Smooth surface reconstruction from cross-sections using implicit methods

Michael S. Floater and Geir Westgaard

May 1995

Abstract. Implicit methods for constructing geometric models of three-dimensional objects from cross-sectional data are discussed. The basic approach is to interpolate the contours with an implicitly defined surface which can afterwards be approximated by a triangulation. The main emphasis is put on making the surface smooth. In particular a tangent plane continuous, shape preserving surface is computed.

§1. Introduction

In many applications, for example medicine and reverse engineering, objects are represented in the form of a contour stack, a sequence of parallel cross-sections consisting of one or more closed contours. Typically, a sample of points representing each contour is obtained from a cross-sectional image of a physical object. A major goal is to construct a geometric model of the object from the data. The simplest possible model is a surface triangulation, a non-intersecting collection of joined triangular facets [18].

In this paper we use an implicit method to reconstruct surfaces from cross-sections. We first choose a suitable bounding box $\Omega \in \mathbb{R}^3$ enclosing the contour stack. A surface S is then constructed in the form of the zeros of a function $f : \Omega \to \mathbb{R}$. By the Implicit Function Theorem [16], the smoother f is, the smoother S is. Moreover if f is C^1 we can calculate the surface normal of S given by ∇f. It will be shown how a suitable f can be constructed with at least C^1 continuity. The surface S will therefore be tangent plane continuous.

Once f has been found, a visual image of S can easily be generated by techniques from computer graphics. This is often a goal in itself in medical imaging. However our main interest is reverse engineering where one wants to construct smooth surfaces from measured data for use in computer-aided design (CAD). We therefore generate a triangulation which approximates S for this is the obvious start point for a more sophisticated approximation using piecewise polynomials for example.

The methods we present here are motivated by the paper by Jones and Chen [10]. They constructed surfaces from cross-sections using a field function in each slice and the marching cubes algorithm to generate a surface consisting of polygonal facets. Our aim is to make smoother surfaces by employing some simple spline methods and differential equations.

The method of Jones and Chen is really a reformulation of the set-valued approximation introduced by Levin [11]. Moreover Levin had already suggested using shape preserving spline techniques in order to increase the order of approximation of the construction method. In a survey article on various methods for surface reconstruction from cross-sections Schumaker [17] describes Levin's method in terms of a *voxel map*. If a three-dimensional object is scaled and placed inside the unit cube and the cube divided into *voxels*, small cubes $V_{i,j,k}$, then a *voxel map* is a mapping which assigns a value of 1 or -1 to each voxel. Those voxels lying inside the object are given the value 1 and the remainder -1. This provides a fast way of generating a computer image of the object.

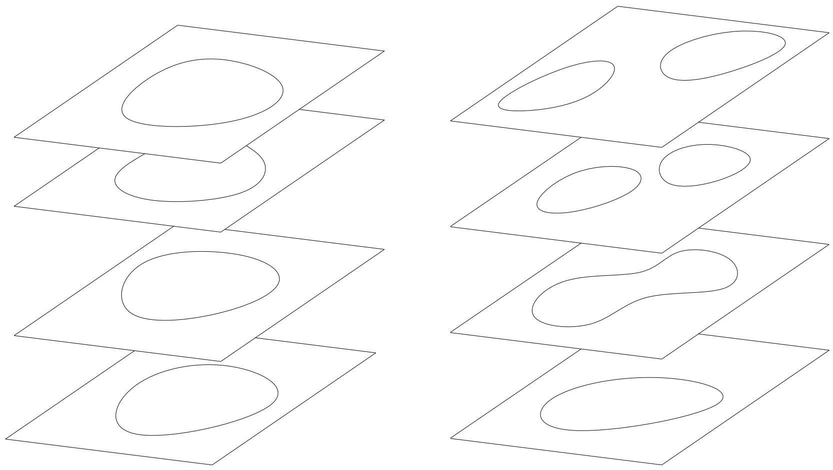

Fig. 1. Contours from objects without and with a branch.

The voxel map will only be useful if there are very many voxels, and a total of $512 \times 512 \times 512$ is not uncommon. The view taken in the present paper is to concentrate on the theoretical and numerical construction of f, independently of computer graphics techniques. Provided f is smooth enough, a good approximation to S can be generated by using a relatively small 3D grid. Grids of the order of $50 \times 50 \times 50$ seem to often be sufficient.

