in: Weatherill et al., eds., Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, Pineridge Press Ltd, Swansea, U.K., 1994.

Interactive techniques for correcting CAD data

Brian A. JEAN [1] and Bernd HAMANN [2]

Summary - The paper presents a technique for correcting discontinuities (holes, overlapping surfaces, and intersecting surfaces) in CAD data. The technique approximates faulty geometries by several new B-spline surfaces which match properly. Each B-spline surface is constructed from four user-defined boundary curves.

1. Introduction

Commonly, surfaces created by a CAD system are not connected properly or intersect with each other. Surfaces must be connected continuously for further processing, *e.g.*, grid generation, flow simulation, and manufacturing. The surface correction technique described in this paper is implemented in the grid generation system currently being developed at the NSF Engineering Research Center for Computational Field Simulation at Mississippi State University ("National Grid Project").

Each B-spline surface of the global approximant is constructed from four user-specified boundary curves. These four curves define an initial local approximant, a Coons patch, which might lie on original surfaces or might only be "close" to them. The Coons patch is projected onto the given geometry, and the projections are interpolated by a B-spline surface. The geometric modeling methods used in this algorithm are described in [1], [2], [3], [4], [7], [8], [10], [11], and [14]. Grid generation is discussed in [6] and [13]. Creating a single, local B-spline approximant requires these steps:

(i) Specifying four boundary curves

(ii) Constructing a bilinear Coons patch from these four curves

(iii) Projecting the Coons patch onto the geometry

(iv) Performing scattered data approximation in case some projections can not be found

[1] Research Assistant I, NSF Engineering Research Center for Computational Field Simulation, Mississippi State University, P.O. Box 6176, Mississippi State, MS 39762, U.S.A., brian@erc.msstate.edu

[2] Assistant Professor of Computer Science, NSF Engineering Research Center for Computational Field Simulation/Dept. of Computer Science, Mississippi State University, P.O. Box 6176, Mississippi State, MS 39762, U.S.A., hamann@erc.msstate.edu

(v) Interpolating the data resulting from steps (iii) and (iv)
(vi) Estimating the error

Eventually, all single, local B-spline approximants are adjusted such that they define a globally continuous approximation of the entire geometry.

2. Creating and projecting the initial local approximant

The first step is the generation of four curves defining the boundary curves of a bilinearly blended Coons patch (see [3], [4], and [8]). These four curves can be curves on the original geometry or curves "close" to it. They are evaluated such that the resulting point distribution on each curve is nearly uniform with respect to arc length. The finite point sets on each curve are bilinearly blended defining a finite point set on the Coons patch.

The resulting points on the Coons patch are denoted by $\mathbf{x}_{I,J}$, $I, J = 0, ..., N$. The Coons patch is used as an initial local approximation of the given geometry, and the points on it must be relatively close to the given surfaces. In the next step, all points $\mathbf{x}_{I,J}$ are projected onto the original surfaces. For each point $\mathbf{x}_{I,J}$, a unit outward normal vector $\mathbf{n}_{I,J}$ is approximated using finite differences. Each point $\mathbf{x}_{I,J}$ is then projected onto the original surfaces by computing the intersection(s) of the line $\mathbf{L}_{I,J} = \mathbf{x}_{I,J} + d_{I,J}\mathbf{n}_{I,J}$, $d_{I,J} \in \mathbb{R}$, with a triangulation of the given surfaces. Provided that the original surfaces are defined parametrically, this triangulation is obtained by evaluating all parametric surfaces on rectilinear grids in their parameter spaces and splitting the resulting 3D quadrilaterals into pairs of triangles.

An intersection with the surface triangulation is considered only if its distance from $\mathbf{x}_{I,J}$ does not exceed a certain percentage of the length of the diagonal of a scaled bounding box of the Coons patch. If multiple intersections are found along a line $\mathbf{L}_{I,J}$, the one closest to $\mathbf{x}_{I,J}$ is chosen. Using the concept of barycentric coordinates, each intersection point is mapped onto the surface, provided its parametric definition is known. The points that have been mapped onto the given surfaces are denoted by $\mathbf{y}_{I,J}$. Fig. 1 illustrates the projection of the Coons patch onto the given geometry.

