SANDIA REPORT
SAND2001-1283
Unlimited Release
Printed May 2001

Cloud to CAD

Arlo L. Ames

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Sandia National Laboratories

SAND 2001-1283
Unlimited Release
Printed May 2001

Cloud to CAD

Arlo L. Ames
Engineering and Manufacturing Software
Intelligent Systems and Robotics Center
Sandia National Laboratories
P. O. Box 5800
Albuquerque, NM 87185-1010

Abstract

This paper documents work performed to convert scanned range data to CAD solid model representation. The work successfully developed surface fitting algorithms for quadric surfaces (e.g. plane, cone, cylinder, and sphere), and a segmentation algorithm based entirely on surface type, rather than on a differential metric like Gaussian curvature. Extraction of all CAD-required parameters for quadric surface representation was completed. Approximate face boundaries derived from the original point cloud were constructed. Work to extrapolate surfaces, compute exact edges and solid connectivity was begun, but left incomplete due to funding reductions. The surface fitting algorithms are robust in the face of noise and degenerate surface forms.

Acknowledgments

Thanks to the following for contributing to the development of the Cloud to CAD: Pat Xavier, Charles Little and Chris Wilson. Special thanks are due to Ralph Peters, who got excited enough about the results to field them, ultimately validating the work.

Thanks to Ralph Peters and David Hensinger for serving as willing reviewers and Sharon Blauwkamp for help in document preparation.

Contents

Figures

Introduction

This report documents the results and recommendations of the Cloud to CAD LDRD project.

A fundamental problem in applying automatic geometric reasoning algorithms to the real world is that of acquiring appropriate models of real world objects. The problem exhibits itself whether the task is to field an autonomous mobile robot or trying to manage products throughout their life cycle. We have a variety of computational tools to assist in reasoning about collision avoidance, grasp design, product design, analysis, manufacturing, and inspection. All of these tasks rely on accurate geometric models. The geometric models they require are either CAD solid model representations, or can be directly derived from such models. Objects in the real world frequently differ from their original CAD descriptions due to manufacturing process, modification or damage, and many (most?) objects in the real world have no CAD description.

Producing CAD representations of existing parts by hand requires tremendous labor, involving tedious (and expensive) measuring and detailing. We have developed an intelligent system that "scans in" objects and produces the desired solid model representations (although it might be incomplete due to incomplete information).

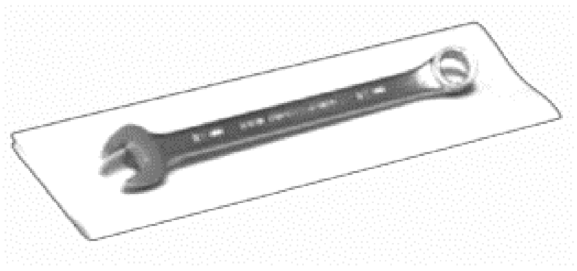

Figure 1: A photograph of a wrench

Due to the availability of range sensors, intelligent systems are beginning to visually “see” their environments in three dimensions. The sensors produce a loosely connected cloud of data points that describe the distance from the camera to the object (Figures 1,2). The data describes how far away objects are in any viewing direction, and thus can be used, for example, to plan collision-free paths through the environment.

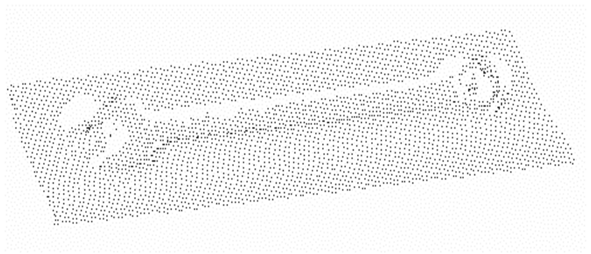

Figure 2: Raw scanned data. The “cloud” of points.

Beyond the ability to “see” is the ability to “understand” the environment. Raw scanned range data fails this test, because it contains fictitious and missing boundaries (due to occlusions) and it fails to explicitly represent surface curvature. A robot can use range information to avoid collisions, but will have difficulty manipulating objects because the data neither says that a bolt is round nor that it is separate from the table it is laying on. A wide variety of current geometric reasoning capabilities, including assembly and fixture planning, currently rely on CAD solid model data. Raw scanned data currently cannot support such applications.

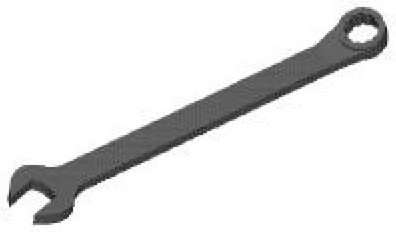

Figure 3: A CAD model of the wrench.

The information we seek is implicit in the scanned range data; people are able to determine the shapes of objects in a scene merely by looking.

Previous Work

At the inception of this project, most of the previous work had been performed at universities. Current university work [1] focused on producing a solid model by using scan facets directly for solid model faces, without computing exact surfaces. Multiple views were accounted for by performing Boolean intersections of solids produced from each view. Such an approach produces incredibly complex models, which require hours to process and still lack the exact surfaces required for use beyond visualization and production of crude approximations of real parts. Other work in the area focused mainly on producing surface models from scanned data, stopping far short of closed boundary representation topology.

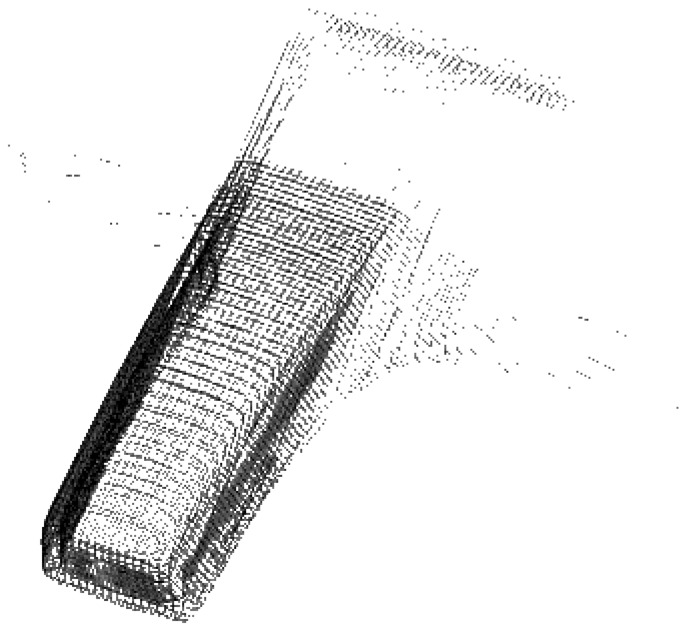

Figure 4: An example surface fit from a commercial product.