The main advantage of implicit methods are that objects having branches are automatically handled in a natural way. A different method is to generate a tensor-product bivariate spline surface through the contours. But this only applies when there are no branches. A third method which has received a lot of attention consists of building an interpolatory triangulation when the contours are represented as piecewise linear curves. One matches each pair of slices using some heuristics. This method can be very complex when there are many branches. The paper by Choi and Park [2] illustrates the kind of complexity which may be encountered. Furthermore the smoothness of these triangulations is restricted by the density of the slices. There will only be one layer of triangles between each pair of slices.

Schumaker [17] provides a good overview of various methods for constructing surfaces from cross-sections.

Another area where an implicit method has been used is surface construction from unorganized points [9]. One could regard cross-sectional data as a special case of scattered data sampled from a surface. The advantage of cross-sectional data is that it already has some topological structure.

§2. Constructing a smooth surface

Suppose we are given a set of planes in $\mathbb{R}^3$, each normal to the z axis. Their z coordinates are assumed to be in increasing order $s_1, \ldots, s_n$. In each plane $\{z = s_l\}$, one or more contours $\Gamma_{m,l} \subset \mathbb{R}^3$, assumed to be *simple*, i.e. closed and non-intersecting, are specified. Since the z coordinate is constant in each contour we can write $\Gamma_{m,l} = (\gamma_{m,l}, s_l)$ with $\gamma_{m,l} \subset \mathbb{R}^2$. Two examples of data sets are illustrated in Figure 1. It will be assumed in this section that the contours are smooth.

Given the planes and contours, the goal is to construct a smooth implicitly defined surface followed by a triangulation which approximates it. Here we consider a triangulation to be connected, although it can be open or closed.

To get an idea of how the implicit method behaves, consider the first example in Figure 1 where there is one contour per slice. We will find, roughly speaking, that because the interiors of each consecutive pair of these contours overlap when viewed from above, a single surface will be produced, whose topology is that of an open-ended cylinder. In the second example of Figure 1 a single surface will be generated but its topology will be different to the first example.

The method used to construct a triangulation has two phases.

(1) Find, for a suitable bounding region $\Omega \subset \mathbb{R}^3$, a function $f : \Omega \to \mathbb{R}$ satisfying the properties: (a) that the surface $S = \{(x, y, z) \in \Omega : f(x, y, z) = 0\}$ interpolates the contours, i.e. such that $f(\Gamma_{m,l}) = 0$; and (b) that within each plane, $f\mid_{z=s_l}$ changes sign if and only if a contour is crossed, and (c) that $f\mid_{z=s_l}$ is positive on the boundary of Ω.

(2) Approximate S by a triangulation. It is an advantage to make f as smooth as possible for then the Implicit Function Theorem ensures that S will be smooth except at critical points ($\nabla f = 0$). In many of the examples we construct, f has no critical points on S.

The remainder of this section is devoted to the construction of f. The implementation of both (1) and (2) will be handled in Section 3.

The first step is to bound the contours by a suitable set Ω. The easiest way is to choose an enclosing box whose edges are parallel with the coordinate axes. The obvious choice is to define

$$\Omega = [x_1, x_2] \times [y_1, y_2] \times [s_1, s_n],$$

to be the smallest set containing every contour $\Gamma_{m,l}$. However since we would like the surface to interpolate the contours smoothly it may necessary to enlarge the box a little in the x and y coordinates. For all the examples we have run, we have found that a magnification in (x, y) of 1.33 is more than adequate.

Now let

$$D = [x_1, x_2] \times [y_1, y_2],$$

the projection of Ω onto $\mathbb{R}^2$. Constructing f now is split into two parts.

(a) For each plane l, construct $f_l : D \to \mathbb{R}$ such that $f_l(\gamma_{m,l}) = 0$. In addition ensure that f_l changes sign if and only if a contour is crossed, and that f_l is positive on ∂D.

(b) Construct f by interpolating the f_l with a spline (piecewise polynomial) in the z direction.