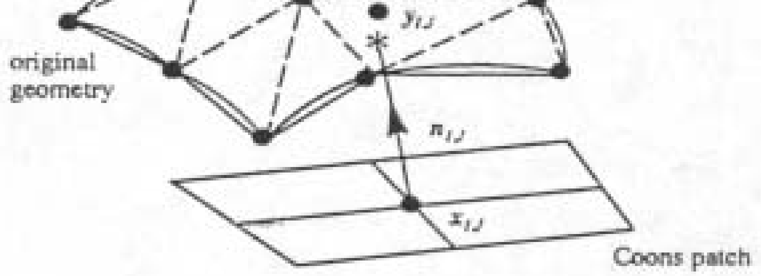

Fig. 1 : Projecting Coons patch onto original geometry.

3. The octree data structure

During the construction of the approximation, points on the Coons patch are projected onto the original set of surfaces. This is the most computationally expensive and time consuming step in the construction process. In order to accelerate this step, the set of triangles approximating the given surfaces is stored in an octree data structure. Octree structures have been used by a number of researchers for finite element mesh generation (see [10]) and for object representation (see [9]). The octree affords a substantial increase in execution time because the spatial information contained in the octree allows rapid culling of large numbers of triangles that need not to be considered for the projection.

An octree is a hierarchical data structure which results from successively subdividing a cube in each dimension to form eight regions or octants. The subdivision is continued until some stopping criterion is satisfied. Each node in the tree corresponds to some region of space bounded by the cube associated with that node. Since the tree is constructed in cartesian space, the faces of the cube are constant-x, constant-y, and constant-z planes. Therefore, only six coordinate extrema are stored. If a node is not a terminal or leaf node, then it has eight children or subcubes, the union of which completely fill the cube associated with the node. In the current implementation, the surface data associated with an octant is stored in the form of an array of pointers to surface triangles associated with the octant. An octree data structure with three levels is shown in Fig. 2. The actual data elements associated with every node in the tree are

- the integer identifier of the node,
- a pointer to the parent of the octant,
- minimum xyz coordinates among the cube vertices,
- maximum xyz coordinates among the cube vertices,
- pointers to the eight children of the octant,
- a list of pointers to triangles contained partially or completely in the octant (triangle list), and
- an integer indicating the number of triangles in the triangle list.

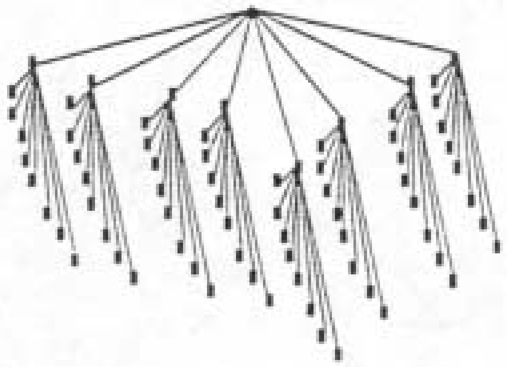

Fig. 2 : Octree with three levels.

4. Building the octree

Building the octree consists of the following steps:

(i) Determining the bounding box containing all triangles to be considered

(ii) Using the min/max xyz coordinates of the bounding box vertices as bounds for the root node of the tree and adding all triangles to the root node

(iii) Subdividing the current cube

(iv) Determining the triangles contained in each of the children and adding them to the associated triangle list

(v) Recursive subdivision of the child, if stopping criteria are not satisfied; otherwise, terminating subdivision

(vi) Returning to the root and traversing the tree, removing triangle lists from every node that is not a leaf

Subdivision of an octant is the process of bisecting the octant in the x, y, and z directions to form eight equal octants. Fig. 3 illustrates an octant with its eight children. Although there is no standard, the numbering scheme shown in Fig. 3 is typical. Note that child 0 is not visible.

