ELSEVIER Computer Aided Geometric Design 11 (1994) 675–686

COMPUTER AIDED GEOMETRIC DESIGN

Construction of a $VC1$ interpolant over triangles via edge deletion

Claudia Cottin, Ruud van Damme

Department of Applied Mathematics, University of Twente, P.O. Box 217, 7500 AE Enschede, Netherlands

Received August 1991, revised October 1993

Abstract

We present a construction of a visually smooth surface which interpolates to position values and normal vectors of randomly distributed points on a 3D object. The method is local and uses quartic triangular and bicubic quadrilateral patches without splits. It heavily relies on an edge deleting algorithm which, starting from a given triangulation, derives a suitable combination of three- and four sided patches.

Keywords: 3D reconstruction; Interpolation to scattered data; Visually smooth interpolants; Bézier patches

1. Introduction

The construction of visually smooth surfaces which interpolate to position values and normal vectors of randomly distributed points on a 3D object, is an important issue in CAGD. For easy evaluation and manipulation, one often aims at a local method which uses low degree polynomial patches. A recent review and classification of such methods can be found in (Peters, 1990a).

The first task that one encounters, but that we do not address to here, is the construction of a polygon with vertices in the interpolation points such that a polynomial element can be rendered over each of the facets. Usually, the most simple and versatile polygons result from a triangulation of the data points (see, e.g., (Schumaker, 1987)), and we will start with such a configuration. Our challenge is to avoid the use of blending methods, which would enforce the use of rational polynomials, and of splits, which would enlarge the total number of facets.

An interesting and thorough discussion of the conditions under which a polynomial interpolant over three- and four-sided facets can be constructed without splits

SSDI 0167-8396(93)E0052-F

is given in (Peters, 1991). It turns out that the interpolation problem can be solved in a local fashion with polynomial patches of coordinate degree 4, provided that certain compatibility conditions are satisfied. The major difficulties arise if an even number of patches meets around one vertex.

The interpolation method that we present in this paper overcomes these problems by deriving, from a given triangulation, a combination of three- and four-sided patches such that only odd numbers of patches around each vertex occur. (In the sequel such vertices will be called "odd vertices", in contrast with "even vertices".) This enables us to construct a smooth quartic/bicubic interpolant by using certain well-known $VC1$ conditions developed by Farin (1982, 1983). The reason for choosing Farin's conditions is that the resulting algorithm is relatively simple as it does not contain a lot of free parameters and the necessary computations are fairly easy to perform. We remark however, that in principle our construction could of course be carried out in combination with other (more complex) $VC1$ conditions, since it is a general phenomenon that the case of odd vertices is easier to handle than the one of even vertices.

In Section 2, we briefly review the $VC1$ conditions used for the construction of our interpolant. The interpolation algorithm is outlined in Section 4. It heavily relies on an edge deletion algorithm which we present in Section 3.

2. A sufficient $VC1$ condition

In (Farin, 1982, 1983), Farin derived simple conditions for a visually smooth ($VC1$) transition between two triangular or rectangular polynomial patches, i.e., a join with continuously varying tangent plane. He pointed out that the conditions for such a join between any combination of two bicubic four-sided and quartic three-sided patches with cubic boundary curves are basically the same. We represent the polynomial patches in the well-known Bézier–Bernstein form. Let $S_0, \ldots, S_3$ denote the control points along the common cubic boundary curve.

The interior control points of the bicubic quadrilateral or quartic triangular patches are denoted by R_1, R_2, T_1 and T_2 (See Fig. 1). The boundary curves are always taken to be *cubic*. Therefore we can describe them by four control points S_0, S_1, S_2 and S_3. Moreover, we let R_0, R_3, T_0 and T_3 be the neighbouring control points on the edges, also in their *cubic* representation.