Presently, CAD vendors are introducing a variety of products in this area. While many of these efforts develop closed, surfaced representations, they fall short of the goal of this project. In general, they tend to produce NURBS[1]-surfaced representations, independent of the presence of lower-order surface interpretations of the geometry. Little regard is paid to the topology of the result – the products are mainly concerned with obtaining a reasonably pretty picture of the scanned objects. Note in Figure 4, we find an example of the results produced by a commercial reverse-engineering package for a scan of a water faucet part. Note that the curves shown are isoparameter curves on NURBS surfaces. They are identical to lines produced during the data acquisition process. Discontinuities in surface are smeared over in the representation, rather than producing different surfaces. It is not uncommon in such representations to find artifacts of the original scanning process.

Our concern here is rather different: to produce a model with minimally complex geometric surfaces, with a minimum topology. We insist on being sensitive to lower order (e.g. quadric) surfaces. We seek to minimize or eliminate scanning artifacts from the final models.

Technical Issues

A variety of technical issues are present in the problem of converting scanned data to CAD models.

Coping with noise. Scanned data is subject to many sources of noise, including specular reflections and sensor errors. Noise can significantly hamper efforts to recognize shape, and will

[1] Non-Uniform Rational B-Spline Surface

affect the accuracy of the final results. Thus, means for minimizing the impact of noise will be required.

Integrating data from multiple viewpoints. From a single viewpoint, it is only possible to view the visible side of an object. Any surfaces of the object that don't face the sensor will be missing from the model. Multiple sensor views can be used to fill in gaps in the model, but they must be reconciled with each other.

Segmenting the model. Scanned data provides a map of coordinates on the surface of the model, but does not distinguish between different surfaces of a part or between surfaces of different parts. It is necessary to partition the scanned points into groups that can be associated with CAD model surfaces, and these groups are not known a priori.

Dealing with multiple geometric interpretations. Scanned data is subject to a vast number of geometric interpretations. At one extreme, every three adjacent points can define a planar facet, which will produce models with large numbers of planar faces (Figure 4). At another extreme, the grid of sample points could define the basis for one large, extremely complex spline surface. Neither extreme case is "correct", because correctness (for design and analysis purposes) requires the model to have a minimum number of surfaces of minimum complexity. Algorithms for proposing geometric interpretations must be very efficient in exploring various interpretations.

Achieving model closure. CAD systems require closed models. Even with multiple views, it is possible for a model to lack some surfaces (e.g. surfaces in contact). Creating completely bounded manifold solid models requires the ability to construct artificial boundaries. A more subtle solid model closure issue relates to edges. Edges define the intersection of surfaces, so adjacent surfaces must have equations of a form that produces well-defined edges. A global adjustment of surface equations might be necessary to insure valid edge definition.

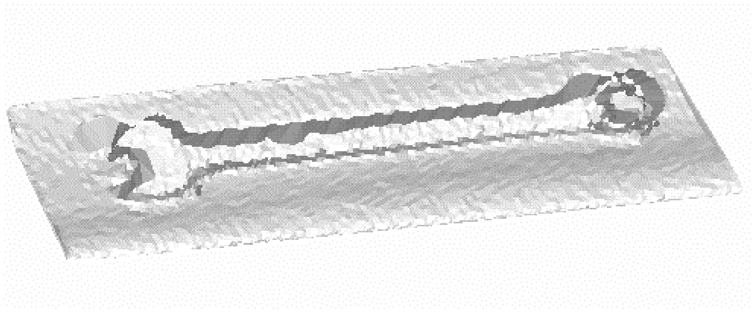

Figure 5: Facets derived directly from the cloud of points.

Representations

We deal here with two fundamental representations: point clouds and boundary representation solid models.

Point Cloud

The data produced by a scanned range system is typically referred to as a "point cloud". A point cloud is a collection of points, together with a loosely defined topology. The points are typically three-dimensional data, defined in a coordinate system relative to the scanner (or relative to a larger world coordinate system that the scanner is positioned within). The loosely defined topology is an artifact of the scanning process. A scanner acquires point data by sweeping through the scene. At any given instant, some collection of points is produced. Adjacency between points is determined either by geometric proximity or by some adjacency defined by the scanner (e.g. adjacent pixels in a camera suggest some adjacency. It is possible, due to occlusion, reflection, noise, and a variety of other effects, for points to be "missing" – no return to the sensor occurs where we might expect adjacent points to be. Where points are close, both topologically and geometrically, it is possible to infer adjacency.

Point cloud data is inherently noisy. Data taken from any given position can only show what is visible from that location. Without augmentation, point clouds only convey positional information – they lack any notion of the color or surface texture of the object being viewed.

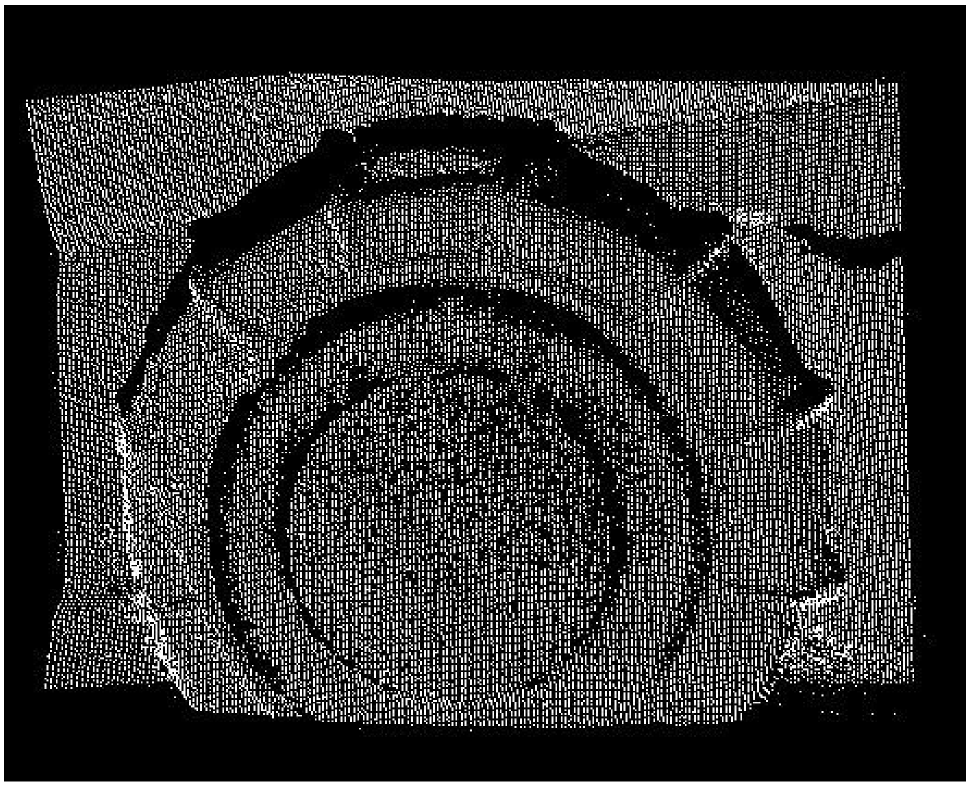

Figure 6: Scanned representation of a landmine. Note that real-world object themselves can be dirty, contributing further to noise in the scan.

