Importing and reshaping digitized data for use in rapid prototyping: a system for sculpting polygonal mesh surfaces

David G. Alciatore and Terry T. Wohlers

The authors

David G. Alciatore is based at the Department of Mechanical Engineering, Colorado State University, Fort Collins, Colorado, USA.

Terry T. Wohlers is with Wohlers Associates, Fort Collins, Colorado, USA.

Abstract

Focuses on the development and testing of software for reading and formatting digitized data and exporting it to rapid prototyping (RP). Research and development over two years has involved the implementation of special computer-aided sculpting software that runs on UNIX workstations and which imports 3D polygonal mesh data in STL, OBJ and DXF formats, then re-shapes it, much like the pushing and pulling on the surface of a rubber membrane. Specifying a wall thickness gives the model volume, prior to exporting an STL file. Describes how data has been imported form laser digitizing systems, had its shape changed and then how RP parts have been created from the model data.

Rapid Prototyping Journal
Volume 2 · Number 1 · 1996 · pp. 13–23

Introduction

The following presents a methodology for interactive computer-aided editing and free-form sculpting of 3D polygonal mesh surface models. It differs from other free-form surface methods because it manipulates the polygonal mesh data directly without using an alternative surface representation. The computer-aided sculpting software serves as a powerful reverse engineering and artistic editing tool. The process involves scanning an object with a 3D digitizer, manipulating it with the sculpting software, and then reproducing it with an RP system for possible subsequent processing (e.g. mould making).

Free-form computer-aided sculpting of a polygonal mesh surface involves modifying the wireframe vertex, edge, and facet geometry in useful and intuitive ways. Modifications could include adding to or removing from portions of the surface, free-form stretching of the surface, smoothing, and others. We will refer to the methodology presented in this article as *sculpting* since it provides a means of sculpting and reshaping triangle surface meshes common in surface digitizing and rapid prototyping applications where the models are usually stored in STL[1] or other polygonal mesh formats (e.g. DXF, OBJ).

Potential applications

Applications of computer-aided sculpting include:

- *Reverse engineering*. This involves the digitization of physical parts such as models or patterns for tooling. Sculpting serves as a rapid reverse engineering tool which can be applied to surfaces scanned with 3D digitizing systems[2,3]. Once the surface is scanned in, it can be quickly modified and a physical prototype of the new design can be produced with a rapid prototyping process such as stereolithography[4].
- *Surgical applications*. Surgeons can use the sculpting software to predict and plan the results of plastic surgery[5]. A patient requiring reconstructive face surgery can be scanned and the surgeon can manipulate the model to show the patient

The authors thank David Addleman, president of Cyberware, Monterey, California, USA, for providing interesting 3D digitized data files for this work.

Rapid Prototyping Journal
Volume 2 · Number 1 · 1996 · 13–23

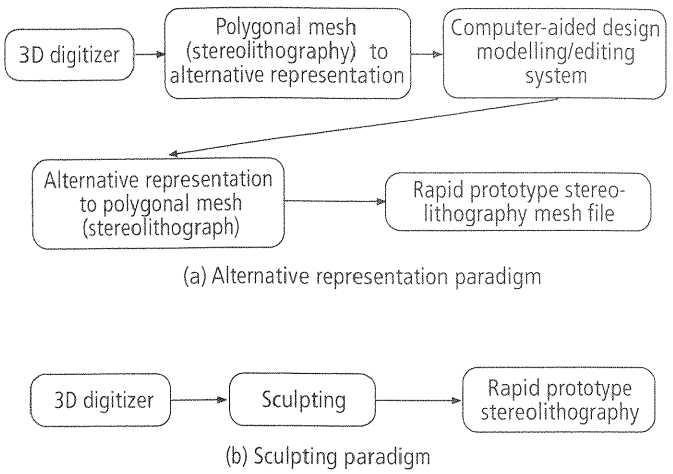
what they might look like after surgery. Computer-aided sculpting can also be used to reshape digitized data of bone structures for reconstructive surgery and implants.

- *Animation*. It is possible to customize digitized objects or characters for computer graphics animation. A complex free-form object, such as a clay sculpture of a cartoon character or plastic doll, can be digitized and reshaped. The resulting surface model can then be used in an animation package (e.g. Alias/Wavefront, SoftImage, or 3D Studio).
- *Generating caricatures of scanned faces*. A person's face could be scanned and then deformed in flattering or unflattering ways in 3D. With the addition of recently developed 3D painting tools[6,7], which allow the user to apply realistic textures and colours to the 3D models, the artistic possibilities are endless.
- *Editing 3D models created from serial section contours*. For example, at Colorado State University, the Vesalius project[8,9] and the Glaxo virtual anatomy project[10] use the sculpting implementation described later in this article to edit human anatomy surface models created from MRIs, CTs, and cadaver sections[11]. Sometimes the data acquisition and surface triangulation processes result in imperfections such as bumps, valleys, or roughness where there should be none. Sculpting allows a trained anatomist to intuitively modify the models making them anatomically correct.

Approaches to the solution

Figure 1 illustrates two paradigms for computer-aided sculpting and reproduction. The first, more traditional paradigm, which is the topic of much research, involves applying free-form manipulations to a nonpolygonal mesh surface representation (e.g. NURBS[12]) instead of using digitized mesh data directly[13-21]. In the defence of these works, some were not developed with the goal of accepting and delivering polygonal mesh data for rapid prototyping applications. The benefit of an alternative representation, such as NURBS, is that it may be easier to import and edit using some commercially available CAD and graphics packages. However, there are several disadvantages when applying this approach to digitized mesh data:

Figure 1 Computer-aided sculpting and reproduction paradigms



- Sometimes there are difficulties and many steps involved in converting from a polygonal mesh to an alternative representation. Also, the representations and the mathematics associated with the methods can be extremely complex.
- The original scan data are lost when the representation is changed.
- There are more steps in the process in going from the digitized polygonal mesh to the final sculpted rapid prototype part.

There is some work in the literature where triangulated surfaces are used to represent free-form deformations[19,20]; but even with these, shape changes are applied to some other representation (e.g. fairing functionals with character lines and curves[21], or physical bubble models applied to feature boundaries[19]) and only the resulting geometry is generated in polygonal mesh form.