For actual computation of a triangular patch in a final stage, these boundary points in the *quartic* representation can be obtained by the degree elevation algorithm, i.e.,

$$\widetilde{R_0} = \tfrac{1}{4}S_0 + \tfrac{3}{4}R_0,$$

and so forth (Boehm et al., 1984).

If we suppose that, for fixed control points on the edges, there exist numbers ν, w_1, and w_2 such that

$$\begin{aligned} \Delta_{\text{up}}(T_0 - S_0) &= w_1(S_1 - S_0) + \nu\Delta_{\text{down}}(R_0 - S_0), \\ \Delta_{\text{up}}(T_3 - S_3) &= w_2(S_2 - S_3) + \nu\Delta_{\text{down}}(R_3 - S_3)\ , \end{aligned} \tag{1}$$

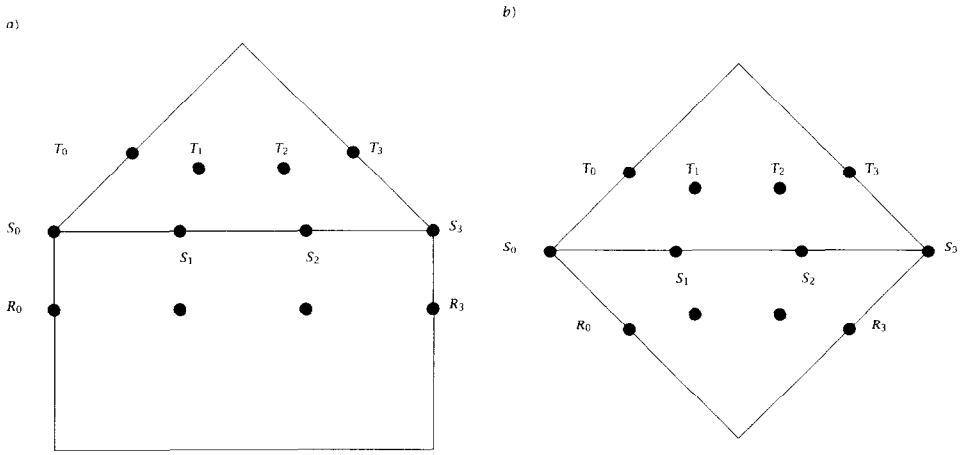

Fig. 1. A quadrilateral-triangle join (a) and a join between two triangles (b).

then a $VC1$ join can be obtained by choosing

$$\begin{aligned} T_1 - \nu R_1 &= (1-\nu)S_1 + \tfrac{2}{3}w_1(S_2 - S_1) + \tfrac{1}{3}w_2(S_0 - S_1), \\ T_2 - \nu R_2 &= (1-\nu)S_2 + \tfrac{1}{3}w_1(S_3 - S_2) + \tfrac{2}{3}w_2(S_1 - S_2). \end{aligned} \tag{2}$$

The factor Δ_{up} (Δ_{down}) in (1) equals $\frac{3}{4}$ or 1 when the upper (lower) patch is a triangle or a quadrilateral, respectively, and stems from the degree elevation proccss.

We stress that the conditions (2) are only sufficient and not necessary. However, they are easier to handle than more general ones which can be found, e.g., in (Peters, 1991, 1990b).

By (1), we arrive at the geometrical interpretation of the coefficient ν that

$$\frac{\text{area}(T_0, S_0, S_1)}{\text{area}(R_0, S_1, S_0)} = -\nu\frac{\Delta_{\text{down}}}{\Delta_{\text{up}}} = \frac{\text{area}(T_3, S_2, S_3)}{\text{area}(R_3, S_3, S_2)}. \tag{3}$$

Now we consider one vertex V with M surrounding three- and four-sided patches. The interior control point of patch Π_m next to V is denoted by C_m, and the control points on the mth edge by P_m and R_m, respectively, see Fig. 2. Here and in the sequel the indices are defined modulo M.

