SURFACE ACQUISITION THROUGH VIRTUAL MILLING

MARSHAL L. MERRIAM
NASA AMES RESEARCH CENTER

Surface Acquisition Through Virtual Milling

Marshal L. Merriam
CFD Branch
NASA Ames Research Center
Moffett Field, CA 94035

Kumaran Kalyanasundaram
Iowa State University
Ames, IA 50011

Abstract

Surface acquisition deals with the reconstruction of three-dimensional objects from a set of data points. The most straightforward techniques require human intervention, a time consuming proposition. It is desirable to develop a fully automated alternative. Such a method is proposed in this paper. It makes use of surface measurements obtained from a 3-D laser digitizer - an instrument which provides the (x, y, z) coordinates of surface data points from various viewpoints. These points are assembled into several partial surfaces, using a visibility constraint and a 2-D triangulation technique. Reconstruction of the final object requires merging these partial surfaces. This is accomplished through a procedure that emulates milling, a standard machining operation. From a geometrical standpoint the problem reduces to constructing the intersection of two or more non-convex polyhedra.

1. Introduction

The field of surface definition has gained considerable importance in the past couple of years. Advances in computers and numerical flow algorithms have made simulation of 3-D fluid flow computationally tractable. The single greatest impediment to the use of this technology on complex 3-D objects, such as complete aircraft, is defining the shape of the objects themselves. This observation has focused considerable attention on surface definition and surface modeling.

One commonly used technique for surface definition involves re-creating an object from a series of body cross-sections, coordinates of which are available from a computer-aided design (CAD) database or direct measurements [1,2,3]. This process requires human intervention and is susceptible to human error. A more automated approach, both for measuring the object and for constructing a surface conforming to the measurements, is needed.

Three-dimensional objects can be measured quickly and automatically using a laser digitizer [4]. This device, like a coordinate measuring machine, returns the coordinates of a number of surface points. Instead of a mechanical probe, the digitizer uses optics for its measurements. The lack of mechanical inertia and physical contact in the measurement process allows a five order of magnitude improvement in speed over a coordinate measuring machine. The digitizer collects points at the rate of 14500/second, to an accuracy of about $0.2-0.5$ millimeters depending on the surface albedo and orientation. The object is held on a solidly built machinist's table on which it can be translated or rotated by computer driven

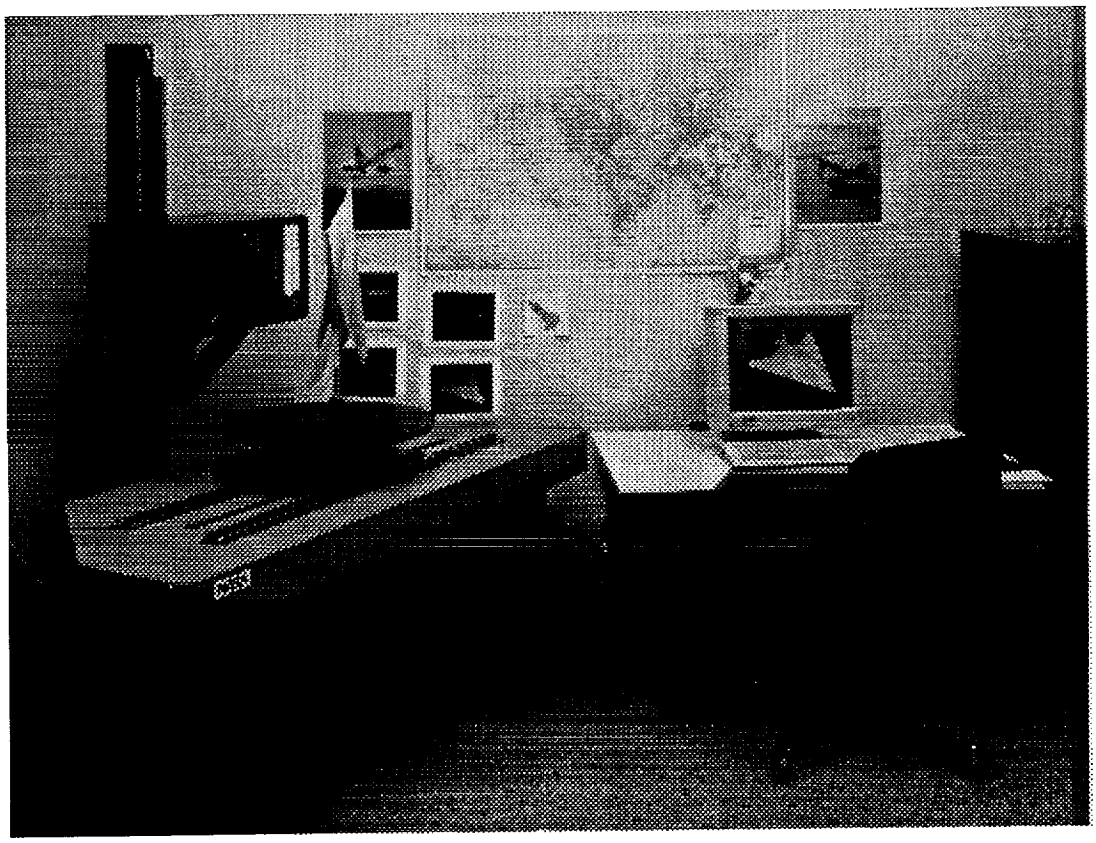

FIGURE 1. A Cyberware laser digitizer at NASA Ames Research Center.

servo motors, thus allowing observations from several different viewpoints to be expressed in a single coordinate system (see Figure 1). Merriam & Barth [5,6] have described the laser digitizer at length, interested readers are directed to their paper for further details.