Our solution to the problem

The methods presented in this article overcome all of the disadvantages described in the previous section by providing intuitive and simple tools for a user to perform free-form sculpting of the original mesh data without having to convert between other representations. There are a few commercially available software packages with mesh editing and modelling capabilities (DataSculpt from Laser Design (Minneapolis, Minnesota); ARKGeometry from Triple iii (Culver City, California); SCULPT4D from Byte by Byte (Austin, Texas); and Surfacer from Imageware (Ann Arbor, Michigan)], but their free-form sculpting and editing features are limited. A recently

Rapid Prototyping Journal
Volume 2 · Number 1 · 1996 · 13–23

developed package, Pixel Puddy from The Valis Group (Tiburon, California), has good free-form sculpting capabilities which use a physically based model of the mesh allowing the user to pull on points on the model with a natural effect, but the user is limited to and constrained by the modelled physics.

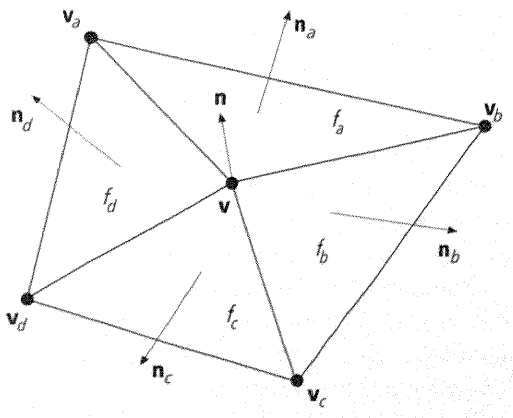
Presented below are key techniques used to implement the method. They include the data structures used to store and modify the model information; the free-form stretching algorithm; resolution enhancement and decimation techniques; and an algorithm for determining surface geodesics, which allows a user to intuitively and efficiently draw 3D shapes directly on the surface of the model.

Data structures

For the methods presented below, the polygonal mesh surface can consist of a variety of polygon types, can be an open bounded surface or a closed surface representing a solid boundary representation (B-REP), and can have holes. To simplify the discussion, triangle meshes are assumed.

Figure 2 shows a portion of a polygonal mesh with corresponding notation. In the data structure used to represent the mesh, each vertex is aware of its neighbour vertices and facets, and each facet is aware of its constituent vertices. The structure is very efficient for this application and much less cumbersome than other more general structures[22], such as the winged edge data structure. The model vertices and facets are stored in separate linked lists with the following information:

Figure 2 Portion of polygonal mesh with notation



Vertex

$\mathbf{v} = (x, y, z)$ vertex position vector.

$\mathbf{n} = (n_x, n_y, n_z)$ vertex outward unit normal vector (= average of neighbour facet normals).

$\mathbf{v}_a, \mathbf{v}_b, \mathbf{v}_c, \ldots$ unordered linked list of neighbour vertex pointers (e.g. $\mathbf{v}_a$, $\mathbf{v}_b$, $\mathbf{v}_c$, and $\mathbf{v}_d$ for $\mathbf{v}$ in Figure 2).

$f_a, f_b, f_c, \ldots$ unordered linked list of neighbour facet pointers (e.g. f_a, f_b, f_c, and f_d for $\mathbf{v}$ in Figure 2).

next pointer to next vertex in the list.

Facet

$\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ counter-clockwise ordered list of vertex pointers for the facet (e.g., $\mathbf{v}_a$, $\mathbf{v}$, $\mathbf{v}_b$ for f_a in Figure 2).

$\mathbf{n} = (n_x, n_y, n_z)$ facet unit outward normal vector implied by the vertices.

next pointer to next facet in the list.

Unlike other representations[22], the edges between vertices are not stored directly in a separate list but they are implied through the vertex neighbour lists. As will be clear in the descriptions of some of the sculpting algorithms below, the data structure is sufficient in providing efficient implementations.

Free-form stretching of the mesh

Allowing the user to easily select custom areas directly on the 3D surface for manipulation is a key feature required to implement the methods below. The first step in acquiring a custom area is having the user define a closed perimeter on the surface. To avoid having the user indicate connected points manually, surface geodesics are automatically determined between user-defined points. (See the section titled "Surface geodesic algorithm".) The user indicates just enough vertices, with clicks of the mouse button, to guide the shape of the perimeter, and the geodesic algorithm fills the gaps between them. The cursor position (at mouse button click) is used to identify the vertex by mapping the cursor screen co-ordinates into a world co-ordinate ray which is processed by an efficient ray-triangle intersection routine[23] to choose the selected facet. Once the facet and intersection point are identified, the closest vertex is determined easily[23].

Once the perimeter of the custom shape is defined, the set of interior vertices must be

Rapid Prototyping Journal
Volume 2 · Number 1 · 1996 · 13–23

determined. The algorithm used for this is a recursive fill using one interior vertex as a seed. An interior vertex seed is found by first ensuring the perimeter is ordered counterclockwise by checking the sum of the projected turning angles around the perimeter using:

$$\sum_{i=1}^{N} \sin^{-1}\left(\frac{\|\mathbf{a}_i \times \mathbf{b}_i\|}{\|a_i\| \, \|\mathbf{b}_i\|}\right)$$

where $\mathbf{a}_i$ and $\mathbf{b}_i$ are the adjacent edge vectors ($\overline{\mathbf{v}_{i-1}\mathbf{v}_i}$ and $\overline{\mathbf{v}_i\mathbf{v}_{i+1}}$) projected into the plane perpendicular to the ith vertex normal ($\mathbf{n}_i$):

$$\mathbf{a}_i = (\mathbf{v}_i - \mathbf{v}_{i-1}) - [(\mathbf{v}_i - \mathbf{v}_{i-1})\cdot\mathbf{n}_i]\mathbf{n}_i$$

$$\mathbf{b}_i = (\mathbf{v}_{i+1} - \mathbf{v}_i) - [(\mathbf{v}_{i+1} - \mathbf{v}_i)\cdot\mathbf{n}_i]\mathbf{n}_i.$$

The summation is positive for a counterclockwise perimeter. In calculating the summation, a check must be performed to make sure $\mathbf{a}_i$ and $\mathbf{b}_i$ are not collinear. In this case, the three projected vertices lie on a straight line and the cross product is undefined. If $\mathbf{a}_i$ and $\mathbf{b}_i$ are collinear (i.e. $\left|\mathbf{a}_i \cdot \mathbf{b}_i - \|a_i\| \, \|b_i\|\right| < \varepsilon$), the contribution to the turning angle summation is zero.