We suppose that, for each edge, Eq. (1) is satisfied with suitable numbers ν^m, w_1^m, w_2^m. Trying to satisfy (2) around V, leads to the equations

$$C_{m+1} - \nu^m C_m = B_m, \tag{4}$$

with

$$B_m = (1-\nu^m)P_m + \tfrac{2}{3}w_1^m(R_m - P_m) + \tfrac{1}{3}w_2^m(V - P_m).$$

Since the coefficient $-\nu^m$ denotes a ratio of two areas (3), we have that

$$\prod_{m=0}^{M-1} \nu^m = (-1)^M. \tag{5}$$

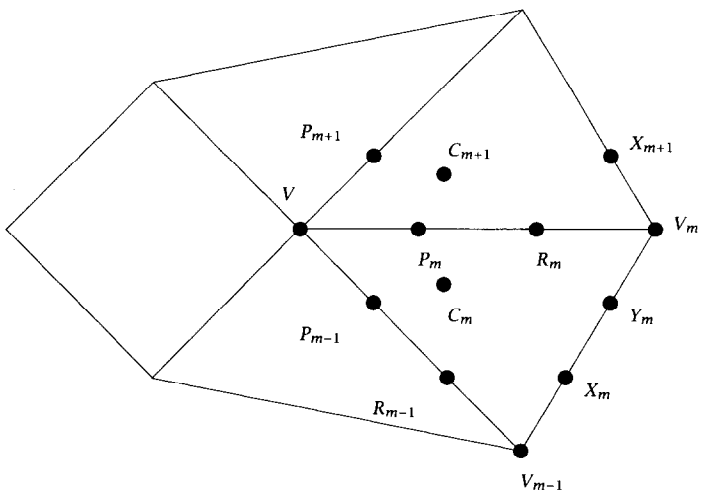

Fig. 2. A vertex and the surrounding Bézier points. A vertex and the surrounding Bézier points. Note that C_{m+1} and C_m correspond to T_1 resp. R_1 in the previous figure.

Hence, the determinant of the linear system (4) is $1+(-1)^{M+1}$, and thus there always exists a unique solution for the M points C_m, if M is odd. If M is even, the points P_m and R_m have to satisfy certain rather complicated additional conditions in order to guarantee existence of a solution; compare also the discussion in (Peters, 1991).

3. An edge deletion algorithm

The main idea of the $VC1$ construction presented here is based on the previous observation that system (4) can easily be solved for vertices with an odd number of surrounding three- and four-sided patches. Starting with a given triangulation, we delete edges in a suitable way such that we end up with only such odd vertices. If we are dealing with closed bodies, this is always possible under the mild restriction that the total number of vertices is even. Then also the number of even vertices is even, equal to $2N$, say. The proof of this statement can be given as follows: Let E, T, V_n be the number of edges, triangles and vertices (with n edges connected to it). Then Euler's formula (doCarmo, 1976) gives

$$E - T = \sum_n V_n - \chi,$$

where $\chi \in \mathbb{N}$ is the Euler–Poincaré characteristic, which is even for closed bodies ($\chi = 2$ for sphere-like objects). The following identities are straightforward:

$$3T = 2E = \sum_n nV_n,$$

leading to

$$\sum_n (6-n)V_n = 6\chi.$$

With this one easily proves that the number of odd vertices is always even, from which the assertion follows.
Of course for non-closed surfaces the restriction that the number of even vertices must be even is not necessary, because boundary vertices may also be even.

In principle, the edge deletion can be carried out by forming N (disjoint) pairs of all even vertices, and by deleting a connecting path of edges for each couple. The main problem is that we must guarantee that, in doing so, no n-sided patches with $n > 4$ are generated. To this end, we formalise the edge deletion algorithm in the following way.

A path consists of edges $(e_1, \ldots, e_{K-1})$ and vertices $(V_1, \ldots, V_K)$, where $e_k = (V_k, V_{k+1})$. A path is called allowed if any two edges of this path are not part of the same triangle. From any path between two even vertices, the following algorithm constructs an allowed path without adding new vertices on this path.