Once the surface measurements are done, the surface incorporating the measured data must be reconstructed. Maksymiuk *et al.* [7] have proposed an algorithm based on pruning of unstructured grids. The method involves performing a Delaunay tessellation of the data points in three dimensions. This results in a solid body made up of tetrahedra, a valid reconstruction only for convex objects. The key insight is that no part of the object can obstruct the line of sight between the laser source and the object. This allows an improved shape to be reconstructed by deleting tetrahedra that intersect that line of sight. The algorithm is of $O(N \log N)$ complexity, where N is number of observed points. It has two main drawbacks: it generally removes more material than virtual milling, and the topological correctness of the reconstructed surface is sensitive to small errors in the experimental measurement. Both problems come from the discreteness of the pruning process: a tetrahedra is either removed or left untouched.

A very similar, but independently developed, procedure has been used by Faugeras *et al.* [8] for reconstructing 3-D scenes from stereo photographs. They also have shown how the reconstructed surface converges to the true surface when the sampling density increases.

Other algorithms for surface reconstruction exist. For example see Uselton[9] or Hoppe *et al.* [10]. We believe our algorithm to be fundamentally different and to have the following good properties.

i) It is reasonably efficient, having a formal complexity of $N \log N$ where N is the number of observed points.

ii) It always yields a topologically correct surface.

The remainder of the paper covers some of the algorithmic details of virtual milling including the relevant data structures and search techniques.

2. Surface Reconstruction From Digitizer Data

Our input data comes from a 3-D laser digitizer. This device provides prodigious amounts of data, but the data is given as a set of independent measurements. The desired output is a triangular faceted polyhedron which approximates the shape of the object being scanned.

Physical milling is the process of carving away material from an initial "blank" until the remaining material has the desired shape. Virtual milling (VM) simulates this process using computational geometry techniques. This immediately solves the most difficult problem; incorporating information from many different scans into a single part. The virtual cutting head resolves any small inconsistencies between scans. Whichever scan cuts the deepest prevails.

Two problems remain. First, the information from a single scan must be formed into a polyhedron which represents the volume to be milled out. Second, that polyhedron must be subtracted (in a solid modeling sense) from the workpiece.

2.1 Forming Surface Fragments From Individual Scans

The first job is to establish a triangular faceted surface fragment, an open two-manifold in 3-D, such that every measured point is fairly close to it. We give two separate strategies for doing this. One involves continuously adding points to gradually improve the surface approximation in the L_∞ norm. The other, which will be covered first, simply includes all the measured points from the outset, thereby avoiding the considerable expense of repeatedly computing the norm, but often resulting in a surface with many more vertices. In our experience, the difference is often a factor of 10.

There are a very large number of ways to triangulate N measured points. Each of these triangulations results in a surface fragment which includes all N points (by construction). Most of them can be eliminated by the use of visibility constraints.

It is known that the laser passes unimpeded from its source to the each point because observation requires illumination by the laser. This means that any triangulation which puts a triangle between the laser source and any observed point can be immediately discarded. One way to efficiently avoid such triangulations is to use projection methods.

Imagine for a moment that the laser originates from a point infinitely far from the workpiece ($z = -\infty$) so that all the rays are parallel to the z axis (the coordinate system is illustrated by Figure 2). Now project all the measured points onto the x, y plane (ignore the

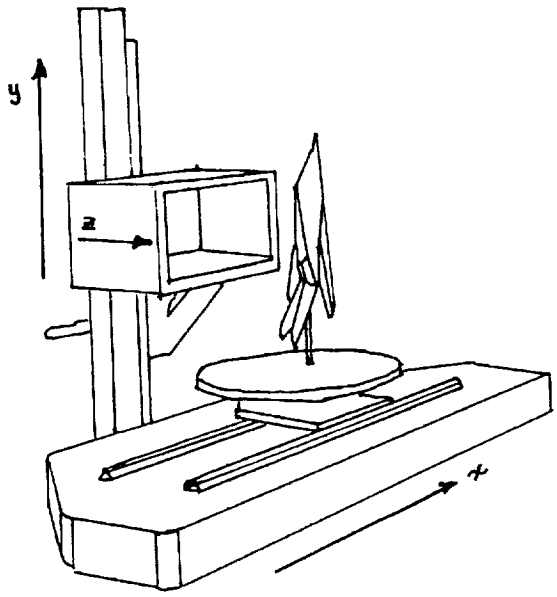

FIGURE 2. Coordinate systems for the laser digitizer.

z component) and perform some 2-D triangulation to establish connectivity between points. The corresponding 3-D triangulation (the one which has the same connectivity between points) does not violate the visibility constraint. That would imply that some edges of the projected triangulation cross. The converse is also true; every set of connections which satisfies visibility is a valid triangulation in the projected plane.