Once the perimeter is made counterclockwise, by reversing the order of the vertices if necessary, a seed interior vertex is found by choosing a neighbour vertex ($\mathbf{v}_n$) common to two adjacent perimeter vertices ($\mathbf{v}_a$ and $\mathbf{v}_b$) which is to the left of the perimeter edge, i.e.

$$[(\mathbf{v}_b - \mathbf{v}_a) \times (\mathbf{v}_n - \mathbf{v}_a)] \cdot \mathbf{n}_a > 0.$$

A recursive algorithm is used to branch out from this seed creating the desired list of interior area vertices. This is done by recursively adding to the list all neighbour vertices of the seed which are not on the perimeter and which are not on the area list already. To deal with a "bottleneck" problem which can occur when there is a single facet constriction in the perimeter, additional non-visited perimeter-adjacent seeds are searched for and the process is repeated until the area is complete (i.e. until there are no more non-visited seeds).

Once the area is defined, for each vertex within the area, the normalized degree of stretching, s, is determined from the following cubic relation (see Figure 3):

$$s = 2r^3 - 3r^2 + 1$$

where r is the normalized shortest distance from the vertex to the perimeter (0 implies it is on the perimeter, 1 implies it is furthest from the perimeter). The direction of stretching can be controlled by the user. Figure 4 illustrates the effects of a directional free-form cubic stretch in two dimensions.

Figure 3 Free-form cubic stretching profile

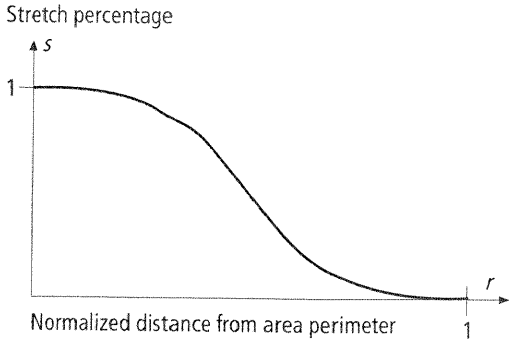
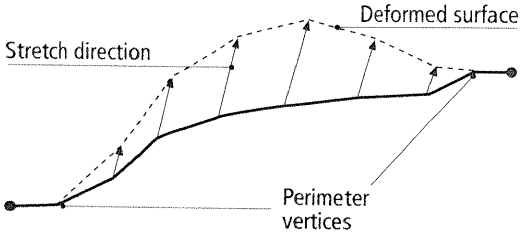


Figure 4 2D illustration of free-form cubic stretch



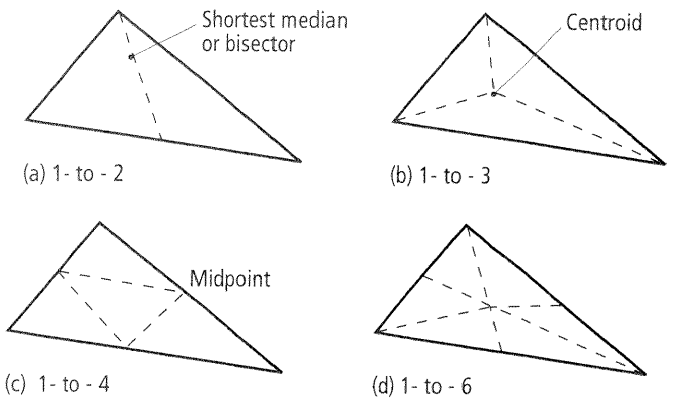
Other profiles could be chosen or interactively defined by the user to achieve a different effect (e.g. flat boss, chamfer, or fillet), but for free-form applications, the cubic profile provides an intuitive free-form stretched shape due to the tangency at the perimeter.

Resolution enhancement and decimation

If the user wants to stretch a portion of the surface where resolution is limited, the area's mesh must first be refined. Refinement (or resolution enhancement) divides each triangular facet into two or more facets as shown in Figure 5. The 1-to-3 and 1-to-6 refinements tend to result in long, obtuse triangles which may be undesirable. The 1-to-4 refinement is the best for sculpting applications since it subdivides symmetrically and since it helps reduce the amount of obtuseness when subdividing long, narrow triangles.

Refinement of the entire model is straightforward. The 1-to-4 refinement is applied to each facet while being careful to create edge midpoints only once, sharing them between neighbouring facets. When refining an area as

Figure 5 Mesh refinement strategies

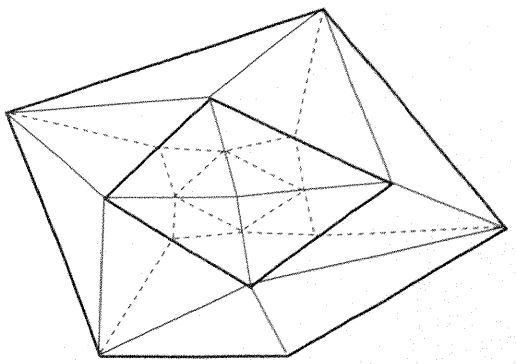


illustrated in Figure 6, extra care is required at the boundary or perimeter of the area. The area perimeter is shown dark and the refinement is shown as dashed lines. As shown, the edge midpoints created on the perimeter must be joined to adjacent vertices outside of the area creating two additional facets for each perimeter edge. This is required to prevent holes in the mesh topology.

Another useful tool in the sculpting process is mesh decimation. Here, the number of facets in parts of, or in the entire mesh, are reduced to decrease storage and I/O requirements and increase graphics performance. The vertex-based algorithm developed by Schroeder *et al.*[24] was used as the model for the software implementation described below. The algorithm works by successively checking the feature height of each vertex above the average plane through its neighbours. If this height is below a user-specified tolerance, the vertex is a candidate for decimation (removal) since it is not contributing very much to the shape (curvature) of the surface. The vertex and its neighbouring facets are removed and the resulting polygonal hole is re-triangulated in an optimal way[24] (see Figure 7). The decimation tool can be applied selectively, using the custom shape tool, or to the entire model. There are other decimation methods presented in the literature[25,26] others cited in[24], but the one chosen here executes quickly, preserves much of the original data, is relatively easy to implement, and produces good results.