In [10], f_l is taken to be the *signed distance function*. In order to describe the signed distance function, let

$$h_l(x, y)$$

be defined to be the distance between the point (x, y) and the nearest point amongst the contours $\{\gamma_{m,l}\}_m$. If (x, y) lies inside any contour, we set $f_l(x, y) = -h_l(x, y)$, while if it lies outside all the contours, we set $f_l(x, y) = h_l(x, y)$.

If we allow contours within contours, as in Figure 2, it is a trivial task to extend f_l accordingly. We count the number of intersections of any straight line segment running

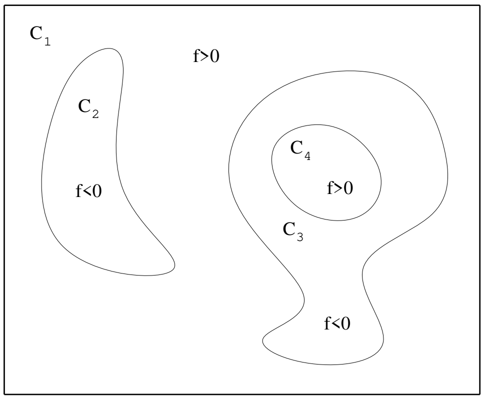

Fig. 2. The connected components of D in one slice.

from (x, y) to any point lying outside D. If this number is even $(0, 2, 4, \ldots)$, we define $\mathrm{sgn}_l(x, y) = 1$, and if odd, $\mathrm{sgn}_l(x, y) = -1$. We then set

$$f_l = \mathrm{sgn}_l \; h_l.$$

A piecewise biharmonic function. The signed distance function satisfies (a) and is continuous but is not in general differentiable. We now propose an alternative function whose zeros have the same topology, yet is C^1.

We require each f_l to be zero on contours and to change sign there. The signed distance function has the properties $f_l \mid (\gamma_{m,l}) = 0$ and $\frac{\partial f_l}{\partial n} \mid (\gamma_{m,l}) = \pm 1$. This suggests boundary conditions for some partial differential equation to be solved in each region bounded by contours. The contours $\{\gamma_{m,l}\}_m$ decompose D into a set of connected components $\{C_r\}$, each having one or more contours as its boundary; see Figure 2. Part of the boundary of the outer component, is ∂D. We may assign a sign to each C_r, setting $\mathrm{sgn}(C_r) = \mathrm{sgn}_l \mid C_r$.

We propose that a natural candidate for the differential equation is the biharmonic equation. Thus we define f_l piecewise. We define f_l, in each inner component C_r, to be the solution of

$$\Delta\Delta u = 0, \qquad u\mid_{\partial C_r} = 0, \qquad \frac{\partial u}{\partial n}\mid_{\partial C_r} = \mathrm{sgn}(C_r), \tag{1}$$

where $\frac{\partial u}{\partial n}$ refers to the inward normal derivative of u. In the special case of the outer component (C_1 in Figure 2) we solve the same equation but set the boundary conditions on the outer boundary ∂D to be

$$u\mid_{\partial D} = g_l\mid_{\partial D}, \qquad \frac{\partial u}{\partial n}\mid_{\partial D} = \frac{\partial g_l}{\partial n}\mid_{\partial D},$$

where g_l is the signed distance function defined earlier. Figure 3 shows the signed distance function and the piecewise biharmonic function for a slice containing four contours.

It is well-known that if ∂C_r is C^1, see for example Garabedian [7], that (1) has a unique C^1 solution inside and on the boundary C_r. Since f_l clearly has continuous derivative across contours it follows that f_l is C^1. Moreover, since u minimises the energy functional