Now in practice, the focal length of the laser is not infinite, but only about three times its field of view[6]. The rays are perpendicular to the x axis, but are not parallel, forming an angle with the (x, z) plane that can be as much as 8.5 degrees. The appropriate projection in this case is cylindrical, rather than orthogonal, with the axis of the cylinder running parallel to the x axis and containing the laser source. In this coordinate system, the location in the (x, y) plane is given in polar coordinates (r, θ), the origin of which is at the laser source.

There are still an exponentially large number of ways to triangulate N points in the projected plane. The Delaunay triangulation in two dimensions is employed here. Delaunay triangulation is a classical problem in computational geometry and a well established technique for connecting scattered points [11 & 12].

A variation [13] involves using an incremental insertion algorithm for the Delaunay triangulation. After each insertion, the projected distance from the surface fragment to each measured point is computed and the one farthest away is determined and inserted. This process continues until the largest error falls below 0.25 mm, the nominal accuracy of the measurements.

Once the triangulation is done, the connectivity information is retained and the points are transformed back to their original (x, y, z) values. This gives a reasonable surface

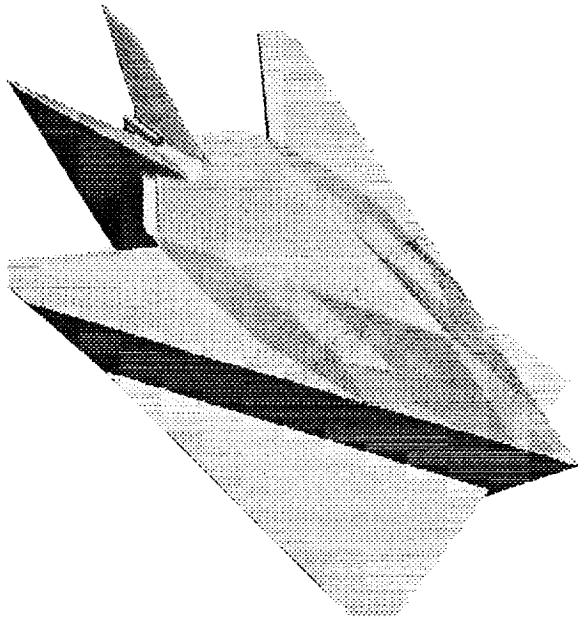

FIGURE 3. Unstructured triangulation of the top view of the F117A obtained using 2D Delaunay triangulation.

description (Figure 3), valid for shapes which can be reconstructed from a single scan, *e.g.*, sheet metal stampings.

For complex geometries, such as aircraft, multiple viewpoints are required. It is necessary to have, at least, a scan of the front and back views for reconstruction. In these situations a number of surface fragments have to be assembled. The end result is an approximation of the model as a non-convex polyhedron with triangular faces.

2.2 Combining Surface Fragments From Different Views

Combining the different views is achieved through a technique that emulates milling, a machining operation in which a workpiece is cut to the desired shape by careful removal of material. During each operation, the motion of the cutter is constrained to remove as much stock as possible without touching the finished part. Finally, only the finished part remains.

In the numerical analog the workpiece is any polyhedron (*e.g.*, a bounding box) with triangular faces, chosen to enclose the entire model. For each scan (view), it can be inferred that a polyhedral volume between the laser source and the object contains no material. This volume is excluded from the workpiece through a polyhedral intersection algorithm to be described later. In a solid modeling sense, the excluded volume is subtracted from the workpiece. Subtraction in this sense is commutative. Combining views consists of constructing the excluded polyhedra for each one and subtracting it from the partially finished part.

The problem here, is to subtract the volume of the polyhedron generated from a surface fragment, from the polyhedral workpiece $(P - Q)$. In the following sections an

algorithm for computing the intersection between two non-convex polyhedra with triangular faces is described in detail. The change needed to adapt this algorithm to perform the problem at hand, *i.e.*, to construct the intersection between a polyhedron and the compliment of another polyhedron, is also described.

3. Forming the Intersection of Two Non-Convex Polyhedra

Intersection problems have a wide variety of industrial applications [14], related to the fact that two objects cannot occupy the same space at the same time. Efficient, even optimal, algorithms have been developed for solving polygon intersection problems, but comparatively little is known about polyhedron intersections. Generalizations of the 2-D algorithms to 3-D are not straightforward.

In this work, only polyhedra with triangular faces are considered. This is done without loss of generality, since any higher degree polygon can be triangulated. This simplification allows reasonably efficient solution of polyhedron intersection problems in three-dimensions.

The intersection of an arbitrary number of non-convex polyhedra reduces to finding the intersection of two polyhedra. Given two non-convex polyhedra, P & Q, with triangular faces, form their intersection, $R \equiv P \cap Q$, such that the resulting polyhedron has only triangular faces.

The analogous problem in 2D is considered first. The intersection of two simple polygons A & B is a simple polygon C (Figure 4). Constructing C, from A and B, involves locating its vertices and its edges. Some of the vertices of C are vertices of polygon A, those which lie inside polygon B. Similarly, the vertices of polygon B which lie inside polygon A are vertices of C. The intersections of the edges of A and edges of B form the remaining vertices.

The edges of polygon C are all complete edges or edge fragments from polygons A or B. Edges of polygon A which lie entirely within polygon B, (*e.g.*, edge 1 in Figure 4), are edges of C. On the other hand, an edge of polygon A which lies entirely outside polygon B (*e.g.*, edge 2), is not. When an edge of A intersects one or more edges of B (*e.g.*, edge 3) only the edge fragments which lie inside polygon B are edges of C. Similar rules apply to edges of B.