The combination of the custom area stretching, refinement, and decimation tools is powerful. If the user desires to sculpt a low resolution area sparse in vertices, she/he can apply refinement to the degree desired, stretch the area, and decimate the result to remove data in low curvature regions.

Figure 6 1-to-4 refinement of an area



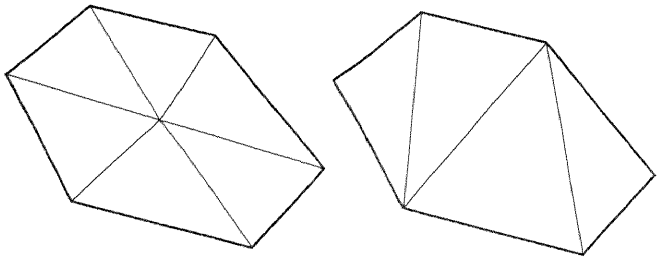
Surface geodesic algorithm

The surface geodesic algorithm is based on a common dynamic programming (DP) algorithm called the A* algorithm[27]. The A* algorithm, also called Dijkstra's algorithm[28], is useful in solving a large variety of graph search problems including motion and path planning for robots[29,30], automated pipe routeing[31], and other problems involving finding a path through a graph connecting two states while minimizing some metric – in this case, distance.

The algorithm is illustrated with the mesh shown in Figure 8. The goal is to find the geodesic, or shortest path, from vertex A (the start vertex) to vertex Z (the end vertex). The algorithm creates a dynamic graph through the vertices based on distance measurements and estimations. The algorithm is efficient because it searches only selected branch and path combinations between the start and end vertices. An exhaustive search of paths would be extremely computationally expensive and therefore is not viable.

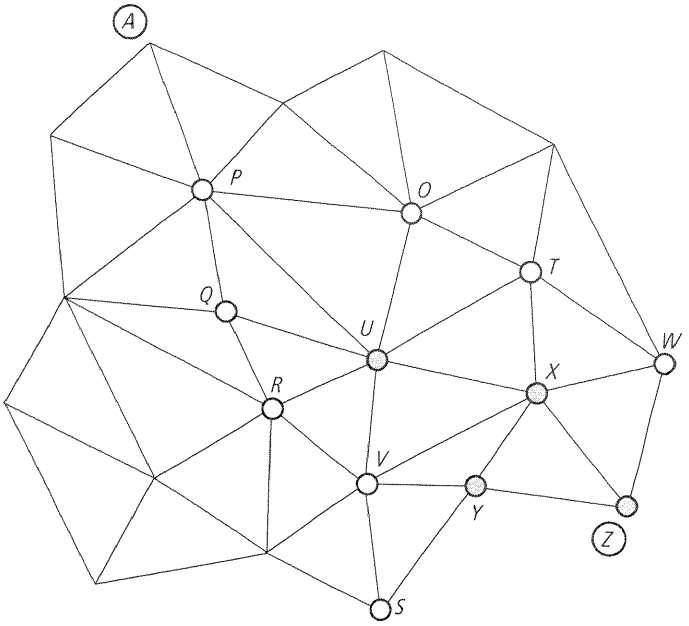
Two separate vertex linked lists are stored during the algorithm execution: an active vertex list and an inactive vertex list. Active list vertices (shown as circles in the Figure 8) are candidates for branching, and inactive list vertices (shown as filled circles in the Figure 8) have already been branched from and are candidates for the geodesic path. The following information is stored for each vertex on the active and inactive lists.

Figure 7 Vertex decimation



(a) Six facets before decimation (b) Four facets after decimation

Figure 8 Surface geodesic DP algorithm



Active or inactive list vertex

- **v**: pointer to vertex.
- *cumm_dist*: cumulative traversal distance (sum of edge lengths) from the end vertex.
- *parent*: pointer to the vertex from which this list vertex was branched (NULL for the end vertex).
- *next*: pointer to the next list vertex (NULL for the last list vertex).

The steps in the DP mesh surface geodesic algorithm are:

(1) *Begin*. Clear the active and inactive lists, add the end vertex (Z) to the active list. The reason for beginning with the end vertex will become clear in step 6.

(2) *Pick best candidate*. Choose the vertex on the active list with the minimum projected path distance between the start and end vertices according to:

proj_dist = cumm_dist + est_dist_remain

where est_dist_remain is the estimated distance between the active list vertex and the start vertex. This estimate must be a lower bound on the actual optimal distance to guarantee finding the optimal (minimum distance) geodesic[27]. There may be methods to better approximate this, but a simple Euclidean straight line distance between the vertices provides an inexpensive and adequate lower bound estimate. If the active list is empty, then there is no solution to the problem because there is no connected path between the start and end vertices.

(3) *Check for convergence*. If the selected active list vertex is the start vertex, the solution has been found. If so, go to step 6.

(4) *Branch out from the selected active vertex*. Add the selected active vertex to the inactive list and remove it from the active list. Add all neighbour vertices of the selected vertex to the active list provided they are not on the active or inactive lists already.

(5) *Iterate*. Go to step 2.

(6) *Retrieve the geodesic path*. Since the algorithm started at the end vertex, the geodesic path between the start and end vertices is easily obtained, in linked list form, by finding the recursive parents of inactive vertices beginning with the start vertex.

Referring to Figure 8, the algorithm begins with vertex Z, branches out to W, X, and Y, chooses X since it has the minimum projected distance, and branches out to T, U, and V. At this point, Y has the smallest projected distance of the current active vertices (T, U, V, W, Y), so S is added to the active list. The next selected active vertex is U, causing O, P, Q, and R to be added. The process continues until A is selected.