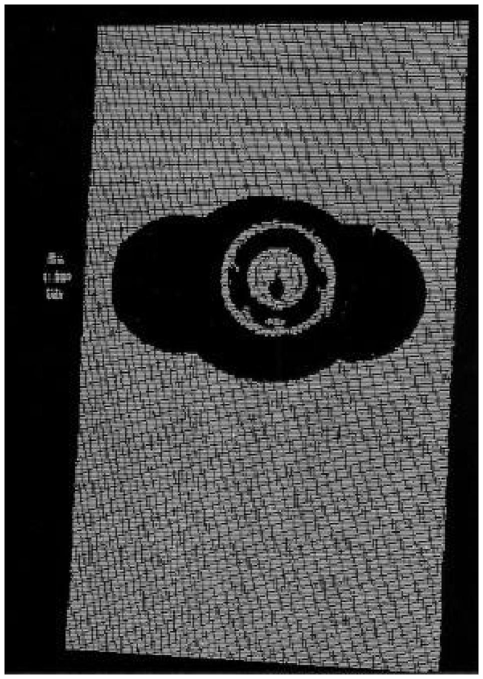

Figure 7: A doorknob. This doorknob was shiny, resulting in scanning loss in annular regions around the keyhole. Note also the knob has two shadows – one shadow was occlusion of the illuminating laser, while the other is occlusion of the camera's view.

Boundary Representation

Boundary representation (b-rep) is a means of representing solid objects. It is one of a family of geometric representations (including, for example, constructive solid geometry and spatial enumerations), and is used in design systems because it is an evaluated representation that conveniently represents precise surface information.

A minimal three-dimensional boundary representation (as present in, for example, Pro/Engineer) consists of faces, loops (also called contours) and edges[2]. *Faces* are bounded portions of *surfaces*. A face is bounded by one or more *loops*. A loop is a list of directed *edges*. Frequently, edges are shared by adjacent faces – for such a representation, it is necessary to provide a representation (e.g. coedge) of which direction the edge is traversed on each of the adjacent faces. For a given face, two notions of "interior" are necessary. The first tells which side of the unbounded surface constitutes the outside of the face; a flag is used to define whether the normal of the surface or its complement is outside. The second notion of interior tells which side of each contour is in the bounded region of the face. The tangent direction of traversal of edges, crossed with the direction of surface normal at a point in question, determines which side of each edge is on the interior of the face. An edge is a bounded portion of a *curve*; two *points* are used to describe the beginning and ending of the interior of the edge.

[2] Boundary representations can be much richer than Pro/Engineer's. ACIS, for example, represents bodies, lumps, shells, faces, loops, coedges, edges and vertices.

Each surface is represented by a geometric equation, typically in parametric (rather than implicit) form. CAD systems typically represent the following types of surface: plane, cylinder, sphere, cone, torus, and NURBS (non-uniform rational b-spline surface). In engineered objects, the natural quadric surfaces (plane, cylinder, cone, sphere) occur much more frequently than free-form surfaces.

Each curve is also represented as a geometric equation, typically in parametric form. CAD systems ordinarily represent the following types of curves: line, ellipse, and spline. Note that for each type of surface-surface intersection, it is possible to use a specific type of curve (e.g. parabolas and hyperbolas for cone-plane intersection). In practice, CAD systems use spline curves for most types of intersection (other than lines and ellipses), and frequently only approximate the true intersection curve.

CAD systems ordinarily represent exact, idealized geometry – that is, geometry as the designer would wish given a perfect world. For example, the angle of intersection between two planes would typically be 90°, and not 89° or 91°.

Approach

The problem of extracting solid model data from scanned models can be viewed as a recognition problem: given a low-level representation of information, attempt to recognize higher-level patterns. The recognition process involves partitioning the point set into collections that define surfaces, finding intersections of those surfaces to produce edge equations, stitching surfaces at edges, and adjusting surfaces and adding new surfaces to improve closure.

Importing scanned data points directly into a CAD system (as either points or polygons) was a basic first milestone. Such representations may seem unimportant, but this data is appropriate in cases where the scan represents non-designed geometry (e.g. a hip joint in a medical application), where parts are designed to *interface* to scanned geometry without being composed of it.

Next, we developed algorithms that search the point/facet information to determine a collection of surfaces (e.g. planes, cylinders) that bound the object. Hints such as adjacency and similarity of surface normals will be used to limit the amount of search required. We will account for noise in the scanned points by allowing tolerance in the surface fitting algorithms. At this point, models consisting of disconnected surfaces can be produced.

After surface representations could be proposed, we proceeded with the task of bounding the surfaces and determining adjacency. Producing solid model topology requires defining edge geometry that correctly represents intersections between adjacent surfaces. In some cases, surface intersections are not well formed due to noise in the data, so surface parameters (or the surface recognition algorithms) can require modification to form a reasonable model.

In support of robotic recognition applications, we began work to search a database of models based on the scanned surface data. Some progress was made there.

Directly Importing Triangles into a CAD System

Previous work (e.g. [Little96]) with scanned range data has produced relatively stable triangulating algorithms. Given that basis, we simply constructed faceted solid models from the triangulations, and imported them into CAD algorithms.

As expected, the triangulations render reasonably well, albeit slowly. Most CAD construction operations are unwieldy with such representations, as a great deal of selecting must occur to perform a simple operation like selecting a face. Many operations, such as mass-properties calculation, take far longer than the same operations on models using many fewer faces having higher-order surface equations. Thus, we are driven by the desire for efficient representation to seek the higher-order representations.

Figure 5 shows the results of importing triangles directly into a CAD system. The results are difficult to use, but could be acceptable if nothing else is available.

Not shown (due to difficulty visualizing the data) is a scanned representation of an automobile tire. We computed mass properties directly from scanned data for the tire. Mass properties computation for a solid model of a complete tire required roughly 20 minutes; mass properties for the faceted scanned representation required 1-1/2 days. Even the model construction of the faceted representation was slow, requiring 8 hours (4 if we eliminated sharing of adjacent edges, essentially producing a surface model rather than a solid).

