition leeds to about the same number of approximation steps, but ensures that long bad edges are replaced early. The overall visual effect increases with the regularity of the approximation, thus with the described sequence of subdivision.

5.4 Data Structure for the Bad Edges

The iteration algorithm for the approximation requires a data structure for the bad edges, that enables a quick execution of the following actions:
(a) Request for the worst edge.
(b) Deletion of the worst edge.
(c) Insertion of new bad edges.

A special version of the heap, the pagoda [6] [11], is an excellent data structure for these actions. It is necessary to define an order for the edges, but this is simply done by the relation "worse". A heap is a tree structure in which the key of a node has no smaller value than its successors. In pagodas the pointers are arranged such that the computational expense for (a), (b) and (c) for n edges is as follows:
(a) is of order O(1)
(b) is of order O(Ld(n))
(c) is of order O(1)

Especially the combination of (a) and (c) is remarkable.

5.5 Using the Approximation Algorithm for Deforming B-rep Solids

After the object has been subdivided into triangles, all vertices are mapped with (1). If two points are connected in parameter space, they will also be connected in object space. The edges are classified and ordered into the pagoda. The approximation algorithm (2) and (3) can be performed just as before.

6. Results

The method described in this paper was implemented in Pascal under VMS. The resulting boundary representation of the deformed solids can be rendered in a variety of ways. In our system we use a ray tracer to be able to achieve shadows and to produce reflecting surfaces. The pictures were produced with the RISS system [5] [21] on a VAXstation 3200 using an additional impuls2300 raster graphics terminal. For this the b-rep representation of the objects was transformed to a Binary Space Partitioning Tree (BSP-Tree) [4] [13].
The approximation algorithm produces about 30 triangles per second.
Fig.12* demonstrates an undeformed scene of two cubes and two prisms embedded in a 3x3x3 grid. This scene was deformed with this control-point grid. The resulting scene consists of 4984 triangles and is illustrated in fig.13.* The resolution of these two pictures is 500x499 pixels.
The car body [7] demonstrated in fig.14* (480x388 pixels) results from the deformation of a cube with a 13x11x3 control-point grid and consists of 4024 triangles.

---

* See page 545 for Figures 12, 13 and 14.

Fig.15*(600x421 pixels) demonstrates a twisted teapot. The original teapot [3] was twisted with a 3x3x3 control-point grid. The surface of the deformed teapot consists of 5184 triangles.

7. Acknowledgements

This work was partly sponsored by the "Forschungsförderungsfonds für die gewerbliche Wirtschaft" and by "Impuls GmbH". We also want to thank our colleagues M. Gervautz and H. Künig for valuable discussions and hints.
The referees led our attention to two further papers which we had no opportunity to read [16] [18].

References

[1]: Barr, A. H.: Global and Local Deformations of Solid Primitives. Computer Graphics, Volume 18, Number 3, 1984.
[2]: Barr, A. H.: Ray Tracing Deformed Surfaces. Computer Graphics, Volume 20, Number 4, 1986.
[3]: Crow, F.: Displays on Display. IEEE Computer Graphics and Applications, Volume 7, Number 1, January 1987.
[4]: Fuchs, H.; Kedem, Z. M.; Naylor, B. F.: On Visible Surface Generation by a Priori Tree Structures. Computer Graphics, Volume 20, Number 4, 1980
[5]: Gervautz, M.; Purgathofer, W.: RISS - Ein Entwicklungs-System zur Generierung realistischer Bilder. Informatik Fachberichte 182, Visualisierungstechniken und Algorithmen, Springer Verlag, September 1988.
[6]: Gonnet, G. H.: Handbook of Algorithms and Data Structures. International Computer Science Series, Addison-Wesley, 1984.
[7]: Griessmair, J.: Verbiegeoperatoren zur Modellierung komplexer Objekte. Diplomarbeit, TU-Wien, 1988.
[8]: Haas, I.: B-Splines und ihre Anwendung in der Computergraphik. Diplomarbeit, TU-Wien, 1987.
[9]: Herzen, B. Von; Barr, A. H.: Accurate Triangulations of Deformed Intersecting Surfaces. Computer Graphics, Volume 21, Number 4, 1987.
[10]: Joy, K. I.; Bhetanabhotla, M. N.: Ray Tracing Parametric Surface Patches Utilizing Numerical Techniques and Ray Coherence. Computer Graphics, Volume 20, Number 4, 1986.
[11]: Jones, D. W.: An Empirical Comparison of Priority-Queue and Event-Set Implementations. Communications of the ACM, Volume 29, Number 4, April 1986.
[12]: Kajiya, J. T.: Ray Tracing Parametric Patches. Computer Graphics, Volume 16, Number 3, 1982.
[13]: Künig, H.: Modellierung mit dem BSP-Modell. Diplomarbeit, TU-Wien, 1988.
[14]: Lasser, D.: Bernstein-Bézier-Darstellung trivariater Splines. Dissertation, TH-Darmstadt, 1987.
[15]: Newman, W. N.; Sproull, R. F.: Principles of Interactive Computer Graphics. International Student Edition, Second Edition, Mc-Graw Hill 1979.
[16]: Parry, S.: Free-form Deformation in a Constructive Solid Geometry Modelling System. Ph. D. Dissertation, Brigham Young University, 1986.
[17]: Rogers, D. F.; Adams, J. A.: Mathematical Elements for Computer Graphics. McGraw-Hill Book Company, 1976.
[18]: Saia, A.; Bloor, M. S.; de Pennington, A.: Sculptured Solids in a CSG Based Geometric Modelling System. Proc. IMA Conf. 'The Mathematics of Surfaces II', Oxford University Press, 1987.
[19]: Sederberg, T. W.; Parry, S. R. : Free-Form Deformation of Solid Geometric Models. Computer Graphics, Volume 20, Number 4, 1986.
[20]: Steinberg, H. A.: A Smooth Surface Based on Biquadratic Patches. IEEE Computer Graphics and Applications, Volume 4, Number 9, September 1984.
[21]: Tschanter, L.: Ray-Casting für ein dreidimensionales Flächenmodell. Diplomarbeit, TU-Wien, 1986.

---

* See page 545 for Figure 15.