$$\int_{C_r} u_{xx}^2 + 2u_{xy}^2 + u_{yy}^2$$

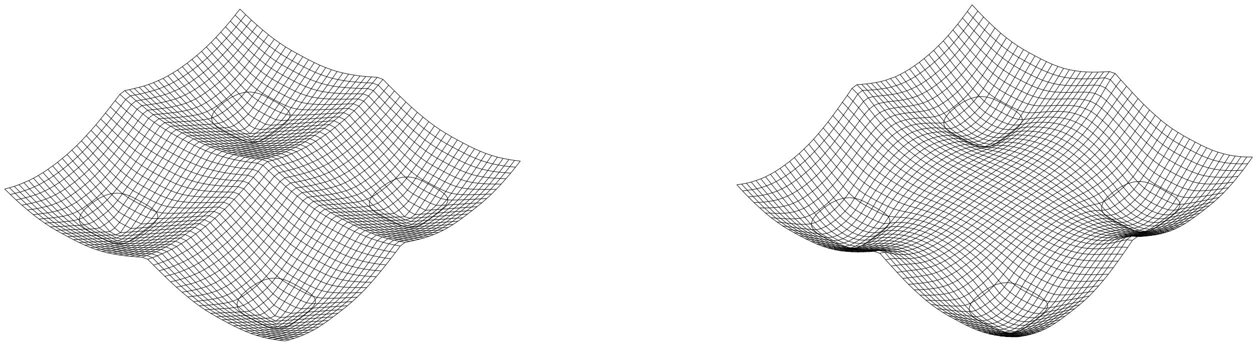

Fig. 3. The signed distance g_l and piecewise biharmonic f_l.

it is reasonable to expect that u has no zeros inside C_r and hence that the only zeros of f_l are the contours. Physically, setting the normal gradient to unity around the boundary is equivalent to clamping a thin plate around its rim at a constant angle of 45°. The biharmonic occurs frequently in the literature on partial differential equations [7], [19] and elasticity [14], but it appears to be an open question whether u is single signed. The best known maximum principle for (1) is the maximum modulus principle due to Miranda [13]. This bounds the size of $|u|$ in terms of the boundary conditions but does not unfortunately bound u itself. We have run many numerical examples on regions of many different shapes, and have never found that u had internal zeros.

Constructing f from the f_l's. In order to construct f from the f_l's we can, for example, use a C^1 quadratic or C^2 cubic spline interpolation [4]. By spline we mean a piecewise polynomial. In the cubic case one would define the knot vector $\mathbf{t} = t_1, \ldots, t_{n+6}$ to correspond with the z values of the slices and have multiplicity four at the ends:

$$s_1, s_1, s_1, s_1, s_2, \ldots, s_{n-1}, s_n, s_n, s_n, s_n.$$

For each (x, y), we then find the unique coefficients $a(x, y)$ such that

$$f(x, y, z) = \sum_{r=1}^{n+2} N_{r,4,\mathbf{t}}(z) a(x, y), \tag{2}$$

satisfies the interpolation conditions

$$f(x, y, s_l) = f_l(x, y), \qquad l = 1, \ldots, n, \tag{3}$$

$$\frac{\partial^2 f}{\partial z^2}(x, y, s_1) = \frac{\partial^2 f}{\partial z^2}(x, y, s_n) = 0. \tag{4}$$

Here $N_{r,4,\mathbf{t}}$ is the B-spline of order four whose support is $[t_r, t_{r+4}]$. The boundary conditions (4) can be replaced by Bessel end conditions [4] or other conditions in the literature. The qualitative result will be somewhat similar in each case. However it is important that they depend smoothly on the data $f_l(x, y)$, $l = 1, \ldots, n$ so that f will be smooth in (x, y). In the case of the C^1 quadratic spline interplant, it is usual to use the 'not-a-knot' placement of knots.

After experimentation it was found that pure spline interpolation can produce unwanted oscillations and great improvements were found when replacing it with a shape preserving interpolation. We implemented a method for constructing a C^1 piecewise cubic where one is free to choose the derivative at each point. Good estimates for the derivatives which preserve monotonicity are described by Foley [5]. The derivatives are initially set to those presented by Gregory and Delbourgo [8] for piecewise rational quadratics. These are then adjusted if necessary in order to ensure monotonicity.

In Section 4 a comparison is made of pure spline interpolation and monotonicity preserving interpolation.

The smoothness of f. Following the procedures above, using the piecewise biharmonic in (x, y), and at least a C^1 spline interpolation in z, f will be overall C^1. By using the higher order equation

$$\Delta^{k+1} u = \nabla^{2(k+1)} u = 0,$$

in place of (1) and adding zero boundary conditions on higher derivatives,

$$\frac{\partial^i u}{\partial n^i} |_{\partial C_r} = 0, \qquad i = 2, \ldots, k,$$

it would in principle be possible to construct f_l having C^k continuity, provided the contours were also C^k. And by using a C^k, spline interpolation in z, f would be C^k. Yet in practice this may be difficult to implement and require unnecessary amounts of computation. Moreover there is still no absolute guarantee that each u has no internal zeros and it is not clear that the surface $f = 0$ will be any better behaved.