Algorithm *Reduce Path* (*N*: *path number*)

1. If any two vertices of the path are identical, then the path intersects itself. We delete the unnecessary loop from the path and repeat this procedure until all self-intersections are eliminated.
2. Without the path intersecting itself, two *adjacent* edges may still be part of the same triangle. We replace any such pair by the third edge of their common triangle, and remove their common vertex from the vertex-list of the path. By repetition of this step, we obtain an allowed path. (See Fig. 3.)

End Algorithm *Reduce Path*

Difficulties may also arise from "unsuitable" intersections between two paths. We call an intersection between two paths allowed if no pair of edges at the intersection point is part of the same triangle. If, for all N pairs of even vertices, we have found allowed paths with only allowed intersection points, then the deletion of all corresponding edges leaves us with the desired configuration of

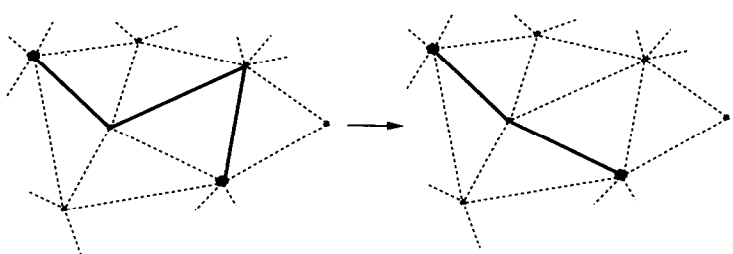

Fig. 3. Reduction of two adjacent edges in one triangle.

triangles and quadrilaterals. A suitable procedure, which again does not include any new vertices, can be formulated in the following way.

Algorithm *Delete Even Edges*

- Form N pairs of even vertices and form paths between them. For example, this can be done with the following algorithm:

  Algorithm *Initialise*

  Make a list A of all the vertices that have to be paired, i.e., initially the even vertices. For all the vertices from A we determine the vertices (even or odd) that are at distance 1.

  As soon as one encounters a possible pair, add this to the list of pairs, the path being essentially the shortest path between these two vertices. Subsequently both vertices are deleted from list A.

  If all the (remaining) vertices in A are treated in this way, and A is non-empty, we proceed by determining all vertices at distance 2, 3, ... of the remaining vertices in A, repeating the previous step until A is empty.

  End Algorithm *Initialise*

  Let the path number of the first path be $J = 1$. (A path with path number J will always be referred to as path J.) Reduce this path by algorithm *Reduce Path*(1).

- While $J < N$ Do

  (1) $J := J + 1$. Perform algorithm *Reduce Path*(J).

  (2) Detect the first (in terms of the vertex-list of path J) non-allowed intersection with any of the previously considered paths. If no such intersection exists, then perform step 1. Else call the intersecting path I and the point of intersection V. Now there are two possibilities:

  (a) Path I and J intersect at least once more. Then we determine the intersection W with the largest index in the vertex-list of J. So the vertex-list $v(J)$ of J has the form

  $$v(J) = (V_{J_l}, \ldots, V, \ldots, W = V_{J_m}, \ldots, V_{J_K}),$$

  and the one of I can be written as (after a possible reversion)

  $$v(I) = (V_{I_K}, \ldots, V = V_{I_m}, \ldots, W, \ldots, V_{I_l}).$$

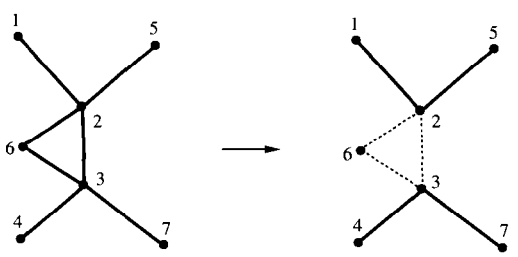

Fig. 4. $v(J) = (1, 2, 3, 4)$, $v(I) = (5, 2, 6, 3, 7) \rightarrow v(J) = (1, 2, 5)$, $v(I) = (7, 3, 4)$.