Surface Fitting

The current surface representations of choice in CAD systems include planes, natural quadric surfaces, tori and NURBS. Some systems use NURBS exclusively, but more frequently the systems use the lower-order special cases as they occur frequently in design. Estimates range that as high as 80 percent of all the designed surfaces in mechanical parts are planar or quadric, so they certainly form an important class of surfaces to recognize. Quadric surfaces provide remarkable representational efficiency, as they possess closed-form solutions for many of the geometric queries required.

Quadric Fitting

Quadric surfaces have the general equation:

$$ax^2 + by^2 + cz^2 + 2fyz + 2gzx + 2hxy + 2px + 2qy + 2rz + d = 0$$

or, in matrix notation,

$$f(x,y,z) = [x \quad y \quad z \quad 1]\begin{bmatrix} a & h & g & p \\ h & b & f & q \\ g & f & c & r \\ p & q & r & d \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 0$$

If we extract a rigid motion (translation and rotation), any quadric can be reduced to one of 17 standard forms. The standard forms include both real and imaginary surfaces, including 3 forms of planes.
For CAD representation purposes, we focus on real ellipsoid, real quadric cone and real elliptic cylinder. These include the important special cases of sphere, right circular cone, and right circular cylinder. We have ignored hyperboloids, paraboloids, and hyperbolic and parabolic cylinders. These surfaces can be easily added if desired, but they occur much less frequently than the others. We ignore the imaginary surfaces, as they don't occur in nature. We also ignore the planar forms, as they can be more easily fit with a specialized plane-fitting algorithm with less error.

Least squares minimizes the equation:

$$S = \sum_{i=1}^{n} [f(x_i, y_i, z_i)]^2$$

A traditional approach would take partials of S with respect to each of the quadric coefficients, set each partial to zero, and solve the resulting system of equations. Such an approach is unstable, due to bad numerical behavior. The direct approach involves a matrix containing fourth degree products of x, y, and/or z coordinate data, summed over all the points being used in the surface fit (e.g. $\sum_{i=1}^{n} x_i^2 y_i^2$). Attempts to use a brute-force approach have led to surface fitters that are extremely sensitive to noise and frequently produce erroneous results, such as a cylindrical fit that occurs at 90 degrees to the correct fit [Feddema97].

Fitting quadrics (or planes, for that matter) involves potential degeneracies which can make fitting difficult. The input data can lie on a global coordinate plane, leaving a geometric component uniquely zero. A degenerate form of the surface can occur, leaving one or more coefficients zero. In fact, all interesting quadric forms involve one or more of the coefficients being zero, after removal of a rigid transformation (translation and rotation). These degeneracies cause terrible numerics, resulting in the brittle fitting behavior noted.

Rather than taking the brute force approach to least squares, we have employed a singular value decomposition, SVD (c.f. [Press88]). The solution of least squares SVD is known to be robust in the face of difficult numerics, producing a least squares result even if the system is over- or under-determined. Ill-conditioned terms have their coefficients set to zero, essentially removing them from the equation.

To use the SVD least squares from [Press88] we merely have to formulate the problem in the form:

$$y(x) = \sum_{k=1}^{M} a_k X_k(x)$$

The $X_k(x)$ are arbitrary fixed functions of x, called basis functions. In the quadric case, we could choose the basis functions as terms in the generalized quadric:

$$\begin{array}{ccc}
 & X_1(x) = 1 & \\
X_2(x) = x & X_3(x) = y & X_4(x) = z \\
X_5(x) = 2xy & X_6(x) = 2yz & X_7(x) = 2xz \\
X_8(x) = x^2 & X_9(x) = y^2 & X_9(x) = z^2
\end{array}$$

We will call this the simple formulation. [Pratt87] suggests either the simple formulation, or an alternative set of basis functions as follows:

$$\begin{array}{ccc}
 & X_1(x) = 1 & \\
X_2(x) = x & X_3(x) = y & X_4(x) = z \\
X_5(x) = 2xy & X_6(x) = 2yz & X_7(x) = 2xz \\
X_8(x) = x^2 + y^2 + z^2 & X_9(x) = x^2 - y^2 & X_9(x) = y^2 - z^2
\end{array}$$

The alternate set of basis functions is expected to provided better fits for reasons cited in [Pratt87].

We performed experiments with both real sampled data and synthetic data degraded by adding Gaussian noise. The alternative basis functions tended to provide more correct curvature estimates than the simple formulation, particularly in lightly sampled surfaces, or surfaces where only a small portion of the curvature is visible.

In formulating our fitting equation for use with SVD, note that we must have a function $y(x)$ to fit to. Note that y(x) is ordinarily a constant, different for each sample. In our case, the standard quadric equation has 0 as that constant. The SVD fitting algorithm produced trivial results (all coefficients zero) for the case where we formulated the fitting problem with $y(x) = 0$. In order to circumvent this limitation, we select a term and move it to the other side of the equation (negate it), assign its coefficient a constant (unity) then solve for the remaining terms.

Note that we have no a priori knowledge of which term should be removed from the basis function. Also, note that selecting the value of the constant has the effect of scaling the entire equation (which scales the effect of noise). We have no a priori knowledge of what would be a good choice for that scaling. We also lack knowledge of which terms will be zero, and removing zero-valued terms from the basis has no useful effect on the solution of the fitting problem (we have a problem because of the zero in the equation).

We opted for an approach that solves reduced basis functions. We iterate through the terms in the original basis function. For each term, we construct a reduced basis function which is simply the original function with the term removed. The coefficient of the term in question is set to 1, and the system solved. We then examine the resulting surface fits and determine which solution fits the data best. We determine goodness-of-fit by summing the distance of each point in the sample from the surface we have found. The surface with least error is the solution of choice.[3]

Experiments in adding noise to the surfaces showed that as the noise increased, the number of correct solutions found decreased, but that degradation of the algorithm was gradual. Sufficient experiments were not performed to precisely quantify the degree of noise permitted.

Plane Fitting

We used a lower-order form of the same equations for fitting planes. Recall that the general equation is $ax + by + cz + d = 0$. In this case, we chose to use $X_1(x) = 1$, $X_2(x) = x + y + z$, $X_3(x) = x - y$, $X_4(x) = y – z$ as the basis functions. This choice was forced by circumstance. We encountered scanned data where the scanner was aimed directly at a flat reference surface x = 0. This is a very degenerate case: all of the x coordinate data is zero, the y and z are arbitrary, but with zero coefficients *b* and *c*, and the constant *d* is only correct if zero. If we use the simple formulation, we cannot find an appropriate term to remove from the basis, because every term ends up zero in one way or another. The alternate formulation forces more terms to be non-zero, permitting the fitting to occur.

The specialized plane fitting algorithm was developed because the quadric fitting algorithm did not simplify well to planar fits. The effect of noise permitted nonlinear terms to be found by the fitting algorithm much sooner for the quadric formulation than was experienced with the specialized planar fitting algorithm. In practice, both algorithms are used, but only non-planar quadrics are sought for comparison to the specialized planar solutions.