Software implementation

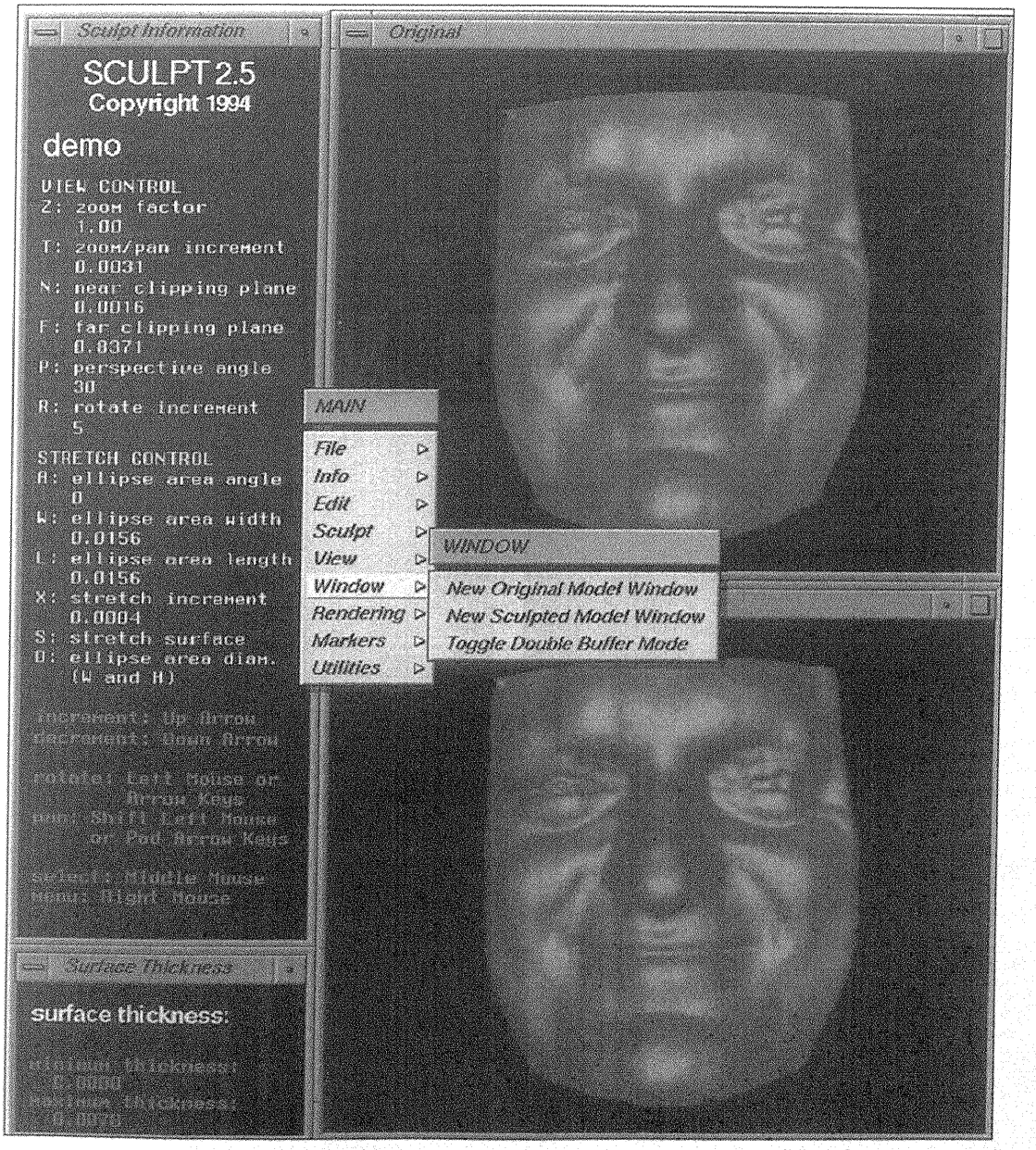
The methods presented in this article have been included in a computer-aided sculpting package consisting of 13,000 lines of C code. The program runs on Silicon Graphics workstations using the GL graphics library. Features of the software include I/O of various file

Rapid Prototyping Journal
Volume 2 · Number 1 · 1996 · 13–23

formats (DXF, OBJ, and STL), complete viewing control (zooming, perspective, rotation, etc.), several rendering options, surface editing tools, and surface sculpting tools. The editing tools include trimming, smoothing, roughening, hole patching, mesh decimation, and mesh refinement. Most of the tools can be applied to a user-defined area (custom shape), to an automatically generated mirrored area (for symmetric editing), or to the entire model. User areas are defined by placement of a sizeable and orientable elliptical shape, or by a custom polygon indicated by the user with the help of the surface geodesic algorithm. See "http://www.lance.colostate.edu/~dga/sculpt.html" for more information and a demonstration copy of the sculpting software.

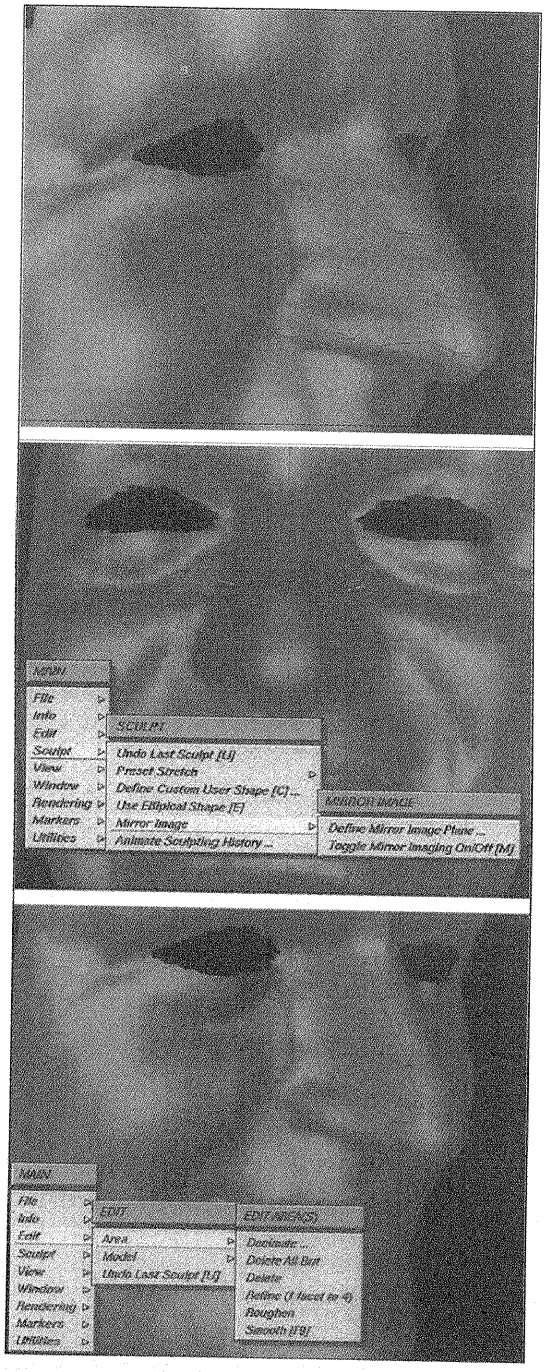
Plates 1 to 5 illustrate a typical editing session using the implemented sculpting software. Plate 1 shows the software's windows and an example data set acquired from a Cyberware laser scanner. Plate 2 shows the model after some trimming operations have been performed. Custom shapes were used to trim around the outside of the surface and to create the holes in the eyes. Plate 2a shows a custom shape being defined. Points (defined by mouse clicks) are indicated by line segments representing the selected vertex normals, and line segments representing neighbour edges. The line strings connecting the