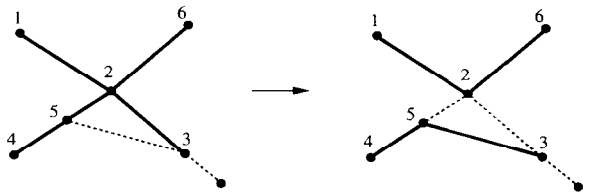

Fig. 5. Second possibility: only one intersection between two paths. $v(I) = (1, 2, 3)$, $v(J) = (4, 5, 2, 6) \rightarrow v(I) = (1, 2, 6)$, $v(J) = (3, 5, 4)$.

Now we re-define the paths I, J as follows (for an example see Fig. 4):

$$v(J) = (V_{J_l}, \ldots, V, V_{I_{m+1}}, \ldots, V_{I_K}),$$

$$v(I) = (V_{I_l}, \ldots, W, V_{J_{m+1}}, \ldots, V_{J_K}).$$

(b) Paths I and J intersect exactly once: As this is a non-allowed intersection, and I and J have been reduced, there exist edges e_I of I, and e_J of J which belong to one triangle. We redefine I, J such that this pair belongs to *one* path and that J contains V_{J_1} (see Fig. 5).
Since one of the two paths can be reduced, the intersection point disappears.

Perform *Reduce Path*(I), *Reduce Path*(J), and repeat step 2.

- Delete all edges which are part of one of the constructed paths.

End Algorithm *Delete Even Edges*

Observe that step (2a) covers in particular the case that two paths have one or more common edges. The case that several paths intersect in one point is treated successively by the algorithm. Moreover, step (2b) also works if the intersection point is an *even* vertex, which can happen if it is the first (or last) point of a path.

Since the algorithm *Reduce Path* does not generate new vertices, no new intersections can be created, and therefore this algorithm is finite: at each step the number of non-allowed intersection points decreases. Moreover, the newly created path I in both (2a) and (2b) cannot intersect any other of the previously considered paths at that point. Therefore it can be treated as an "old" one.

The previous considerations show that there always exists a solution to our edge deleting problem. Of course the construction of an appropriate configuration is by no means unique. A good way to fix the algorithm would be a minimisation of the number of edge deletions: Each edge deletion reduces the number of available control points, and quadrilaterals more-over demand more care than triangles since we have to avoid degeneracies (i.e., angles $\geq \pi$). In general this is an immense optimisation problem.

But it is also clear that a good initialisation of the N pairs and their connecting paths will reduce the time necessary to from a proper configuration. Obviously one never can guarantee that there exists no initial triangulations for which algorithm *Initialise* will perform poorly and/or the number of deletions is high with respect

to the real minimal solution. We can only validate it by saying that in almost in all our test cases it immediately yielded a proper configuration (so the procedure *Delete Even Edges* was not even called upon) *and* the number of deletions was optimal.

As a possible improvement of the total algorithm we mention that possibly a sort of division of space can lead to larger efficiency of the second step of algorithm *Delete Even Edges*. Theoretically however, it is clear that our algorithm is polynomial in N. Moreover in all our examples (with data sets of several hundreds of vertices) almost all time was spent in the reconstruction part of Section 4 and almost none in creating a proper configuration. Therefore we did not take further effort in creating such an optimised method.

4. The $VC1$ Construction

In this section, we carry out a $VC1$ construction that results from the observations in Section 2.

For a given triangulation, we perform the edge deletion algorithm from Section 2 such that we end up with a configuration of triangles and quadrilaterals, and that each vertex is surrounded by an odd number of patches. The Bézier points at the vertices must equal the data points, as we demand interpolation.