§3. Numerical implementation

The zero set S of f is a surface approximation to the contour data. S can be approximated by a triangulation by evaluating f at every node, $\mathbf{p}_{i,j,k} = (x_i, y_j, z_k)$, of a sufficiently fine three-dimensional grid in Ω. Such a grid divides Ω into cells, each having eight of the nodes as corners.

There is a large literature on different methods for approximating the zeros of f within each cell [12,20]. We implemented a method which decomposes each cell into five tetrahedra and then approximate f by a linear function in each tetrahedron. The zeros of a trivariate linear function form a plane and so the zeros in each tetrahedron form either a triangular facet or a quadrilateral which can be divided into two triangular facets. The entire set of facets join together to form a triangulation which approximates S.

In order to reduce the number of long thin triangles, it is advantageous to chose the grid so that the cells are cubes. Thus we only choose the number of cells in the z direction between the planes $z = s_1$ and $z = s_n$. The numbers of cells n_x and n_y in the x and y directions respectively are then determined by the size of Ω and the requirement that all cells have equal dimensions.

Computing f. Since we only require the values of f at the nodes of the grid, we only need to compute the values of f_l within plane $z = s_l$ at the nodes of the corresponding two-dimensional grid in D, consisting of the nodes $\mathbf{q}_{i,j} = (x_i, y_j)$. If f_l is the signed distance function $f_l(\mathbf{q}_{i,j})$ can be computed individually for each (i, j). In the case of the piecewise biharmonic function, we solve (1) numerically on the grid using a finite difference scheme. The standard discretization of the biharmonic operator is the mask

$$\begin{array}{ccccc} & & 1 & & \\ & 2 & -8 & 2 & \\ 1 & -8 & 20 & -8 & 1\,. \\ & 2 & -8 & 2 & \\ & & 1 & & \end{array}$$

In order to approximate the boundary conditions for u along each contour one can fix the values of all nodes which are adjacent to any of the contours to the value of the signed distance function. The outer two layers of nodes along the rectangular boundary of D are also fixed equal to the value of the signed distance function. Then the discrete operator is applied to every remaining free node simultaneously.

We solved the corresponding discretized equations by the conjugate-gradient method since the operator is symmetric and positive-definite. Positive-definiteness is ensured since each line segment of each contour guarantees that the four neighbouring nodes are incorporated in the boundary conditions. An excellent start vector is the signed distance function itself. A similar linear system was solved in [1] for approximating scattered data.

The spline interpolation through each column of points $f_l(\mathbf{q}_{i,j})$ for $l = 1, \ldots, r$, can be carried out for each (x_i, y_j) in turn but the spline coefficients need not stored. The spline is immediately evaluated at each z_k, yielding the values of f in the column of nodes (x_i, y_j, z_k) and is afterwards discarded.

Making use of the normals. Provided f is C^1, we may differentiate it to obtain the normal direction to the surface $f = 0$, at any point, given by ∇f. This means that we can get a much improved shaded image of the triangulation, using for example, Gouraud or

Phong shading [6]. Taking the normals directly from the given analytic function, as in [3], is known to give much better results than averaging the normals of neighbouring triangular facets.

In practice we only have the values of f on a grid in each slice. We therefore need to interpolate the values of f with some smooth function to be able to estimate ∇f. The solution is to make a (trivariate) spline approximation s to f. This means in fact that even if f is not itself sufficiently smooth, as when using the signed distance function, s will be.

For example, one can choose to define s to be C^2 tricubic, interpolating the values of the $f_l(\mathbf{q}_{i,j})$ using the natural boundary conditions (second derivative equal to 0) in all three variables. However it is again preferable to use some shape preservation and we have implemented the monotonicity preserving C^1 cubic scheme, described earlier, in the z direction in the definition of s.

At each of the three corners $\mathbf{k}_0$, $\mathbf{k}_1$, $\mathbf{k}_2$, of each triangle generated, the normal direction, ∇f to $f = 0$ at a point close by, is approximately ∇s. Thus we generate $\nabla s(\mathbf{k}_0)$, $\nabla s(\mathbf{k}_1)$, $\nabla s(\mathbf{k}_2)$, for the corner normals. An example of Gouraud shading using the corners and normals is given in Figure 5b.