Plate 1 Sculpting example with user interface and Cyberware data



Rapid Prototyping Journal
Volume 2 · Number 1 · 1996 · 13–23

Plate 2 Smoothing with mirrored custom geodesic-based user shape



points are determined and displayed automatically by the geodesic algorithm after each mouse click. Plate 2b shows the results of the automatic mirror imaging utility. Given a user-defined plane of symmetry, the program automatically generates an approximate mirror image of the custom area. In Plate 2c, smoothing is applied to both areas simultaneously. The smoothing algorithm simply replaces each vertex with the average of itself and its neighbours. If a vertex is on a surface boundary, it is replaced with the average of itself and its boundary neighbours only. This prevents outer boundaries from shrinking and inner boundaries (e.g. eye hole) from enlarging. The operation illustrated in Plate 2c is used to smooth out some of the artefacts (streaks) from the scanning process.

Plate 3 illustrates the elliptical area tool. In this case both stretching (cubic) and smoothing are used to eliminate a deep wrinkle and roughness. The elliptical cylinder is placed (by selecting its centre point), sized, and oriented by the user. The projected area on the surface is determined automatically by recursively selecting centre point neighbours which are contained within the elliptical cylinder. Plates 4a and 4b show the wireframe mesh before and after 30 per cent decimation applied to the entire model. Notice that the algorithm preserves detail in areas of high curvature. Plates 4c and 4d illustrate the surface extrusion utility, where the polygonal mesh surface is offset in a direction opposite to the surface normals, resulting in a back-facing surface which is stitched together to the front surface at its boundaries. This solid representation, once exported as an STL file, is required to manufacture a part using a rapid prototyping process. Plate 5 shows the

Plate 3 Elliptical area stretching and smoothing

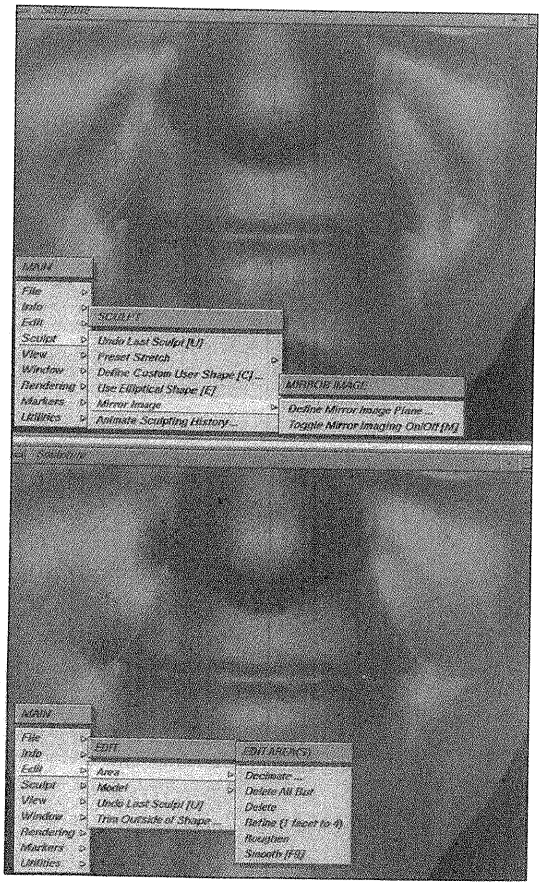
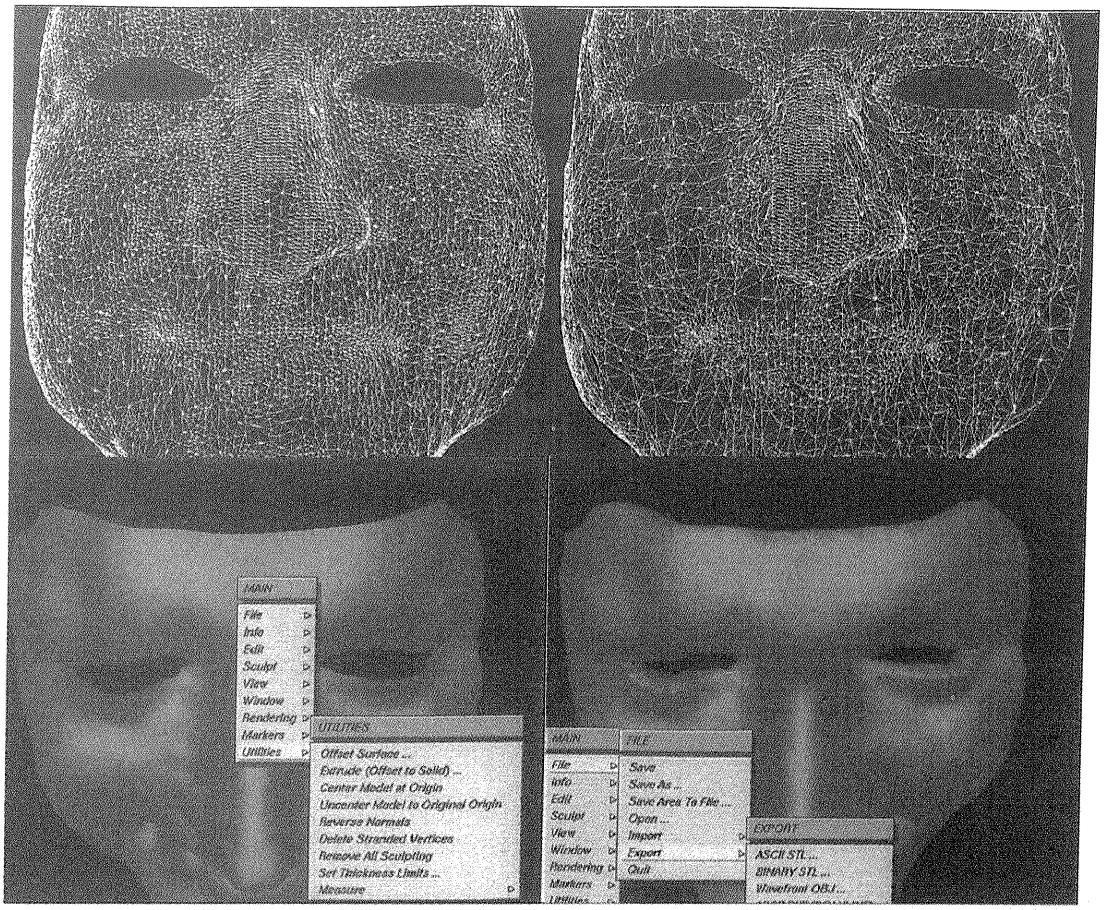