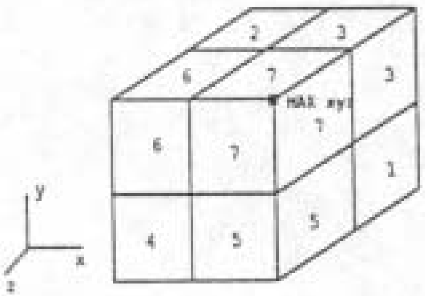

Fig. 3 : Octant showing child numbering scheme.

Step (iv) requires a series of tests to determine whether or not a triangle is to be associated with an octant. Only triangles contained in the parent octant need to be considered. There are five possible conditions which indicate that a triangle is to be associated with an octant. These conditions are:

(i) One or more vertices of the triangle lie within an octant.

(ii) One or more edges intersect an octant face, but no vertex is within the octant.

(iii) Edges of the octant intersect the triangle.

(iv) One or more vertices lie on an octant face.

(v) Part of a triangle's interior coincides with an octant face.

The first three of these conditions are illustrated in Fig. 4, Fig. 5, and Fig. 6. Case (iv) is a special case of (i), while case (v) is a special case of (ii).

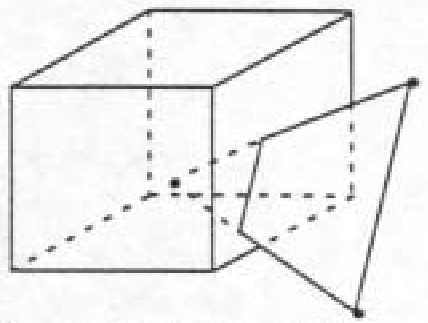
Fig. 4 : Vertex lies inside the octant.

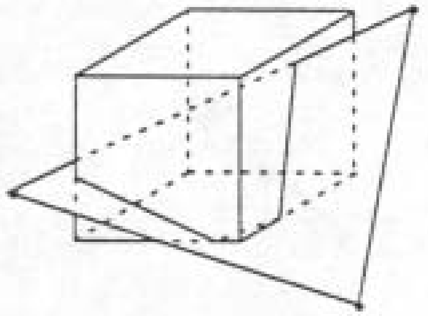
Fig. 5 : Edges intersect octant faces.

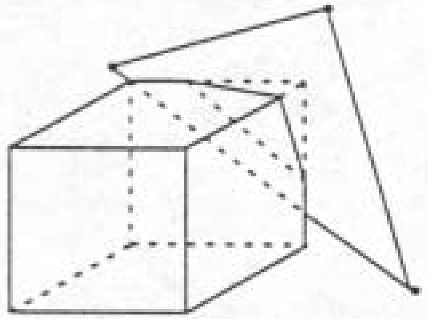
Fig. 6 : Octant edges intersect triangle.

The criterion for stopping the subdivision process is based on two quantities: the number of triangles contained in an octant and the size or volume of the octant. The goal is to optimize the relationship between storage requirements and search speed. Obviously, the minimum memory requirements occur when no tree is implemented, however, searching for the correct triangle is unacceptably slow in this situation. Maximum speed is achieved when each octant contains the minimum number of triangles possible, but this leads to unacceptably large storage requirements for complex geometries. In the current implementation, the maximum number of triangles per octant is ten. This number is increased for complex geometries with the maximum being $\frac{1}{100}$ of the total number of triangles. The "minimum-volume" rule is necessary to handle surfaces with degenerate edges. Given a sufficiently high resolution, a degenerate edge will always violate the "maximum-number-of-triangles" rule. Due to the fact that a single vertex is shared by every triangle associated with the degenerate edge (see Fig. 7). The "minimum-volume" rule allows the algorithm to detect such cases and treat them appropriately.