Possible improvements.

(1) A drawback of our signed distance function (and the smoother piecewise biharmonic) is that it only measures distance in 2D within each (x, y) plane. If $f_l(x, y)$ was taken to be the signed distance in 3D from the point (x, y, s_l) to the surface of the object we would expect the examples in Section 4 to be even smoother. In particular when the surface normal is close to the z axis the signed distance in 2D is a poor approximation to the real one. Unfortunately S is the surface we are trying to find!

 This suggests an iterative method. Let $f^0(x, y, z)$ be the initial function generated together with its zero surface S^0, using the 2D signed distance (and some fixed method for spline interpolation in z). Now recompute each f_l as the 3D signed distance to the surface S^0 and generate a new implicit function f^1 and surface S^1. One could continue to generate f^{k+1} from f^k until some measure of change is small. The surface S^k may already be good after only one or two iterations. Such a method would require a method for finding the nearest point on an implicitly defined surface and this, though time-consuming, could be done by Newton iteration.

(2) It would be interesting to find the zeros of s by subdividing it adaptively rather than using a uniform grid. This would quickly throw away data in regions where s has no zeros, using the variation diminishing property of splines. This would hopefully lead to few triangles in regions of low curvature and conversely many where the curvature is high. Adaptive subdivision is the standard technique in computer-aided geometric design for computing contours of bivariate splines. We have not implemented such an algorithm however. Each subdivision would increase the amount of storage space and connecting the triangles together on the way back up the subdivision tree could be complicated.

(3) Another possibility, if one could make a piecewise quadratic spline approximation to f, would be to use the fact that the roots of a quadratic polynomial are easy to compute. This fact was used to advantage by Powell [15] to generate contours of bivariate piecewise quadratics defined from scattered data. The surface S could then be approximated by quadric patches instead of triangular facets.

(4) Provided the trivariate spline s is smooth enough, higher derivatives of s can be evaluated at each triangle vertex. In particular, the second partial derivatives provided information about curvature.

A number of schemes are known for constructing polynomial patches to each triangle retaining some order of geometric continuity overall.

The cubical lattice. In [10] the number of cells in the z direction had to be $n-1$, so that the upper and lower faces of each cell lie in two consecutive planes. Moreover the distance $s_l - s_{l-1}$ between consecutive planes was assumed to be constant. By using spline interpolation these are no longer restrictions. If there are relatively few planes, the number of cells in the z direction can be chosen to be a lot greater than $n-1$. Each plane may also lie strictly between the upper and lower faces of some layer of cells. The greater the number of cells, the closer the triangulation will be to the tangent continuous surface S.

§4. Numerical examples

Figure 4 shows two examples of contour stacks. In Figure 5 the first of these was generated. Figure 5a shows the triangulation generated using the signed distance function in each slice and monotonicity preserving C^1 piecwise cubic interpolation in z. The number of cells used was $33 \times 33 \times 99$. Figure 5b is a shaded image of the same triangulation. The normals used in the Gouraud shading were generated by evaluating the first derivatives of a corresponding monotonicity preserving C^1 trivariate spline approximation to f.

Figures 5c and 5d show the effect of replacing the signed distance by the piecewise biharmonic. The ridges which are visible Figure 5b disappear in Figure 5d, reflecting the greater smoothness of the latter function.

Figures 6a, 6b, and 6c show the triangulation produced from the second contour stack, using respectively C^0 linear, C^1 quadratic spline, and C^1 monotonicity preserving piecewise cubic interpolation in the z direction. Figure 6d shows a shaded image of Figure 6c. The piecewise biharmonic function was used throughout in Figure 6 and the numer of cells was $40 \times 10 \times 60$.

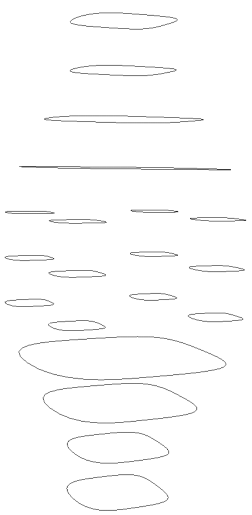

Fig. 4a.