Determination of Surface Type

[Zwillinger96] provides a recipe for identifying the type of quadric given a quadric in general form. The approach involves constructing a matrix form of the quadric and its first derivative, computing matrix ranks, eigenvalues and determinants, and performing a table lookup based on matrix ranks, sign of a determinant, and the number of sign changes in the eigenvalues. In

[3] Note that the SVD produces a measure of error, χ^2, which could be used to compare solutions. Attempts to compare solutions based on that error measure were not sufficiently reliable; this problem might be due to the immature state of our algorithms at the time those attempts were made.

practice, noise in the parameters can cause near-zero conditions for some of the matrix parameters, making the determination of surface type sensitive to noise.

Our solution involved determining surface parameters for each surface type of interest, independent of classification. The errors associated with each surface type were measured and the best fitting surface was chosen. In cases where the parameters were clearly unreasonable (e.g. negative radius), no surface was constructed for comparison.

Determination of Surface Parameters

Information on determining the values of surface parameters was provided by [Levy00]. The center of central quadrics is found by taking the partial derivatives of the quadric equation, setting to zero, and solving for the center (x_0, y_0, z_0) of the quadric. This solution is equivalent to determining the solution of the matrix equation

$$\begin{bmatrix} a & h & g \\ h & b & f \\ g & f & c \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = 0$$

The principal axes and parameters related to those axes can be determined directly from an eigenvalue analysis of the matrix of coefficients above. Each eigenvector is a dominant vector of the quadric; its associated eigenvalue is the square of the reciprocal of the associated parameter (i.e. radius) in that direction. A case-by-case analysis of each interesting quadric is required to extract the relevant parameters.

Surface-Fitting Based Segmentation

Our approach to segmentation is very similar to [Besl88]. Our work directly uses the surface fitting algorithms to drive segmentation. The approach is simple – take a small collection of points, fit a surface, and attempt to extend the surface by adding points that belong to the initial group. It is necessary to begin with a sufficiently large group to permit the surface fitting algorithm to operate (Nyquist criteria), while at the same time carefully avoiding the possibility of crossing edges of the object. We try a number of starting places within a geometric region. We take the best grouping and grow surfaces from it, if we find a sufficiently accurate fit.

Problems with initially choosing too small a patch for fitting can result in a surface that is self-limiting. Consider a cylinder for which we choose a collection of points that are sufficiently small that a plane is fit. In growing the surface, points that would encourage the cylindrical fit to be favored are rejected as they don't belong to the plane that was originally fit. For this reason, we favor initial patches that are significantly larger than the surface-finding algorithm requires. This requirement drives us towards fine sampling of points. It also can produce problems in regions having many very small surfaces. In such cases, we have to acknowledge that tiny surfaces are beyond recognition.

Each group of points is associated to a solid model face, by attribute attachment. The error associated with each point can be conveniently stored along with the point. This information permits querying of the resultant model to determine whether and where there are opportunities for more detailed scanning, should a higher degree of accuracy be required.

Grouping close points for beginning the segmentation process requires an efficient model for determining proximity. An efficient model for this is traversing a triangulation of the point set to locate close points. The scanning hardware itself can suggest appropriate models for triangulation. If multiple scans must be triangulated, something like Delaunay triangulation is appropriate. The triangulation reduces the search for adjacent entities to an O(1) algorithm, which is necessary for propagation to be fast.

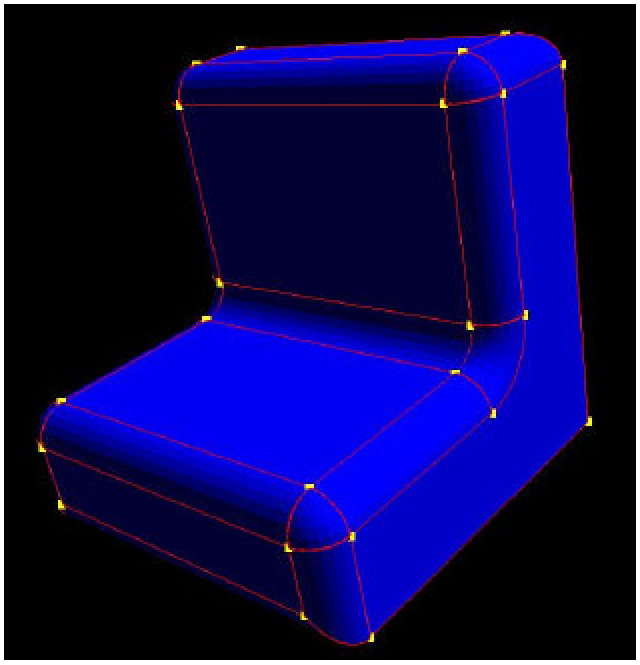

Figure 8: CAD model used for generating synthetic data.

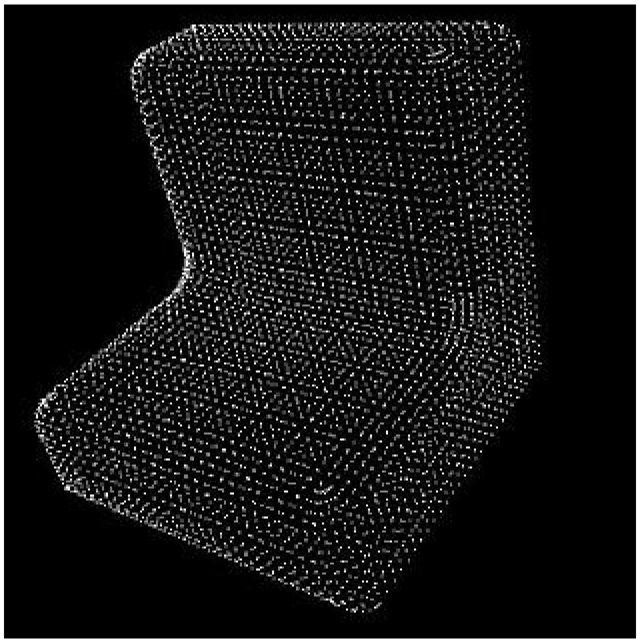
Figure 9: Synthetic data for CAD model

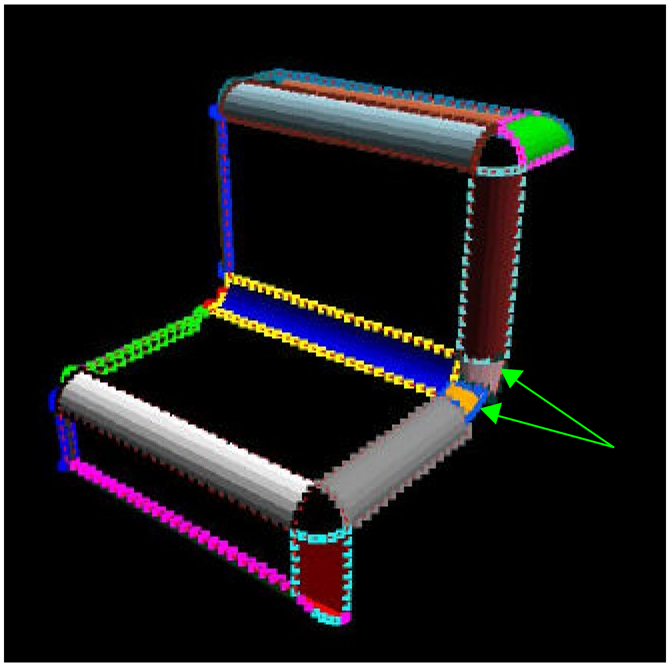
Figure 10: Cylindrical surfaces automatically segmented and converted to CAD representation. The arrows indicate an incorrectly classified toroidal surface.