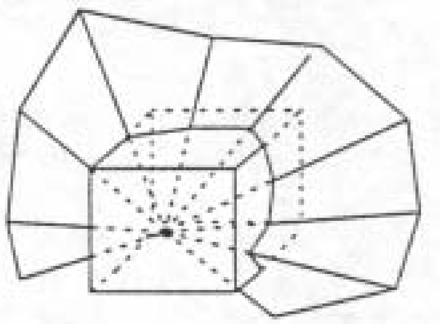

Fig. 7 : Octant containing a degeneracy.

5. Projecting the Coons patch using the octree

Determining the triangle (or triangles) with which a line $\mathbf{L}_{I,J}$ intersects requires finding all the leaf octants through which the line passes. This is accomplished by searching the tree and determining all the leaf octants containing the line. The search speed decreases exponentially with the depth of the search since children of octants that are not intersected by the line segment may be eliminated. If an octant is to the left, to the right, above, below, in front of, or behind both end points of a line segment it need not to be considered. The line is intersected with the faces of the remaining children of an octant. The search is done recursively, starting with the root octant. The result is a linked list of octants through which the line passes. Only leaf octants which contain surfaces are contained in the list. The line $\mathbf{L}_{I,J}$ is then intersected with each triangle in each of the octants to obtain projections (see Section 2).

6. Deriving additional approximation conditions

Usually, certain lines $\mathbf{L}_{I,J}$ do not intersect any triangle due to the existence of discontinuities in the geometry. If no intersection is found for a line $\mathbf{L}_{I,J}$, a bivariate scattered data approximation technique is used to derive "artificial projections." This is necessary, since the B-spline approximation to be constructed requires exactly $(N+1)\times(N+1)$ points to be interpolated. The method used for deriving these "artificial projections" is based on Hardy's reciprocal multiquadric, a standard bivariate scattered data approximation method (see [5]).

Each point $\mathbf{y}_{I,J}$ lying on the original geometry is projected perpendicularly onto the associated line $\mathbf{L}_{I,J}$. The (signed) distance $d_{I,J}$ between $\mathbf{y}_{I,J}$ and the projection onto $\mathbf{L}_{I,J}$ is computed (see Section 2). Thus, the problem of deriving "artificial projections" for lines $\mathbf{L}_{I,J}$ without intersection with the geometry becomes a bivariate scattered data approximation problem. The interpolation conditions that must be satisfied by the scattered data approximant are

$$d_{I,J} = \sum_{i,j \in \{0,\ldots,N\}} c_{i,j} \left(R + (u_{I,J} - u_{i,j})^2 + (v_{I,J} - v_{i,j})^2 \right)^{-p},$$

$$I, J \in \{0, \ldots, N\}, \quad (6.1)$$

where all distance values $d_{I,J}$ and parameter values $(u_{I,J}, v_{I,J})$ are considered for which a projection is known. It has been found that the value for R should be of the same order of magnitude as the spacing in the (u, v)-parameter space of the Coons patch. The value for p should be smaller than $\frac{1}{10}$, assuming that the Coons patch is defined over $[0,1] \times [0,1]$ (see [5]).

In most cases, it is advantageous to localize this scattered data approximation scheme by considering only a fixed number of values $d_{I,J}$ for the computation of each "artificial projection," *i.e.*, the number of equations in (6.1) to be considered for the generation of each local scattered data approximant is constant (see [5]). This approach is faster and tends to preserve local surface properties better than the global approximation approach. Once the scattered data approximant(s) is(are) computed, "artificial projections" $\mathbf{y}_{I,J}$ are defined by $\mathbf{y}_{I,J} = \mathbf{x}_{I,J} + d_{I,J}\mathbf{n}_{I,J}$, using distance values $d_{I,J}$ resulting from evaluating (6.1).

Fig. 8 shows the data to be considered in the scattered data approximation step. Distance values $d_{I,J}$ that have been obtained by projecting the Coons patch onto the geometry are indicated. The resulting scattered data approximant is evaluated for all parameter values $(u_{I,J}, v_{I,J})$ without known distance value.