In the next step, we compute target control points for the cubic wire frame curves; i.e., control points which are, in some sense, in an ideal position. We chose to do it as in (Cottin and van Damme, 1990): For all edges (VV_m as in Fig. 2) we determine a plane $\mathcal{T}$ in which the wire frame curve should lie such that it forms a

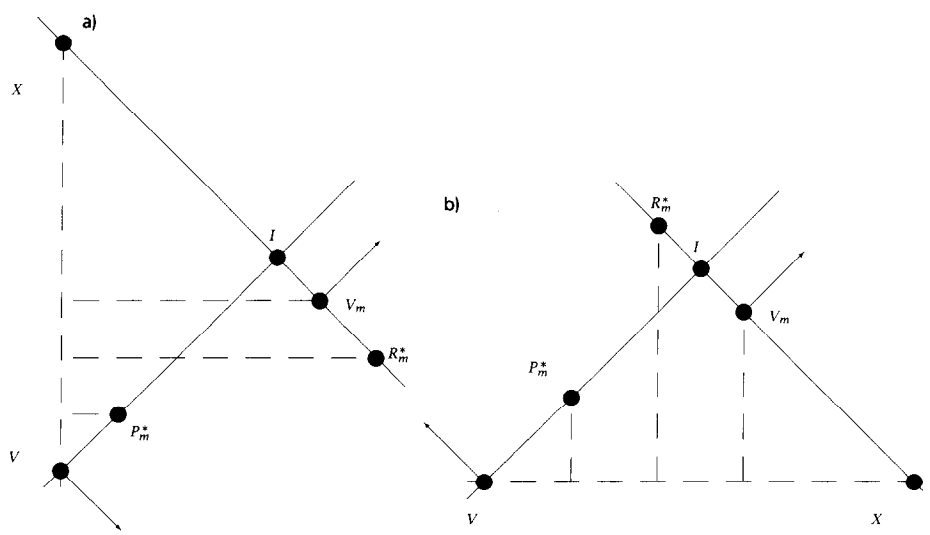

Fig. 6. Boundary curve construction: the two possible choices of X.

reasonably short path between V and V_m. A possible choice is the plane through V and V_m which makes equal and minimal angles with the two normals at the vertices. The intersection of $\mathcal{T}$ with the two tangent planes at the vertices $\mathcal{T}$ and $\mathcal{T}_m$ defines two tangent lines t and t_m through V and V_m respectively. Now let I be the intersection of t and t_m. If $|I-V|>|I-V_m|$, we pick X on t_m such that $|X-I|=|V-I|$ (and visa versa). Then we construct P_m^* and R_m^* as in the functional case with domain axis $X-V$, $X-V_m$ respectively. There are always two possible choices for X (cf. Fig. 6), but only one is consistent with the orientation of the tangent planes. Note that this also works in the case of parallel tangent planes. Only in the case that $\mathcal{T}$ and $\mathcal{T}_m$ *coincide* but both normals are opposite in direction, it is clear that a cubic method can never work.

There is no need to impose the planarity of the boundary curves, but it it seems to be the simplest choice. In fact, P_m^*, R_m^* are only targets points and as regularity demands

$$\text{angle}(P_m - V, P_{m+1} - V) < \pi$$

the final reconstruction will in general not contain only planar boundary curves. We will come back to this later on.