Finding the intersection of two polyhedra can be accomplished by applying a similar procedure. The polyhedron R, formed from P $\cap$ Q, has nodes which are either nodes of P, nodes of Q, or intersections between the faces of P and the faces of Q. This problem, finding the intersection of two triangles in three-space, involves finding the endpoints of the line segment of intersection.

These line segments themselves constitute some of the edges of R. The other edges are formed from existing edges (the edges of P & Q) by treating them the same way as in polygon intersections. At this point, the intersection is a polyhedron with planar polygonal faces, some of which are not triangular. The higher degree polygons are triangulated so that the final polyhedron (R) has only triangular faces.

Summarizing then, computing the intersection of two polyhedra involves three main algorithms: polyhedron inclusion, line segment of intersection of two triangles in three-

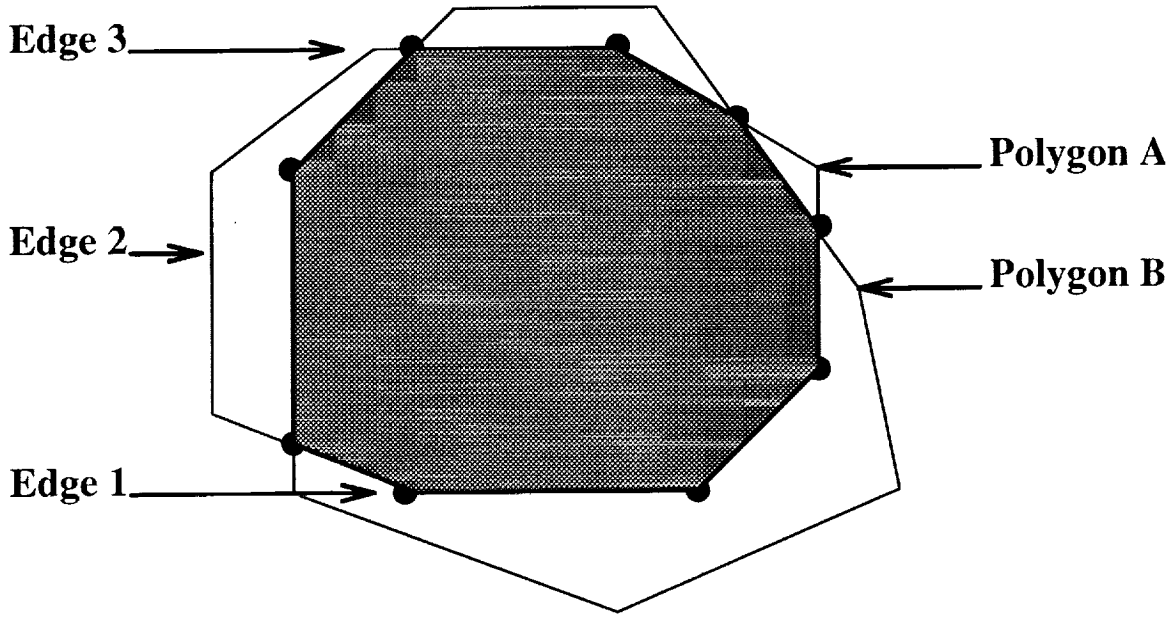

FIGURE 4. Intersection of two simple polygons A&B. The intersection is a simple polygon C, shown shaded. The thick dots denote the vertices of C.

space, and triangulating the interior of a simple polygon. These are described in the following sections.

3.1 Polyhedron Inclusion

Given a polyhedron and a point, is the point inside the polyhedron? To answer this a ray is drawn, emanating from the given point, typically along one of the three coordinate directions. The number of intersections between the ray and the polyhedron are counted. If the number is odd the point lies inside the polyhedron. Otherwise it lies outside the polyhedron. This algorithm is well known in 2D [14] and the 3D case is completely analogous.

Since the polyhedron is entirely composed of triangles, this only requires finding the 3D intersection of a ray with a triangle. By projecting both the ray and the triangle onto a plane normal to the ray, this problem is largely reduced to the 2D problem of point inclusion in a triangle.

An exhaustive search of all triangles will give the correct number of intersections. This is expensive. Sorting the triangles into a tree like structure drastically decreases the number of triangles searched each time. The data structure employed here is the alternating direction binary tree developed by Bentley [15]. Exposition of the search and sort algorithms is done, briefly, in a later section.