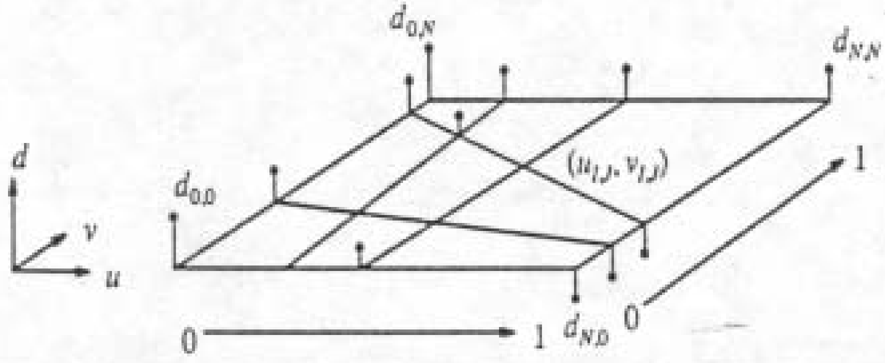

Fig. 8 : Input data for scattered data approximation.

7. Constructing the local and the global approximant

Having projected points of the Coons patch onto the original surfaces and having performed scattered data approximation, the resulting $(N+1) \times (N+1)$ points are interpolated by a C^1 continuous, bicubic B-spline surface written as

$$\mathbf{s}(u, v) = \sum_{j=0}^{3N} \sum_{i=0}^{3N} \mathbf{d}_{i,j} \, N_i^4(u) \, N_j^4(v) \,, \quad (7.1)$$

where $\mathbf{d}_{i,j}$ is a 3D B-spline control point and $N_i^4(u)$ and $N_j^4(v)$ are the normalized B-spline basis functions of order four (see [1], [3], or [8]). The (normalized) knot vectors have quadruple knots at the ends and triple knots in the interior of the domain. The control points $\mathbf{d}_{i,j}$ are the Bézier control points of the segments of a piecewise bicubic Bézier surface (see [1] and [3]).

In general, a real-world geometry can not be represented using a single B-spline surface. Thus, one must generate several B-spline surfaces, each one locally approximating the given geometry. The user must make sure that the number of B-spline surfaces is small and each Coons patch is "properly placed." If an initial Coons patch differs very much from the given geometry, it is not guaranteed to obtain a good B-spline approximation. More work needs to be done regarding this aspect.

Once all single B-spline approximants have been "wrapped around" the entire geometry, they are unioned such that the resulting global approximant is continuous. Since each B-spline surface consists of several bicubic Bézier surface segments, continuity conditions for two B-spline surfaces sharing a common boundary curve are assured by enforcing continuity conditions for all Bézier surface segments sharing common boundary curves. Special care is required at points where several B-spline surfaces (two, three, four, or more) come together and must share a common corner point. The conditions to be enforced for positional (C^0) and derivative/gradient (C^1) continuity are described in [1], [3], [4], [7], [8], and [14].

It is possible to enforce either C^0 or C^1 continuity along the boundaries of B-spline surfaces. Currently, it is not possible to interactively modify the order of continuity along boundaries of B-spline surfaces locally. At this point, the algorithm enforces C^0 continuity everywhere. The reason for this are slope/tangent plane discontinuities in the original geometry that must be preserved by the global B-spline approximation. The user must make sure that the boundary curves of certain Coons/B-spline surfaces conform closely to those curves on the given geometry where slope/tangent plane discontinuities occur.

8. Error estimation

The error measure that is used to measure the distance between a locally approximating B-spline surface $\mathbf{s}(u,v)$ and the given surfaces is an estimate for the maximum (absolute) distance between the local B-spline approximant and the given surfaces. This estimate is obtained by computing the maximum distance between a finite set of points on the local B-spline approximant and the given surfaces. The points on $\mathbf{s}(u,v)$ used for this process are

$$\begin{aligned}
\mathbf{a}_{I,J} &= \mathbf{s}\Big(\frac{2I+1}{2N}, \frac{J}{N}\Big), \;\; I=0,\ldots,(N-1), \;\; J=0,\ldots,N, \\
\mathbf{b}_{I,J} &= \mathbf{s}\Big(\frac{I}{N}, \frac{2J+1}{2N}\Big), \;\; I=0,\ldots,N, \;\; J=0,\ldots,(N-1), \;\; \text{and} \quad (8.1) \\
\mathbf{c}_{I,J} &= \mathbf{s}\Big(\frac{2I+1}{2N}, \frac{2J+1}{2N}\Big), \;\; I=0,\ldots,(N-1), \;\; J=0,\ldots,(N-1).
\end{aligned}$$