Plate 4 Decimation and surface extrusion



original trimmed surface and the final sculpted surface after several more sculpting (stretching) and smoothing operations. The appearance of the model has been changed dramatically with only a few editing and sculpting operations.

Summary

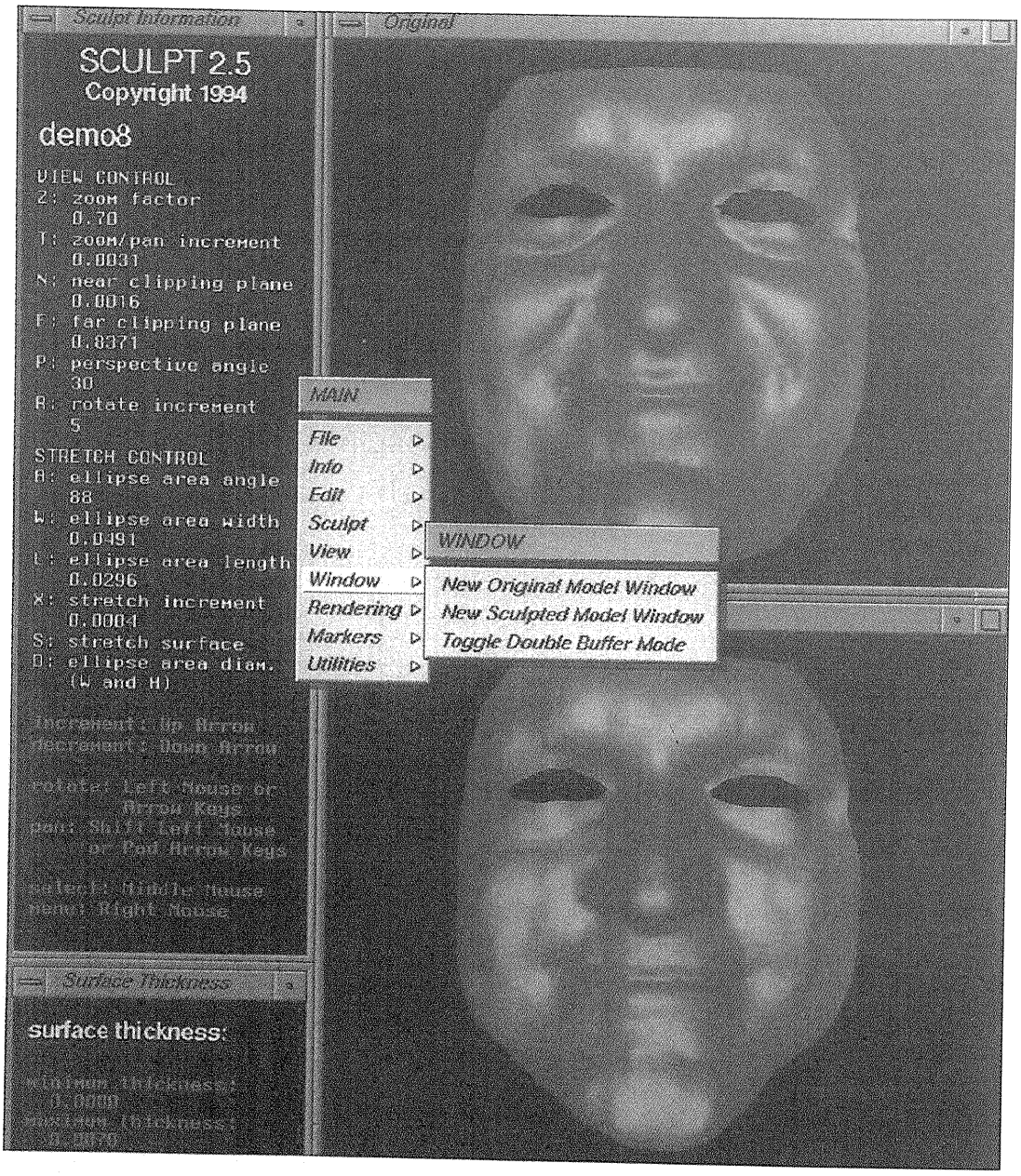
A methodology for direct interactive computer-aided free-form sculpting of 3D polygonal mesh surface models has been presented. Also presented were key techniques used to implement the method including an algorithm for determining surface geodesics, an algorithm for determining the interiors of a closed polygon, and the application of mesh refinement and decimation techniques. These methods differ from other free-form surface modeling and editing techniques because they manipulate polygonal mesh data directly. Also, the resulting sculpting implementation is relatively simple; and for the user, it is intuitive and easy to control. Polygonal mesh based computer-aided sculpting serves as a powerful reverse engineering and artistic editing tool, permitting an object to be scanned with a 3D digitizer, rapidly manipulated with sculpting, and then reproduced with a rapid prototyping system.