3.2 Intersection of Two Triangles in Three Dimensions

The polyhedron inclusion test determines which of the original vertices of P and Q will appear in R. The next step is to compile a list of intersecting pairs of triangles. These

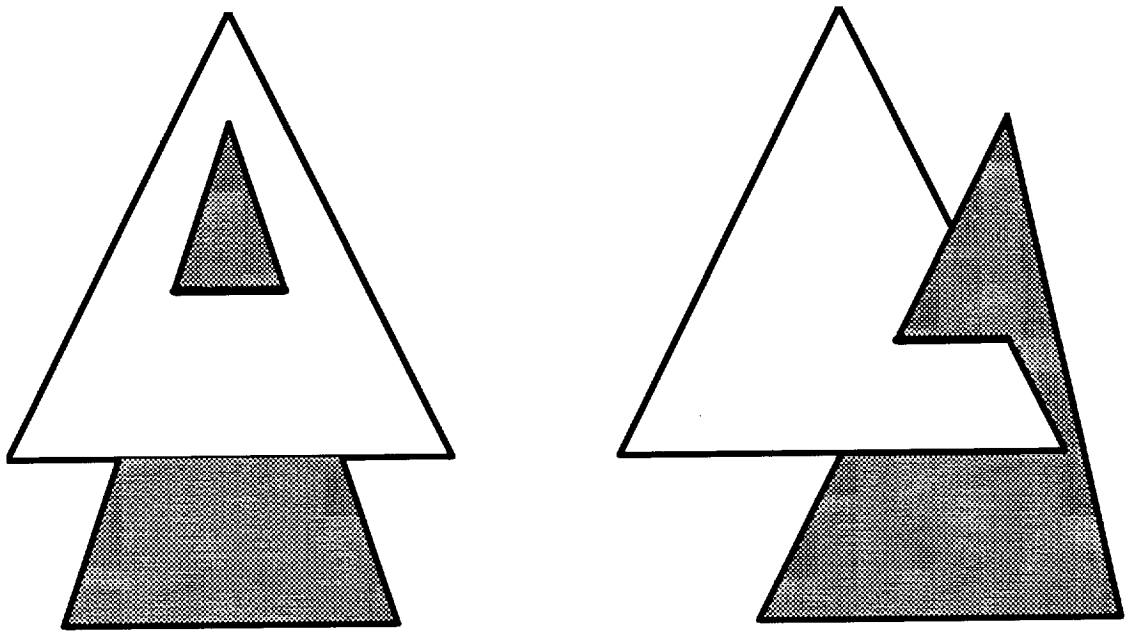

FIGURE 5. Two types of intersections of two triangles in three-space.

provide some of the edges of the final polyhedron along with all the remaining vertices.

Two triangles in 3D, A and B, intersect (if at all) along a line segment, each end of which lies on a separate triangle edge. To find these endpoints, each edge of triangle A is tested for intersection with triangle B, a problem essentially covered in the previous section. Similarly, edges of B are tested for intersection with triangle A.

Figure 5 shows two possible ways two triangles can intersect. Degenerate cases such as two intersecting, coplanar triangles, were not encountered. There are $O(N^2)$ pairs of triangles to test, most of which do not come close to intersection. Once again, the triangles have been sorted into a binary tree to avoid the expensive exhaustive search.

3.3 Constrained Triangulations

Edges of the two intersecting polyhedra (P $\cap$ Q) can be classified into three categories: a) edges of one polyhedron which lie entirely outside the other. Such edges are not part of the final polyhedron. b) The opposite situation, where edges of one polyhedron lie completely inside the other. Such edges are part of the final polyhedron. c) Edges of one polyhedron which intersect one or more triangular faces of the other. For such edges only those portions which lie inside the other polyhedron remain as part of the final polyhedron.

Figure 6a illustrates a situation where all vertices of a particular triangle lie inside the other polyhedron, and five new nodes have been added. The new node on the face of the triangle, N_5, is the point where an edge of Q intersects. The nodes on the edges (N_1, N_2, N_3, N_4) are the points of intersection between the edges of this triangle and the triangular faces of Q. The polygons shown in Figure 6b, are faces of P $\cap$ Q. The interiors of these simple polygons have to be triangulated in order to assure that the final polyhedron has only triangular faces.

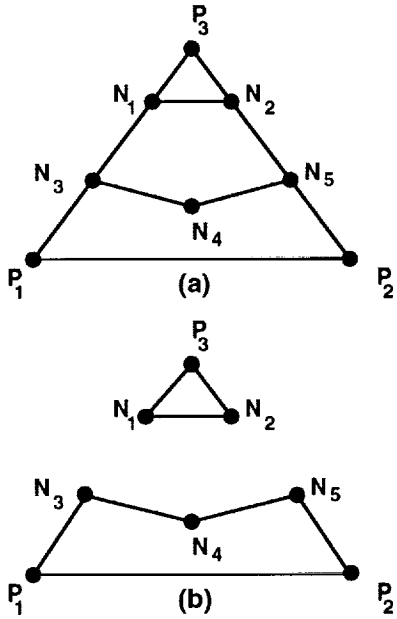

FIGURE 6. (a) This triangle has all three vertices (P_1, P_2, P_3) lying inside Q. N_1, N_2, N_3, N_4 are intersections between edges of P and faces of Q. N_5 is an intersection between this triangle and an edge of Q. All five points are vertices of P $\cap$ Q. (b) These regions are part of P $\cap$ Q.

Decomposing a polygon of degree N into $(N-2)$ triangles is a classical problem in computational geometry. The best algorithms operate in linear expected time (Chazelle [16]). Since we rarely encountered polygons of very high degree, programming simplicity determined our choice of an $O(N^2)$ complexity algorithm by Bern & Eppstein [17].

4. Data Structures

Finding the intersection of two non-convex polyhedra involves answering two types of geometric questions:

a) Which triangles (if any) in a given set, contain a given point?

b) Which intersect a given triangle? Since both queries appear many times, it is essential to use an efficient algorithm in answering them.