Due to the oscillation characteristics of bicubic surfaces, a bicubic B-spline approximant has large distances to the original surfaces in the center of each bicubic surface segment and at the midpoint of each cubic boundary curve segment.

The maximum distance between a single local B-spline approximant and the original geometry is estimated by computing the maximum value among all the (shortest) perpendicular distances between the points $\mathbf{a}_{I,J}$, $\mathbf{b}_{I,J}$, and $\mathbf{c}_{I,J}$ and the original surfaces. The distance of the global B-spline approximant and the original geometry is approximated by the maximum of all the maximal distances of the single local approximants. The perpendicular distances must *not* be computed for points $\mathbf{a}_{I,J}$, $\mathbf{b}_{I,J}$, and $\mathbf{c}_{I,J}$ that lie in a region of a B-spline approximant which is associated with a discontinuity in the original geometry.

9. Examples

Fig. 10 and Fig. 11 show B-spline surfaces approximating geometries reflecting the kinds of discontinuities frequently occurring in real-world CAD data. The examples clearly demonstrate the ability of the technique to approximate geometries with holes/gaps and intersecting surfaces. The boundary curves of the initial Coons patches are constructed from user-specified points and curves. The specified points and curves imply the four boundary curves of the Coons patch. Fig. 9 and Fig. 10 show the line segments $\mathrm{L}_{I,J}$ that are used to generate projections on the given surfaces.

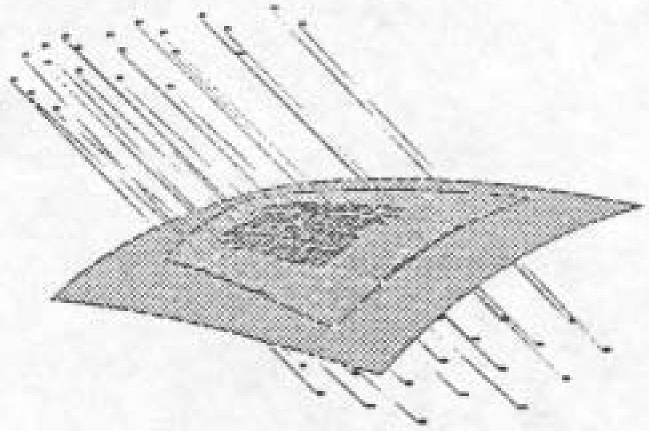

Fig. 9 : Approximation of part of single patch.

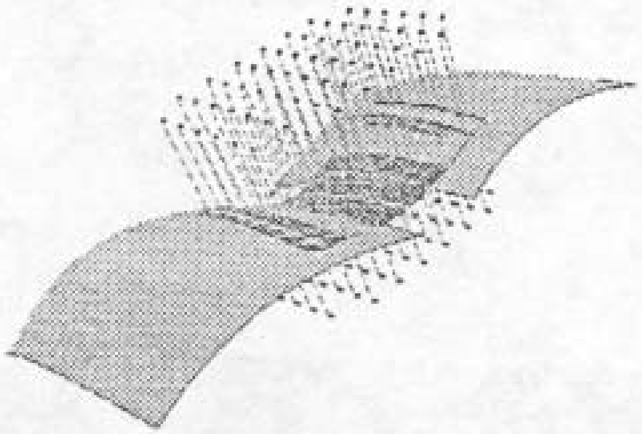

Fig. 10 : Approximation of surfaces with hole.

