Interfacing Reverse Engineering Data to Rapid Prototyping

N.K. Vail, W. Wilke, H. Bieder, and G. Jünemann

Daimler-Benz Research Center
Optische Meßtechnik / Modelltechnik
Ulm, Germany D-89013

Abstract

Rapid prototyping has become an increasing part of product development process chains resulting in reduced time to market and reduced development costs. As manufacturers strive to further reduce development cycles to maintain market competitiveness, the use of reverse engineering technologies have started to play key roles in the product development cycles. Integration of these technologies into existing development cycles provides tools to maintain design integrity during development stages as well as between successive product lines. One aspect of reverse engineering is the interfacing of data obtained from these technologies to manufacturing processes such as rapid prototyping. This paper discusses work at Daimler-Benz to develop a set of interfacing tools as part of a larger reverse engineering process loop. These tools include facilities to generate contiguous surface meshes from a collection of measured views as well as automatic feature detection and hole closure.

(Keywords: Reverse Engineering, Rapid Prototyping, Mesh Generation, Point Clouds)

Introduction

As a developer and manufacturer of products targeted for the global marketplace, Daimler-Benz is committed to the development of methods that reduce both the cost and time of developing these products. Reverse engineering (REEN) technologies are beginning to playing a large role in helping to achieve this goal. This stems from the capabilities of REEN technologies to provide support to a broad range of applications such as continual maintenance of CAD data, on-line inspection techniques,[1] and virtual reality.

Within this framework we are developing the process loop shown in Figure 1. This reverse engineering process loop is seen to encompass all elements of the product development cycle. This process loop describes the flow of information from virtual world of the product description through manufacturing to the physical product and back to the virtual world through REEN technologies. At the peripheries of the central loop are the capabilities to support the input and output of conceptual and physical information. To realize this process loop we are actively involved in the research and development of the necessary technological aspects including optical measurement techniques, sensor navigation, data manipulation, and CAD support. At the heart of this system is a software package that is capable of performing tasks of the reverse engineering process loop such as sensor navigation and control, data acquisition and processing, CAD reconstruction, and interfacing to manufacturing processes. The complete system is modular in design and intended to be independent of target platforms whether they be the sensor device, navigation mechanism, or computing platform.

As indicated in Figure 1, interfacing reverse engineering data to manufacturing processes such as rapid prototyping (RP) technologies, is an important link in the realization of an effective reverse engineering process loop. Current implementations of RP technologies offer capabilities to fabricate functional objects using geometric data from a variety of sources. Traditionally the geometry data used by RP technologies have been derived from CAD information, either volume or surface element based. However, as a result of reverse engineering technologies, non-CAD based surface and volume coordinate data are being increasingly encountered in commercial development and manufacturing applications. In some areas, such as medical applications, only non-CAD based data exist. In this paper we will describe our efforts to link reverse engineering data to RP technologies.

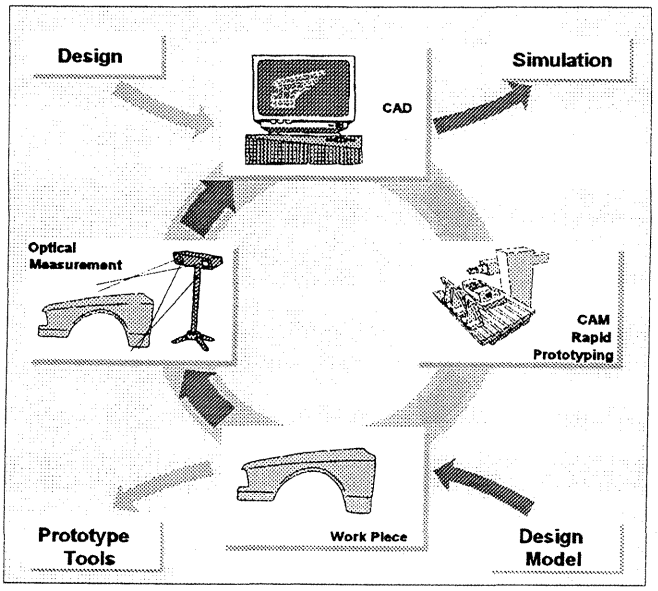

Figure 1. Reverse engineering process loop.

Reverse Engineering Data

Reverse engineering data can come from a variety of sources. Common examples include optical sensor data from devices such as structured light projectors,[2] laser range finders,[3] and photogrammetry methods. Other sources include tactile devices, computed tomography (CT),[4] non-destructive test methods, and finite element analysis. A complete discussion of the various data sources is beyond the scope of this paper. We will focus on data obtained from optical sensors and will note when data comes from other sources.