One approach that will work is to test each triangle for intersection with the relevant point or triangle. This technique (exhaustive search) has a run time of $O(NM)$, where N is the number of triangles, and M, the number of queries. Such a search can be prohibitively slow.

A quicker approach searches some of the triangles each time, with full confidence that the unsearched triangles would return negative responses. This involves presorting the triangles. The domain containing all the triangles is partitioned spatially [18]. Searching is then restricted to the partition where the given point (or triangle as the case may be) lies, and, possibly, a few of the neighboring partitions. Algorithms of this type typically have run times of $O(M \log N)$.

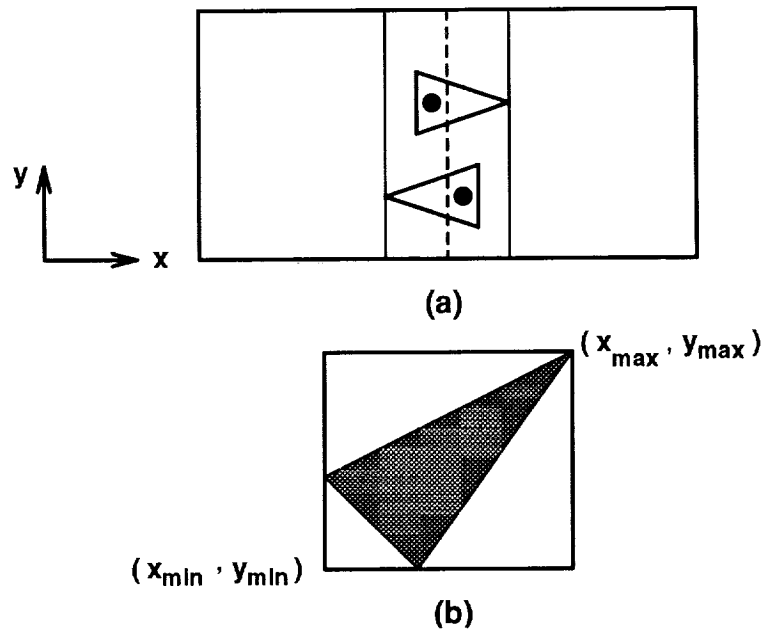

FIGURE 7. (a) A domain containing a set of triangles is partitioned into two regions with a slight overlap. (b) Triangle queries are handled by enclosing the triangle in a bounding box and searching all partitions which contain a portion of the box.

To achieve a suitable partitioning, triangles are treated as points. Each triangle is replaced by a unique point. The centroid has been chosen here, because it always lies inside the triangle. Then, the domain is divided into pieces with roughly equal numbers of triangles. The number of sub-domains created, during each division, depends on the type of data structure. In binary trees [15] the domain is divided into two halves at each level.

A one level example is illustrated in Figure 7a. The rectangular region containing the triangles has been divided into two. The dividing line is chosen to put an (approximately) equal number of triangles into each half. In this case the average of the x coordinates of the centroids was used to locate the vertical divisor. It is represented as a dotted line in Figure 7a. Bounding boxes can be constructed by determining the smallest and largest coordinate values of the vertices of the triangles contained in each half. Notice a small portion of the domain is common to both bounding boxes. Search efficiency depends on this overlap region being small compared to the size of the overall domain.

Now suppose we seek all triangles which contain a given point. By comparing x coordinate values, It can easily be determined which (if either) of the two bounding boxes contain the point. Clearly, if a point lies outside of a bounding box, it lies outside of all the triangles contained therein. If the point does not lie in the overlap region, at most half of the triangles will be searched.

For handling triangle queries, the triangle is enclosed in a bounding box (Figure 7b), and all the partitions which contain a portion of the box are searched. This effectively enlarges the overlap region, but the algorithm is otherwise identical.

A two-dimensional tree search has been implemented here for finding intersections

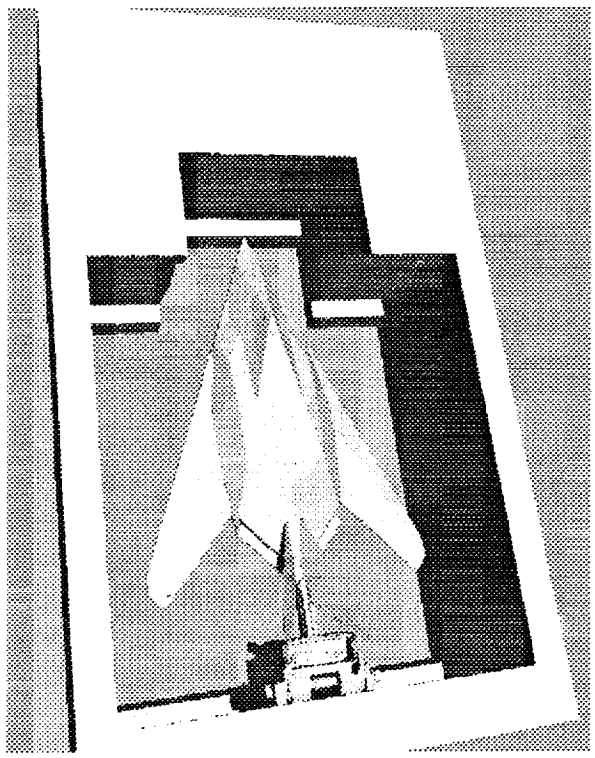

FIGURE 8. The F117a is still attached to the original blank and must be carefully separated. At this point, eight scans have been milled away.

between triangles in 3-D. The triangles are projected on to a plane, say the $x - y$ plane, and then partitioned according to the x and y coordinate values of their centroids. This approach was found to be more efficient than a three-dimensional tree search.