We now have to prescribe the coefficient ν^m for all edges e_m with abutting patches Π_m and Π_{m+1}. As follows from Section 2, the only principal restriction on ν^m is that it can be written in the form

$$\nu^m = -r_m / r_{m+1}, \tag{6}$$

where r_m for every m are positive numbers that depend only on Π_m. In Farin's $VC1$ construction (based on splits) presented in (Farin, 1983), the quantity r_m was taken as the area of the (in that case triangular) patch Π_m. Since we have already computed the target points, we prefer to use them in the determination of r_m. This also avoids difficulties in the case of quadrilaterals. We choose r_m as the average area of the three or four triangles formed by each vertex of the patch Π_m and its closest two Bézier target points on the edges of this patch. E.g., for the triangle Π_m in Fig. 2, we take:

$$r_m = \tfrac{1}{3}\Delta_m(\text{area}(V, P_{m-1}^*, P_m^*) + \text{area}(V_m, R_m^*, Y_m^*) + \text{area}(V_{m-1}, X_m^*, R_{m-1}^*)), \tag{7}$$

where $\frac{1}{3}\Delta_m = \frac{3}{4}\cdot\frac{1}{3} = \frac{1}{4}$ (see Eq. (3)); in the case of a quadrilateral there are four terms with the same factor: $\frac{1}{4}\Delta_m = 1\cdot\frac{1}{4} = \frac{1}{4}$.

The motivation for this choice is obvious from (3): The right-hand and the left-hand side of (3) are *not* necessarily equal for the target control points; thus we compute r_m as a suitable average. Since it must be the same constant for all edges of the patch Π_m, (7) seems the natural choice.

In the next step, we fix the wire frame control points P_m such that a solution of (4) exists. According to (1), these points should satisfy the equations

$$\Delta_{m+1}(P_{m+1} - V) = \nu^m \Delta_m(P_{m-1} - V) + w_1^m(P_m - V). \tag{8}$$

The coefficients w_1^m in (8) are free parameters. Hence, since all P_m lie in the given tangent plane at V, (8) consists of $2M$ equations for the $3M$ parameters P_m, w_1^m, $m = 0, \ldots, M-1$. One can easily convince oneself that this system always has a meaningful solution (i.e., one without the P_m being in a meaningless position, e.g., all identical). One particular solution could be obtained along the lines of the construction in (Farin, 1983, Section 2.5) where a very similar equation is solved. In order to possibly obtain better results, we minimise

$$\sum_{m=0}^{M-1} (P_m^* - P_m)^2$$

subject to (8), where the P_m^* are the previously computed target control points. Moreover, as we wish to exclude patches with angles larger than π we add the inequality constraints:

$$\alpha < \text{angle}(P_m - V, P_{m+1} - V) < \pi(1-\alpha),$$

with α some small positive constant. In practise these constraints are easy to fulfill.

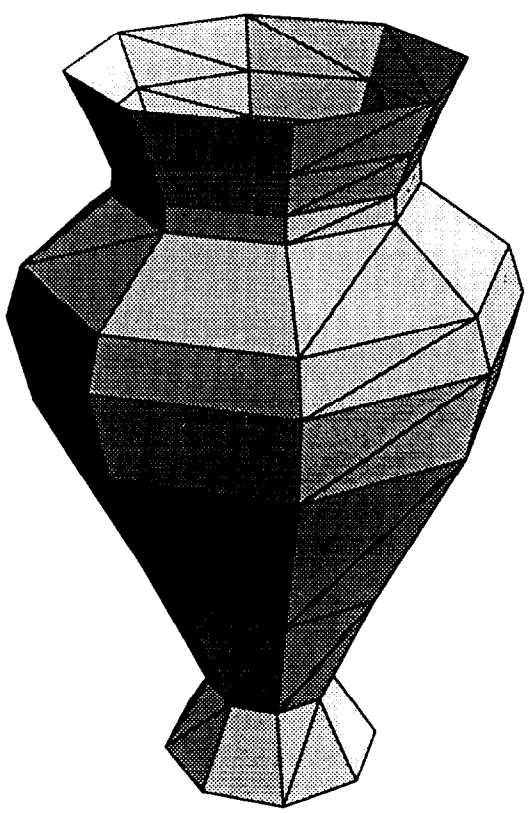

Fig. 7. Example of the "triangulation" after the edge deleting algorithm, discussed in Section 3.