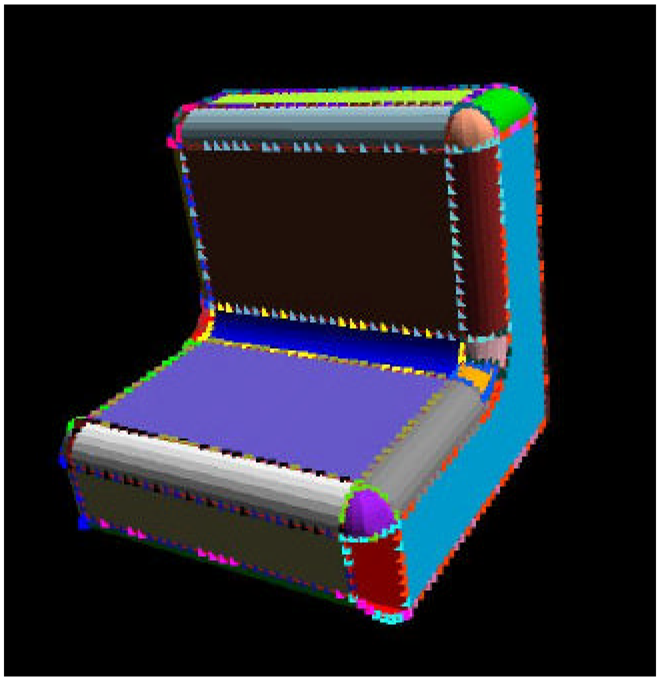

Figure 11: Planar, cylindrical, and spherical surfaces automatically converted to CAD representation.

In Figure 8 is an image of a CAD model used for generating synthetic data. The synthetic data was created with the CAD system's faceter. Noise of varying degrees was added to the point data. In Figure 10 we see an image of the cylindrical faces that were automatically recognized, segmented, and converted to CAD representation. Note the presence of cylindrical surfaces where there were tori in the original model. This error is due to the lack or torus recognition capability. Figure 11 shows the complete model, including planar, cylindrical, and spherical surfaces. Except for the tori, the model is correct, having the same geometry and topology as the original.

Constructing a Solid

After the scanned data has been segmented, and surface equations determined, it is necessary to construct the closed solid model, wherever possible. For this to be possible, it is necessary to determine adjacency between faces, and to determine edge equations.

Face adjacency is easy to determine if points have notions of pointset membership and adjacency. Pointset membership can be trivially implemented by including a pointer in the data structure for each point. As points are included in point sets corresponding to faces, their backpointers are set. Including backpointers to the triangles each point belongs to permits easy searches for nearby points. Neighboring faces are computed by locating boundary points (points with neighbors that don't belong to the face) and finding which pointset neighboring points belong to.

Approximating edges can be constructed from the edges of the triangulation. Such a construction will, of course, contain scanned artifacts, but this is of necessity in regions where the scan is incompletely closed. Boundaries constructed directly from scanned data are specifically marked as being approximate both for use by downstream applications and for use in fitting together scans from multiple viewpoints. These fictitious boundaries are principal candidates for regions in which data from multiple views will be merged.

"Exact" edge equations can, in principal, be determined by intersecting the surface equations of neighboring faces. Tangent surfaces pose a special problem, as it is possible for tangent surfaces to lack any intersection. Tangent edges can be computed on a case-wise basis (e.g. plane/cylinder, cylinder/sphere), but for sufficiently large gaps the resultant solids will not survive many of the geometric operations they might have to endure (e.g. Boolean intersections). We began work in this area, completing the direct intersections, but were unable to work on tangent intersections. Loops constructed from these computed edges were unfortunately unavailable at the end of the project for technical reasons.

Bounding loops of faces are simply computed by locating bounding edges and constructing minimal loops via a minimal spanning tree algorithm. Note that these loops can be constructed out of a combination of exact edges and scanned-data edges, depending on whether the scanned data forms a complete model.

Note that the solids created here might be incomplete, due to occlusion. It may not be possible to scan from a sufficient number of views to find closure faces if the object being scanned cannot be adequately manipulated. It is also possible to have a problem segmenting between separate objects. Neither of these subjects were adequately investigated due to funding shortfall.

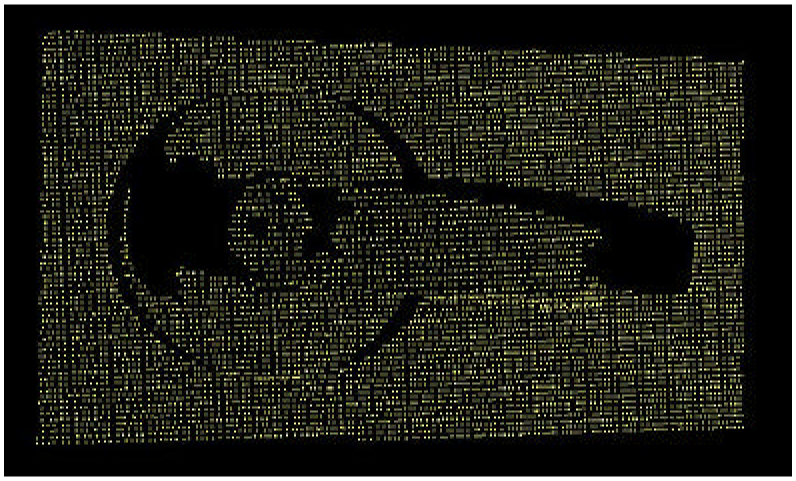

Figure 12: A less-specular doorknob than the previous example. This model has 7596 points, and 15173 triangles.