The polyhedron generated from a surface fragment is much smaller in size than the workpiece that encloses the entire model. It is useful to determine the bounding box of each polyhedron. It is only necessary to test for polyhedron intersection where these bounding boxes intersect, a significant optimization.

5. Finishing Operations

The physical model being scanned has to be supported securely in the digitizer. This usually implies a sting, but sometimes the model rests directly on the turntable. In any case, the desired geometry is usually attached to the remnants of the original blank. This situation is shown in Figure 8. When this happens in actual machining, the finished part is separated very carefully by hacksawing through the last connection. A similar approach is followed here. The cutting operation is simulated, again, through the intersection algorithm.

A number of other finishing operations are performed. The model is separated from the remains of the blank (clearing chips), the surface is given a consistent orientation, and some very small edges and triangles are removed (polishing). Finally, pinholes near the trailing edges are identified for later treatment.

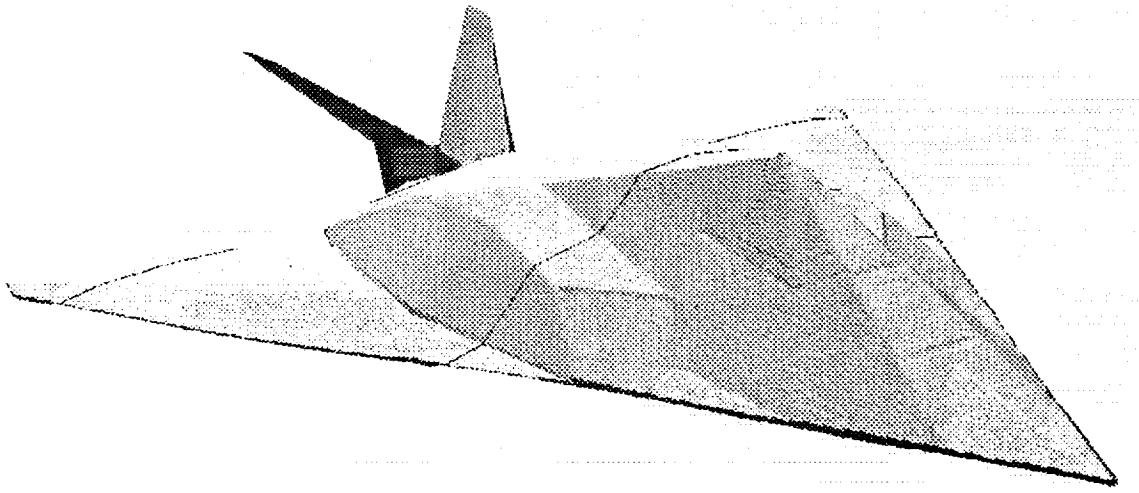

FIGURE 9. Front view of the reconstructed F117A. This incorporates information from eight scans.

6. Results and Discussion

The algorithm for generating an accurate geometric definition of three dimensional objects through virtual milling has been implemented on a Silicon Graphics workstation. The procedure has been tested on a F117A model scanned from several different viewpoints. Figure 9 shows the reconstructed F117A aircraft. The reconstruction involved eight scans and resulted in about 200,000 triangular faces. The sting, which supported the model during the digitizing process, was not completely removed. A portion of it is visible near the tail.

The runtime on an Iris 320/VGX was about six hours, including the scanning time. In an effort to reduce this time, the parallel intersection algorithm has been ported to the iPSC/860. Preliminary indications show a run time of under 10 minutes for the polyhedron intersection operations. The two intersecting polyhedra, P & Q, are equally split among the different processors by employing recursive coordinate bisection. Each processor is then responsible for constructing a portion of the intersection region.

7. Concluding Remarks

An algorithm for reconstructing 3-D objects from scattered data has been presented. The algorithm utilizes the Delaunay triangulation in two dimensions to generate partial surfaces from single views. Combining the different views is accomplished through virtual milling, a numerical analog of the physical machining operation. The technique is a general and automated method for reconstructing surfaces and assembling data from multiple views. Results for an F117A model clearly demonstrate these aspects. Ample research opportunities remain. For example, The resulting part is as accurate as the digitizer itself

but is not very smooth. It would be nice to have an algorithm to produce the smoothest possible part without moving any point more than the nominal measurement accuracy. On another front, Kalyanasundaram [18] has demonstrated a parallel implementation of polyhedron intersection on the Intel iPSC/860.

8. Acknowledgement

The authors would like to thank Ms. Catherine Maksymiuk for her implementation of DeFloriani's decimation algorithm. It is implemented efficiently $O(NlogN)$ and greatly simplifies the virtual milling process without degrading the reconstructed surface.

9. References

1 Edwards, T.A., "Definition and Verification of a Complex Aircraft for Aerodynamic Calculations," AIAA-86-0431, 24th Aerospace Sciences Meeting, Reno, Nevada, January 6-9, 1986.