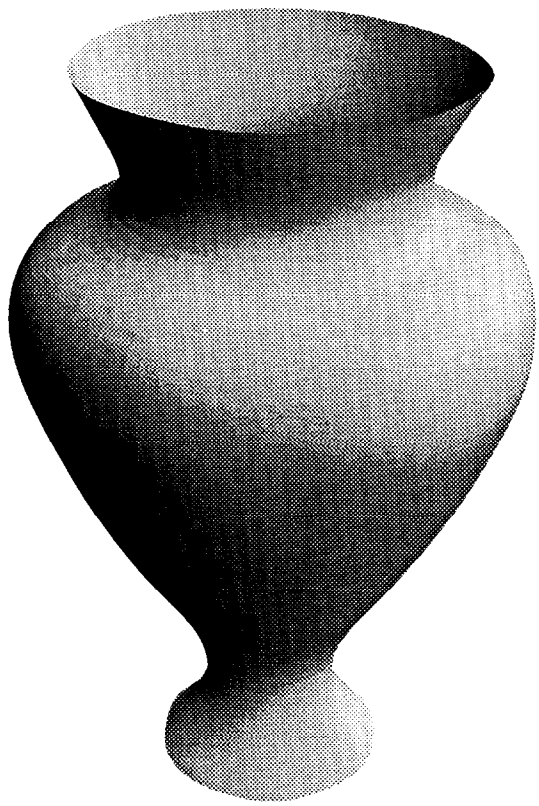

Fig. 8. Smooth surface as a result of our algorithm discussed in Section 4, with data as in Fig. 7 supplemented with normals.

Now that we have fixed P_m and w_1^m, and thus also w_2^m and R_m by performing the same procedure for each vertex, the missing interior control points can uniquely be determined from (4).

5. Example

We implemented the algorithms described in this paper and we now wish to present a resulting example. The triangulated data set after edge deleting is given in Fig. 7, whereas the final smooth surface is depicted in Fig. 8.

For the numerical optimisation we used a simple quadratic programming method, which, due to the low dimensionality of all the subproblems had no difficulty in finding a local optimum.

The method as we propose it, has hardly any free parameters, only the construction of the wire frame can be changed and furthermore one could vary the choice of the numbers r_m in (6). In fact we experimented with the last possibility

for different 3D objects, but the results are pretty insensitive to the different choices. Also other constructions of the initial wire frame led to similar results.

References

Boehm, W., Farin, G.E. and Kahmann J., (1984), A survey of curve and surface methods, Computer Aided Geometric Design 1, 1–60.

Cottin, C. and van Damme, R. (1990), 3D reconstruction of closed objects by piecewise cubic triangular Bézier patches, Memorandum no. 885, University of Twente, 1990; in: *Proc. Mathematics of Surfaces III*, Bath, to appear.

do Carmo, M.P. (1976), *Differential Geometry of Curves and Surfaces*, Prentice-Hall, Englewood Cliffs, NJ.

Farin, G.E. (1982), A construction for visual C^1 continuity of polynomial surface patches, *Computer Graphics and Image Processing* 20, 272–282.

Farin, G.E. (1983), Smooth interpolation to scattered 3D data, in: Barnhill, R. and Boehm, W., eds., *Surfaces in Computer Aided Design*, North Holland, Amsterdam, 43–64.

Peters, J. (1990a), Local smooth surface interpolation: a classification. *Computer Aided Geometric Design* 7, 191–195.

Peters, J. (1990b), Smooth mesh interpolation with cubic patches, *Computer-Aided Design* 22, 109–120.

Peters, J. (1991), Smooth interpolation of a mesh of curves, *Constructive Approximation* 7, 221–246.

Schumaker, L.L. (1987), Triangulation methods, in: Chui et al., C.K. eds., *Topics in Multivariate Approximation*, 219–232.