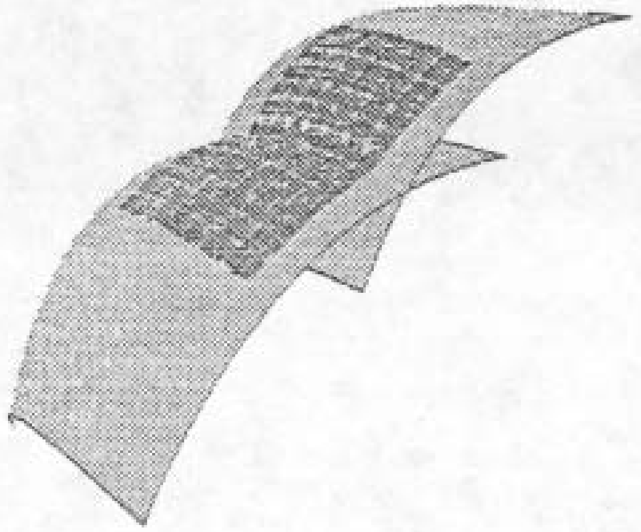

Fig. 11 : Approximation of intersecting surfaces.

Fig. 12 and Fig. 13 show real-world geometries and entire approximations thereof. The two figures show the resulting approximations consisting of several B-spline surfaces.

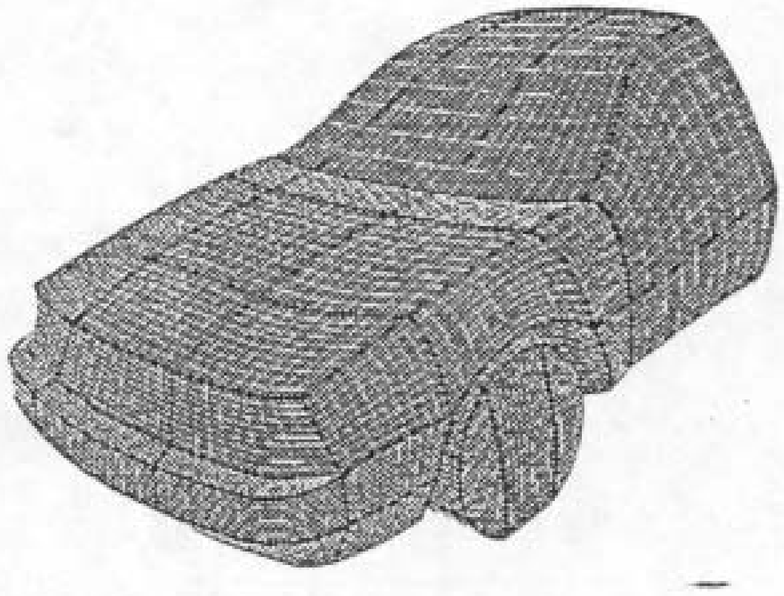

Fig. 12 : Approximation of car body.

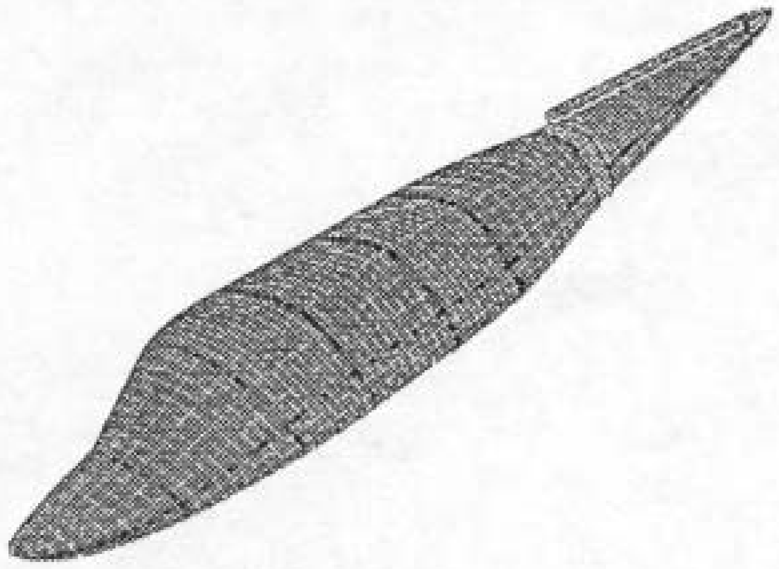

Fig. 13 : Approximation of helicopter.