2 Edwards, T.A., "Geometry Definition and Grid Generation for a Complete Fighter Aircraft," 58th Meeting of the Fluid Dynamics Panel Symposium on Applications of CFD in Aeronautics, Aix-en-Provence, France, April 7-10, 1986.

3 Atwood, C.A. and Van Dalsem, W.R.,"Flowfield Simulation about the SOFIA Airborne Observatory," AIAA-92-0656, AIAA 30th Aerospace Sciences Meeting, Reno, NV, January 1992.

4 Cyberware Laboratory Inc., 8 Harris Court, Monterey, Ca 93940

5 Merriam, M.L. and Barth, T.J., "3D CFD in a Day The Laser Digitizer Project," AIAA-91-1654, 26th Fluid Dynamics, Plasma Dynamics & Lasers Conference, Honolulu, June 24-27, 1991.

6 Merriam, M.L. "Experience With The Cyberware 3D Digitizer," Proceedings of CAD and Engineering Workstations '92, NCGA, Anaheim, California, March 9-12, 1992.

7 Maksymiuk, C. and Merriam, M.L., "Surface Reconstruction from Scattered Data Through Pruning of Unstructured Grids," AIAA-91-1584, 10th Computational Fluid Dynamics Conference, Honolulu, June 24-27, 1991.

8 Faugeras, O.D., Le Bras-Mehlman, and Boissonnat, J.D., "Representing Stereo Data with the Delaunay Triangulation," Artificial Intelligence, v44, pp 41-87, July, 1990.

9 Uselton, S.P., "Surface Reconstruction From Limited Information," Ph.D. dissertation, The University of Texas, Dallas, Texas, December, 1981.

10 Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Steutzle, W., "Surface Reconstruction From Unorganized Points," Computer Graphics 26, July 2nd, 1992.

11 Merriam, M.L., "An Efficient Advancing Front Algorithm For Delaunay Triangulation," AIAA-91-0792, 29 Aerospace Sciences Meeting, Reno, Jan 7-11, 1991.

12 Tanemura, M., Ogawa, T.,and Ogita, N., "A New Algorithm For Three Dimensional Voronoi Tessellation," Journal of Computational Physics 51, pp 191-207, 1983.

13 De Floriani, L., Falcidieno, B., and Pienovi, C., "Delaunay-based Representation of Surfaces Defined Over Arbitrarily Shaped Domains," Computer Vision, Graphics, and Image Processing, vol. 32, pp 127-140, 1985.

14 Preparata, F.P. and Shamos, M.I., "Computational Geometry An Introduction," Springer-Verlag, 1985.

15 Bentley, J.L., "Multidimensional Binary Search Trees Used For Associative Searching," Communications of the ACM, 18(9), pp 509-517, September, 1985.

16 Chazelle, B., "Triangulating a Simple Polygon in Linear Time," In Proc. 31st IEEE Symposium Foundations of Computer Science, pp 220-230, 1990.

17 Bern, M. and Eppstein, D., "Mesh Generation and Optimal Triangulation," Xerox PARC Technical Report, CSL-92-1.

18 Knuth, D.L, "The Art of Computer Programming, Vol.3," Addison Wesley, 1973.

19 Kalyanasundaram, K., "Parallel Surface Definition Through Virtual Milling," Ph.D. dissertation in progress, Iowa State University, Ames, Iowa.

Dr. William Van Dalsem
Computational Technology Branch
NASA Ames Research Center
Moffett Field, CA 95050
(415) 604-4469
e-mail: vandal@nas.nasa.gov

Mr. Guru R. Vemaganti
Lockheed Eng. & Sciences Co.
144 Research Drive
Hampton, VA 23666
(804) 766-9473

Mr. David L. Whitaker
Analytical Services and Materials, Inc.
MS 128
NASA Langley Research Center
Hampton, VA 23681
(804) 864-2150
e-mail: d.l.whitaker@larc.nasa.gov

Dr. Laurence B. Wigton
BOEING
MS 128
NASA Langley Research Center
Hampton, VA 23681
(804) 864-7885
e-mail: lbw9902@tab00.larc.nasa.gov

Dr. Thomas A. Zang, Jr.
MS 150
NASA Langley Research Center
Hampton, VA 23681
(804) 864-4082
e-mail: t.a.zang@larc.nasa.gov

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1993	Conference Publication

4. TITLE AND SUBTITLE
Unstructured Grid Generation Techniques and Software

5. FUNDING NUMBERS
WU 505-90-53-02

6. AUTHOR(S)
Mary-Anne K. Posenau, Editor

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23681-0001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA CP-10119

11. SUPPLEMENTARY NOTES
Proceedings of the NASA Workshop on Unstructured Grid Generation Techniques and Software held at the NASA Langley Research Center, April 27-28, 1993

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified–Unlimited

Subject Category 61

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
The Workshop on Unstructured Grid Generation Techniques and Software was conducted for NASA to assess its unstructured grid activities, improve the coordination among NASA centers, and promote technology transfer to industry. The proceedings represent contributions from Ames, Langley, and Lewis Research Centers, and the Johnson and Marshall Space Flight Centers. This report is a compilation of the presentations made at the workshop.

14. SUBJECT TERMS
Unstructured Grid Generation; Unstructured Grids; Computational Fluid Dynamics

15. NUMBER OF PAGES
360

16. PRICE CODE
A16

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	