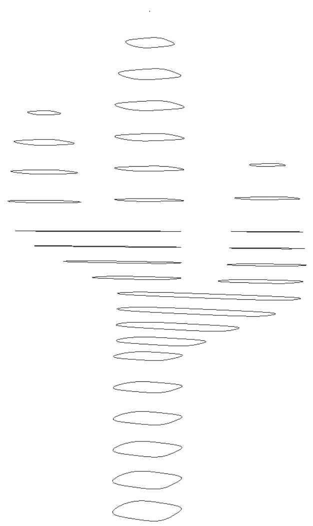

Figure 4b.

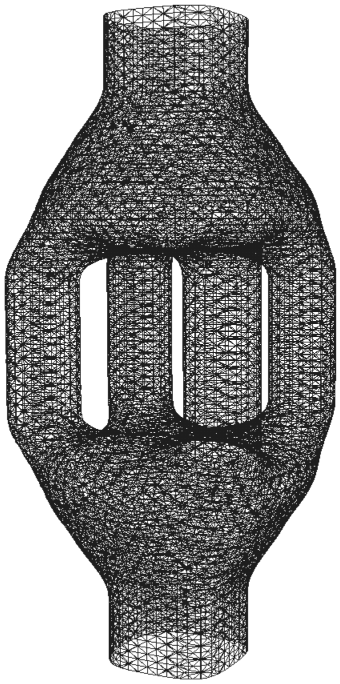

Fig. 5a.

Figure 5b.

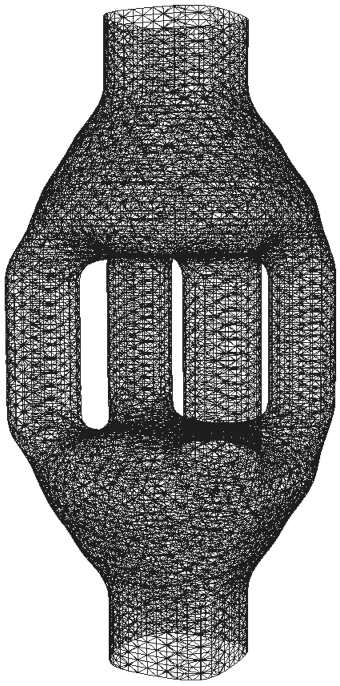

Fig. 5c.

Figure 5d.

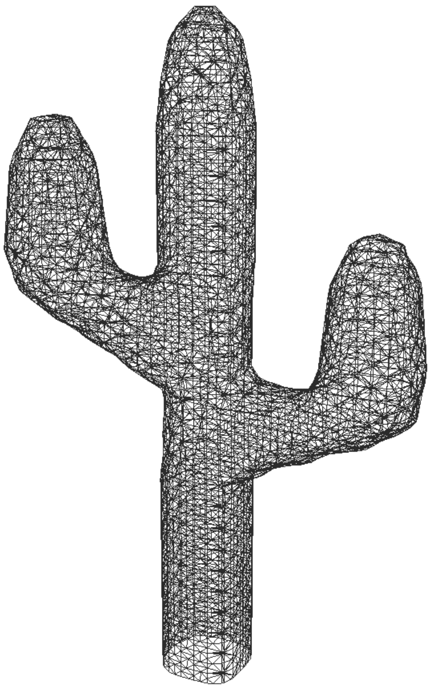

Fig. 6a.

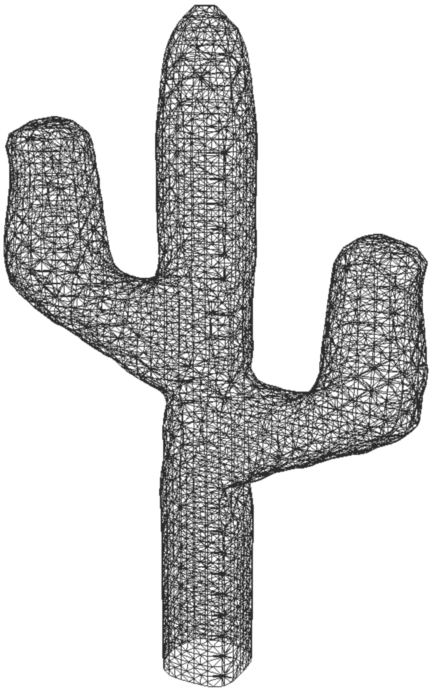

Figure 6b.