10. Conclusion

The method described allows the approximation and correction of faulty CAD data. Geometries containing holes, overlapping surfaces, and intersecting surfaces can be approximated. Each local approximant is constructed from a combination of user-specified points and curves. The final global approximation is C^0 continuous and thus allows the preservation of boundary curves of original surfaces as well as slope/tangent plane discontinuities.

In the current implementation, all surfaces are stored as NURBS (Non-Uniform Rational B-Spline) surfaces using unit weights. Instead of using unit weights, the additional degree of freedom provided by them should be used to reduce the number of control points necessary to approximate a given geometry accurately. Unfortunately, choosing the weights of a rational approximation is still an open research problem.

11. Acknowledgments

This research was supported by the National Grid Project consortium and the National Science Foundation. Special thanks go to everyone involved in designing, implementing, and testing the modules of the approximation technique described, in particular to Ming-Laing Chen, Richard P. Felder, J. Adam Gaither, Shekhar Mahadevan, Kelly L. Parmley, Michael G. Remotigue, Po-Yu Tsai, and John B. Whitmire. John F. Dannenhoffer III, Gerald Farin, C. Wayne Mastin, Bharat K. Soni, Joe F. Thompson, and Nigel P. Weatherill contributed significantly to the success of this research.

REFERENCES

[1] BARTELS, R.H., BEATTY, J.C. and BARSKY, B.A., An Introduction to Splines for Use in Computer Graphics and Geometric Modeling, Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1987.

[2] BOEHM, W. and PRAUTZSCH, H. Geometric Concepts for Geometric Design, A K Peters, Ltd., Wellesley, MA, 1993.

[3] FARIN, G., Curves and Surfaces for Computer Aided Geometric Design, Third Edition, Academic Press, San Diego, CA, 1992.

[4] FAUX, I.D. and PRATT, M.J., Computational Geometry for Design and Manufacture, Ellis Horwood Publishers, Ltd., New York, NY, 1979.

[5] FRANKE, R., Scattered data interpolation: Tests of some methods, Math. Comp., Vol. 38, pp. 181–200, 1982.

[6] GEORGE, P.L., Automatic Mesh Generation, Wiley & Sons, New York, NY, 1991.

[7] HOSAKA, M., Modeling of Curves and Surfaces in CAD/CAM, Springer-Verlag, New York, NY, 1992.

[8] HOSCHEK, J. and LASSER, D. Fundamentals of Computer Aided Geometric Design, A K Peters, Ltd., Wellesley, MA, 1993.

[9] MEAGHER, D.J., Octree encoding: A new technique for the representation, manipulation, and display of arbitrary three-dimensional objects by computer, Technical Report IPL – TR – 80 – 111, Image Processing Laboratory, Rensselaer Polytechnic Institute, Troy, NY, 1980.

[10] MORTENSEN, M.E., Geometric Modeling, Wiley & Sons, New York, NY, 1985.

[11] PIEGL, L.A., *Rational B-spline curves and surfaces for CAD and graphics*, in: Rogers, D.F. and Earnshaw, R.A., eds., State of the Art in Computer Graphics, Springer-Verlag, New York, NY, pp. 225–269, 1991.

[12] SCHOOFS, A.J.G., VAN BEUKERING, L.H.T.M. and FLUITER, N.L.C., A general purpose two-dimensional mesh generator, Advances in Engineering Software, Vol. 1, pp. 131–136, 1979.

[13] THOMPSON, J.F., WARSI Z.U.A. and MASTIN, C.W., Numerical Grid Generation, North-Holland, New York, NY, 1985.

[14] YAMAGUCHI, F., Curves and Surfaces in Computer Aided Geometric Design, Springer-Verlag, New York, NY, 1988.