As demonstrated in the example, the methods can and have been applied and they are successful. With minimal practice, a typical editing session such as the example presented above, can easily be completed in a half hour or less on a reasonably fast machine (e.g. Indy XZ from Silicon Graphics). Sculpting has an abundance of applications including customizing digitized objects or characters for computer graphics animation; reverse engineering and design modification of digitized physical parts or models; planning and predicting the results of plastic surgery; generating caricatures of scanned faces; and editing 3D models created from serial section contours. There may be many more applications involving surface digitization and rapid prototyping where the methodology and implementation has potential.

Conclusions

The computer-aided sculpting approach presented here illustrates a straightforward

Plate 5 Original trimmed model with final sculpted version



method for interactively reshaping computer model data from 3D object digitizing systems. The techniques presented simplify the problem of making smooth-flowing changes to 3D model data. The surface remains as polygonal mesh data from the start to finish. The user can balance the trade off between model resolution and file size using the resolution enhancement and decimation tools. It is possible to give the model volume by specifying a wall thickness and then output a binary or ASCII STL file. With this file type, a user can manufacture a physical model using one of many RP systems currently on the market.

References

1 3D Systems (Valencia, CA), "StereoLithography interface specification", p/n 50065-S01-00, October 1989.

2 Wohlers, T., "3D digitizing systems", *Computer Graphics World*, April, 1994, pp. 59-61.

3 Wohlers, T., "3D digitizers", *Computer Graphics World*, July 1992, pp. 73-7.

4. Ashley, S., "Prototyping with advanced tools", *Mechanical Engineering*, Vol. 116 No. 6, June 1994, pp. 48-55.

5 Sadler, L., "Visualization and analysis of three dimensional shape change: surgical pre-planning, implant and prosthesis design and fabrication", *Proceedings of Annual International Conference of the IEEE*

Engineering in Medicine and Biology Society, Vol. 13 No. 3, 1991, pp. 1042-43.

6 Hanrahan, P. and Haeberli, P., "Direct WYSIWYG painting and texturing on 3D shapes", *Computer Graphics (ACM SIGGRAPH '90 Proceedings)*, Vol. 24, August 1990, pp. 215-23.

7 Robertson, B., "Fresh paint", *Computer Graphics World*, December 1994, pp. 28-37.

8 Alciatore, D. and Miranda, R., "Building three-dimensional computer graphics surface models for anatomy education", *Proceedings of the ASME Computers in Engineering Conference*, Anaheim, CA, August 1992.

9 McCracken, T. and Spurgeon, T., "The Vesalius project: interactive computers in anatomical instruction", *Journal of Biocommunications*, 1991, Vol. 18 No. 2.

10 Webster, J., "Abra cadaver: it's a virtual corpse", *Computer Graphics World*, p. 13, August, 1994, (http://www.vis.colostate.edu/library/gva/gva.html).

11 Spitzer, V. and Whitlock, D., "A volume database of human anatomy", *Advanced Imaging*, March 1989, pp. 48-9.

12 Tiller, W., "Rational B-splines for curve and surface representation", *Computer Vision, Graphics, and Image Procession*, 1985, pp. 36-43.

13 Celniker, G. and Gossard, D., "Deformable curve and surface finite-elements for free-form shape design", *Computer Graphics*, Vol. 25 No. 4, July 1991, pp. 257-66.

14 Chang, Y. and Rockwood, A., "A generalized de Casteljau approach to 3D free-form deformation", *Computer Graphics*, July 1994, pp. 257-60.

15 Esselens, D., "Surfacer: a complete software environment for reverse engineering, product design and analysis", *Proceedings of 2nd International Conference on Rapid Prototyping*, Paris, May 1993.

16 Hsu, W., Hughes, J. and Kaufman, H., "Direct manipulation of free-form deformations", *Computer Graphics*, Vol. 26 No. 2, July 1992, pp. 177-84.

17 Moreton, H. and Sequin, C., "Functional optimization for fair surface design", *Computer Graphics*, Vol. 26 No. 2, July, 1992, pp. 167-76.

18 Muraki, S., "Volumetric shape description of range data using 'blobby model'", *Computer Graphics*, Vol. 25 No. 4, July, 1991, pp. 227-35.

19 Himada, K., Balents, B. and Gossard, D., "Automatic mesh reconstruction for feature-based sculpting of deformable surfaces", *Geometric Modeling for Product Realization*, 1993, pp. 165-88.

20 Szeliski, R. and Tonnesen, D., "Surface modeling with oriented particle systems", *Computer Graphics*, Vol. 26 No. 2, July, 1992, pp. 185-94.

21 Welch, W. and Witkin, A., "Variational surface modeling", *Computer Graphics*, Vol. 26 No. 2, July, 1992, pp. 157-66.

22 Weiler, K., "Edge-based data structures for solid modeling in curved-surface environments", *IEEE Computer Graphics and Applications*, Vol. 5 No. 1, January 1985, pp. 21-40.

23 Haines, E., "Fast ray-convex polyhedron intersection", *Graphics Gems II*, Academic Press, Boston, MA, 1991, pp. 247-63.

24 Schroeder, W., Zarge, J. and Lorensen, W., "Decimation of triangle meshes", *Computer Graphics*, Vol. 26 No. 2, July, 1992, pp. 65-70.

25 Turk, G., "Re-tiling of polygonal surfaces", *Computer Graphics*, Vol. 26 No. 2, July 1992, pp. 175-84.

26 Harmann, B., "A data reduction scheme for triangulated surfaces", *Computer Aided Geometric Design*, Vol. 11, 1994, pp. 197-214.

27 Nilson, N., *Problem Solving Methods in Artificial Intelligence*, McGraw Hill, New York, NY, 1971.

28 Cormen, T., Leiserson, C. and Rivest, R., *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1990.

29 Alciatore, D., "Automation of a piping construction manipulator and development of a heuristic application-specific path planner", PhD dissertation, The University of Texas at Austin, Austin, TX, December 1989.

30 Barraquand, J. and Latombe, J., "Robot motion planning: a distributed representation approach", Stanford Research Report No. STAN-CS-89-1257, May, 1989.

31 Cleveland, A., personal correspondence, Bechtel Eastern Power Corporation, Gaithersburg, MD, March, 1988.

Further reading

Welch, W. and Witkin, A., "Free-form shape design using triangulated surfaces", *Computer Graphics*, July, 1994, pp. 247-56.