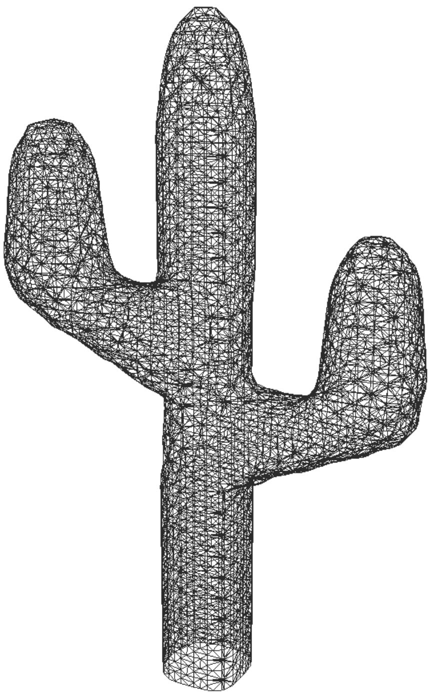

Fig. 6c.

Figure 6d.

§5. References

1. Arge E., M. S. Floater, Approximating scattered data with discontinuities, Numerical Algorithms **8** (1994), 149–166.
2. Choi Y., K. H. Park, A heuristic triangulation algorithm for multiple planar contours using an extended double branching procedure, The Visual Computer **10** (1994), 372–387.
3. Dahmen W., T. Thamm-Schaar, Cubicoids: modelling and visualization, Computer-Aided Geom. Design **10** (1993), 89–108.
4. de Boor, C., *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.
5. Foley T. A., A shape preserving interpolant with tension controls, Computer-Aided Geom. Design **5** (1988), 105–118.
6. Foley, van Dam, Feiner, Hughes, *Computer graphics: principles and practice*, Addison-Wesley, USA.
7. Garabedian P. R., *Partial differential equations*, John Wiley, New York, 1964.
8. Gregory J. A., R. Delbourgo, Piecewise rational quadratic interpolation to monotonic data, IMA Journal of Numerical Analysis **2** (1982), 123–130.
9. Hoppe H., T. DeRose, T. DuChamp, J. McDonald, W. Stuetzle, Surface reconstruction from unorganized points, Computer Graphics 2 **26** (1992), 71–78.
10. Jones M. W., M. Chen, A new approach to the construction of surfaces from contour data, Computer Graphics forum 3 **13** (1994), 75–84.
11. Levin, D., Multidimensional reconstruction by set-valued approximation, IMA J. Numerical Analysis 3 **6** (1986), 173–184.
12. Lorenson W. E., H. E. Cline, Marching cubes: a high resolution 3D surface construction algorithm, Proc. SIGGRAPH '87, Computer Graphics 4 **21** (1987), 163–170.
13. Miranda C., Formule de maggiorazione e teorema di esistenza per le funzioni biarmoniche di due variabili, Giornale di Mat. di Battaglini **75** (1948–1949).
14. Muskhelishvili, N. I., *Some basic problems of the mathematical theory of elasticity*, Noordhoff, Holland, 1963.
15. Powell, M. J. D., Piecewise quadratic surface fitting for contour plotting in *Software for Numerical Mathematics, D. J. Evans (ed.) Academic Press, London* 1974.
16. Rudin W., *Principles of mathematical analysis*, McGraw-Hill, New York, 1976.
17. Schumaker L. L., Reconstructing 3D objects from cross-sections, in *Computation of Curves and Surfaces, W. Dahmen, M. Gasca, and C. Micchelli eds., Kluwer, Dordrecht* (1990), 275–309.
18. Schumaker L. L., Triangulations in CAGD, IEEE Computer Graphics and Applications (1993), 47–52.
19. Villaggio P., *Qualitative methods in elasticity*, Noordhoff, Holland, 1977.
20. Wyvill G., C. McPheeters, B. Wyvill, Data structures for soft objects, The Visual Computer 2 **4** (1986), 227–234.