The initial step in any REEN operation is to obtain information about the geometry of an object. This is accomplished using optical sensors that project a series of structured light patterns onto the object and then, using a digital sensor, capturing each of the reflected patterns from the object, Figures 2 and 3. Using the coded information of the projected patterns and the geometry of the sensor system it is possible to determine the spatial position of a point on the object. The result is a matrix image of spatial points in the coordinate space of the sensor. This image is termed a view and can contain several thousand coordinate points. Generally, between five and fifty views may be required to obtain a complete description of an object. Each of the views in turn must be transformed into a common coordinate space. In current practice this transformation is accomplished by either mounting the sensor on a positioning system or by semi-automatic software registration. The final result is a cloud of points that describe the visible surface of the object, Figure 4. Representative point clouds can contain on the order of a few hundred thousand data points. The uncertainty of a data point is approximately ± 0.1mm in all directions.

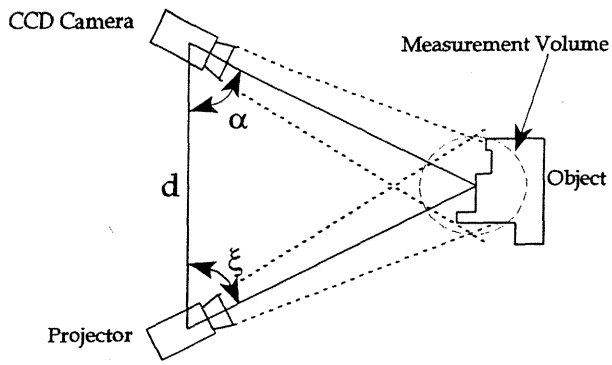

Figure 2. Optical measurement geometry.

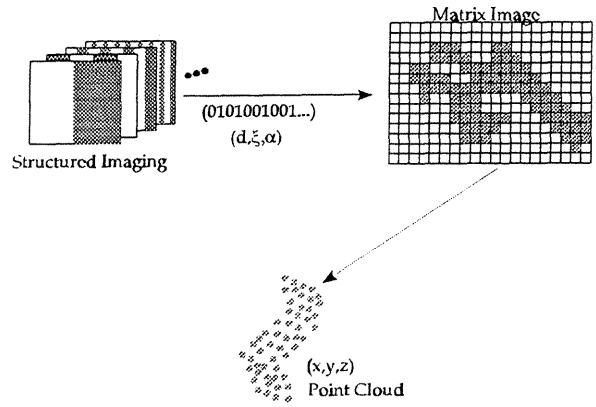

Figure 3. Construction of 3D point cloud data.

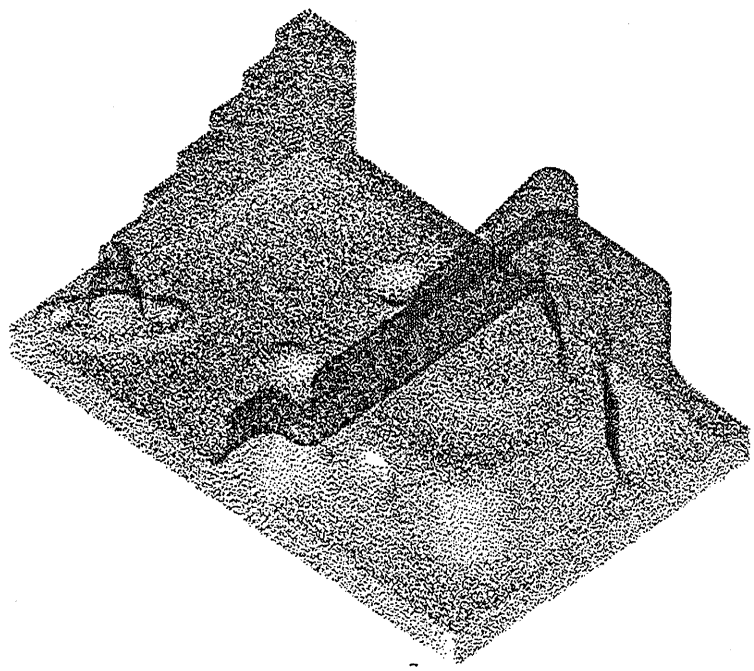

Figure 4. Point cloud obtained from optical measurement of the IMS-T1 standard test part. Approximately 250,000 points and ten views.

Requirements for Reverse Engineering Data Reduction

The intent is to create a data set suitable for input to rapid prototyping systems. Accomplishing this task requires transforming the point cloud with minimal computational time and effort while remaining faithful to the original data.

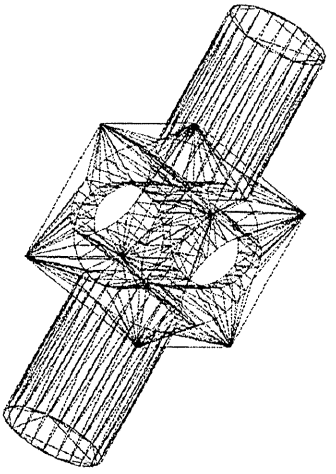

Figure 5. STL triangular mesh.