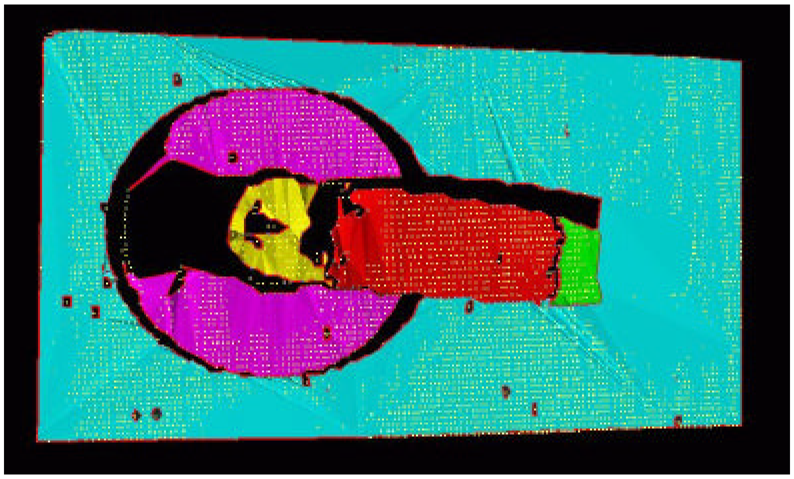

Figure 13: Automatically constructed "solid" representation of the doorknob. This model has 5 faces, and 716 edges, due to use of scanned-triangle edges rather than exact curve boundaries.

In Figure 12 we see the scanned representation of a doorknob. The scanner was facing the doorknob directly, so there are no side faces present. Figure 13 shows the CAD representation that our algorithms are currently capable of creating automatically. Note the ragged edges around the faces – these are constructed directly from scan triangulation edges. The lack of side faces in this model would prevent most intersection edges from being constructed anyway.

The apparent wrinkles in the renderings are due to interaction between the faceting algorithm and the approximate edges. The scan edges have not been adjusted to fit the recognized data (they are raw scan data), so even the facetization of the planar surface shows scan artifacts.

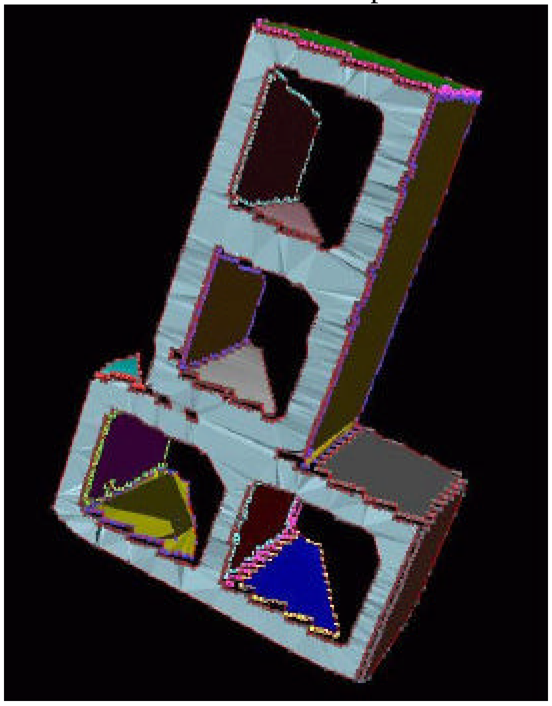

Figure 14: Two cinderblocks. Note that the cinderblocks are not segmented from each other, due to lack of sufficient information in the range data. Additional information is necessary to complete the segmentation.

Searching Previously Defined Models

Clearly, for constructing a complete solid it is necessary to have data from a sufficient number of views to permit construction of the complete manifold. Where such complete data is unavailable, we have the choice of working with an incompletely closed model, searching for previously defined models, or extrapolating the model to achieve closure.

We began work on algorithms for searching to locate previously modeled CAD data. The search involves extracting defining parameters (e.g. plane normal, cylinder axis) from surfaces, edges, and/or vertices, and comparing the scanned model to a collection of previously developed CAD models.

The most reliable information from the scan is surface information. The surface equations are constructed from fitting large numbers of points, negating a significant amount of noise. Noise is still present, and both surface classification and surface parameters can be affected. Edge information less reliable, as it is constructed from intersections of surfaces – if the surfaces are inaccurate, the edges are likely to be more inaccurate. Moreover, edge information may simply be absent -- for incomplete visibility, it is possible that no surface/surface intersections are available for computing edges. Vertices are even more problematic, as we require the presence of three or more faces to properly define a vertex. The point-cloud-based model is thus incomplete and somewhat inaccurate.

Solid models contain many vertices that are arbitrarily placed, such as the vertex linking the start of a periodic curve to its end. The placement of such a vertex is either dictated by the parametric representation of the curve, or might be entirely arbitrary. Comparison algorithms must be careful such artifacts in vertex, edge, and surface data. Edge and face data in CAD is otherwise quite reliable. CAD models frequently show surface, curve, and vertex correspondence of $10e^{-6}$, far better than we expect from scanned data. CAD data can exhibit strange construction artifacts, such as small slivers. It is necessary to confine our comparisons to the largest faces to avoid searching for modeling artifacts.

We are left with the comparison of sketchy, incomplete data to precise, likely completely closed models. Also, the largest, most important faces in the CAD model might be those that were occluded in the scanned data.

We began an algorithm for making these comparisons. We extract surface descriptors: plane normal and origin (corrected for uniqueness by passing a line through the origin); cylinder radius and axis; sphere center and radius; and cone origin, axis, and half angle. These descriptors are associated with the surface area of each face in the model, as well as with a representation of adjacency.   The descriptors are constructed for both models. We sort the faces in the point-cloud-derived model, and begin from the largest. We search the CAD model for similar faces, and seek as soon as possible to orient the model. Continue matching faces until we have a reasonable match, or until failure occurs.

The searching algorithm worked reasonably well for a small sampling of surfaces, but has not been exercised with a large number of CAD models. We created a small database of CAD models of objects, including the object in Figure 15, and searched for the doorknob in Figure 13. The doorknob was correctly found, even though we lacked any side face information. Planar faces in such views are a significant problem, as they are difficult to orient without the presence of orthogonal faces (which might be absent in a single view) or reasonable edges (which can only be approximated without the presence of adjacent faces).

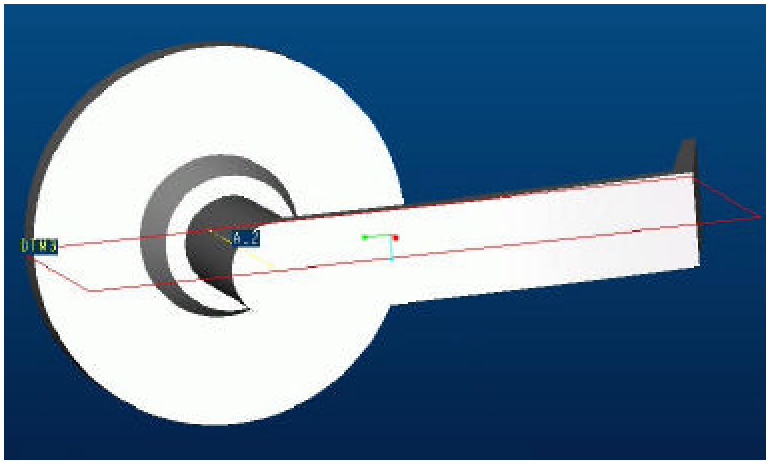

Figure 15: Pro/Engineer model of doorknob located when searching for a match to the doorknob in Figure 13.

Conclusions

This work pursued the development of a system for converting scanned range data into CAD models with specifically tailored surface representations and minimal topology. The work was successful in producing reliable surface fitting algorithms and segmentation based on those fitting algorithms. Construction of solid model topology and edge geometry was partially successful. An algorithm for searching previously created models was demonstrated.

During the course of this work, significant interest was demonstrated from a variety of customers of commercial systems. There continues to be a need for a production-ready version of this kind of capability, as demonstrated by their interest.

Recommendations

Follow-on work to this project includes the following areas:

1. Support for higher order surfaces. Coverage of engineering surfaces must at least include explicit torus and higher-order (NURBS) geometries. Including these surfaces requires that the lower-order surfaces are preferred wherever possible. NURBS support, in order to be most appropriate, should include the provision for the NURBS isocurves to follow directions of greatest curvature, so the NURBS flows with the shape of the object. In order to achieve minimal solid topology, the NURBS surfaces will require extrapolation routines to close over holes and achieve the requisite bounding topology.
2. Recognition of Organic Objects. We are already fielding requests to distinguish trees from ground, and organic from man-made objects. Our recognition algorithms are a first step, but the algorithms are known to run for long times in cases where the geometry is so fractal that no engineering surfaces are recognized.

3. Reduce Sensitivity to Noise. Insufficient time was available for a complete characterization of the algorithms relative to noise. The current algorithms are robust in the face of noise, and appear to degrade gracefully, but more work to guarantee this is warranted.
4. Testing with Many Large Models. Our test suite was significant, but more, larger models are necessary to validate the system.

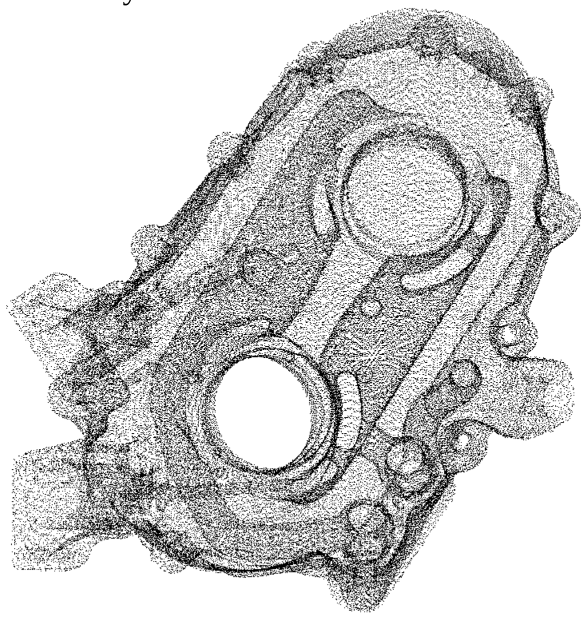

Figure 16: A test case that we were unable to fully address.

5. Speed. Work has already begun to speed up the algorithms, both in terms of selection of initial patches for surface fitting and reducing calculations performed during the search through different basis functions.
6. Finish Direct Construction of CAD Model. The algorithms for computing exact edges need to be integrated into the loop-finding algorithm, and substantial work resides in computing tangent edges. Tangent edge computation, and solid construction overall, can be improved by adjusting surfaces for perpendicularity and tangency. CAD surfaces are rarely nearly tangent or nearly perpendicular without being exactly so, and an algorithm to adjust them will dramatically improve model closure.
7. Recognition Using Additional Sensors, Including Color. Additional physics can be useful in segmenting separate objects from each other, but can be misleading.
8. Delivery in Autonomous and Interactive Environments. The algorithms presented here, as originally conceived, were intended for delivery in an autonomous environment. That delivery has yet to be accomplished.

References

[Allen97]	Allen, P. K., and Reed, M. K., “A Robotic System for 3D Model Acquisition from Multiple Range Images”, *Proceedings of IEEE International Conference on Robotics and Automation*, Albuquerque, NM, April 1997.
[Besl88]	Besl, P. J., Jain, R.C., “Segmentation Through Variable-Order Surface Fitting"” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 2, March 1988.
[Feddema97]	Feddema, J. and Little, C., “Rapid World Modeling: Fitting Range Data to Geometric Primitives”, *Proceedings 1997 IEEE International Conference on Robotics and Automation*, Albuquerque, NM, 1997.
[Levy00]	Levy, S., Personal Communication, March 2000.
[Little96]	Little, C., Wilson, C. W., “Rapid World Modelling for Robotics”, World Automation Congress ‘96 Proceedings, May 1996.
[Pratt87]	Pratt, V., “Direct Least-Squares Fitting of Algebraic Surfaces”, *Computer Graphics*, Volume 21, Number 4, Association for Computing Machinery, July 1987.
[Press88]	Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P., *Numerical Recipes in C: The Art of Scientific Computing, Second Edition*, Cambridge University Press, New York, NY, 1988.
[PTC95]	*Pro/Develop Reference Guide, Release 16*, Parametric Technology Corporation, Waltham, MA, 1995.
[Requicha80]	Requicha, A.A.G., “Representations for rigid solids: Theory, methods, and systems”, *ACM Computing Surveys*, 12(4): 437-464, December 1980.
[Spatial98]	*ACIS 4.0 Online Help CD*, Spatial Technology Corporation, Boulder, Colorado, 1998.
[Zwillinger96]	Zwillinger, D., ed., *CRC Standard Mathematical Tables and Formulae, 30th Edition*, CRC Press, Boca Raton, Fl., 1996.

Distribution

Internal Distribution:

MS1002 P. J. Eicker, 15200

MS1010 M. E. Olson, 15222

MS1010 A. L. Ames, 15222, 20 copies

MS1010 D. M. Hensinger, 15222

MS1010 D. K. Kholwadwala, 15222

MS1008 P. G. Xavier, 15221

MS1003 C. Q. Little, 15211

MS1003 C. Lewis, 15211

MS1004 R. R. Peters, 15221

MS0188 LDRD Office, 1030

MS9018 Central Technical Files, 8945-1

MS0899 Technical Library, 9616, 2 copies

MS0612 Review & Approval Desk, 9612 For DOE/OSTI