Rapid prototyping methods may vary in their implementation but nearly all use the same geometry input format, the STL triangular mesh format introduced by 3D-Systems,[5] Figure 5. Some RP platforms support more standard 3D-CAD object formats such as IGES. Still other platforms support input of geometric information corresponding to object contours which then translate directly into material slices. Contour interfaces are generally supported through public domain formats such as SLC supported by 3D-Systems,[6] the CLI format developed through an initiative of the European Community,[7] and the familiar HPGL format used by many laser printers. Because the STL format is the *de facto* standard, it is the logical form of the reduced REEN data. Before proceeding, however, let us discuss the other input formats.

To use 3D-CAD information as a source for RP processes requires a complete reconstruction of this information from the REEN data. While the reconstruction of 3D-CAD information is one of the primary goals of the reverse engineering process loop, complete reconstruction of 3D-CAD information is still labor intensive and the effectiveness of the reconstruction is dependent on model complexity. Certain aspects of 3D-CAD reconstruction such as recognition of regular geometries or global fitting of data by spline surfaces can be automated. However, considerable post-processing may be required to achieve a valid geometric representation. These operations are time intensive. In addition, the reconstructed model must be further processed to produce a triangular mesh model or sliced to yield contour data. Reconstruction of 3D-CAD data appears an unnecessary step as it would be more efficient to directly reconstruct the triangular mesh from the cloud data.

To use contour information as input to RP processes requires the extraction of this information from REEN data. This step is straightforward. Additionally, some data sets, such as those obtained by computed tomography (CT), are produced as contour information. However, contour data sets can have certain drawbacks in actual practice. A major drawback is the need for interpolation when processing contour data sets. If the contour data is extracted from a point cloud the possibility exists that the point density of the cloud is insufficient to produce a correct contour using the slice resolutions of RP processes. This is illustrated in Figure 6 which shows contour data obtained at a slice resolution of 1.0mm. The points are a result of all points in the cloud occurring in the slice volume element. In this case, the left part of the contour is quite good as this section of the object contains walls perpendicular to the slice plane. However, the right side of the contour is questionable since the object contains a sloped face in this region. Considerable smoothing as well as adjacent contour interpolation would be required to obtain an adequate contour description. It is clear from this example that a finer slice resolution would result in a poorly defined contour. Irrespective of slice resolution, the resulting contours would be a minimal approximation at best. It appears that a better approach would be to consider cloud data in a global manner thereby creating a better overall approximation of the object.

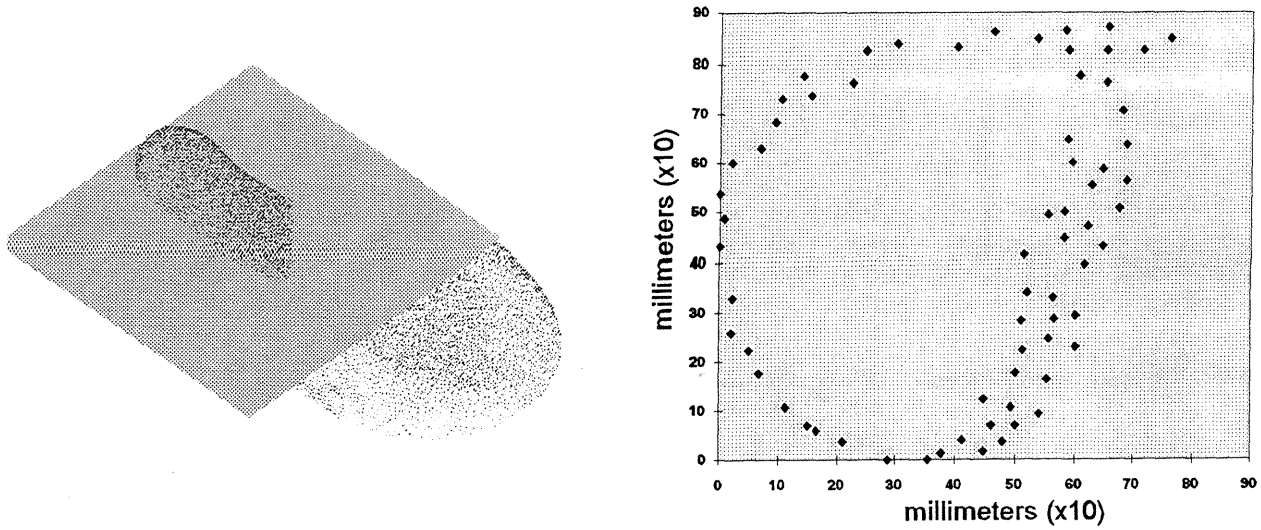

Figure 6. Contour data extracted from the point cloud on the left using a volume slice element thickness of 1.0mm.

Given the apparent drawbacks of using reconstructed 3D-CAD information or extracted contour information it is reasonable to consider the generation of triangular meshes directly from REEN data. Creating a valid triangular mesh is the first requirement of our approach to interfacing REEN data to RP processes. In practice, a triangular mesh is simply a piece-wise linear surface approximation composed of triangular facets, Figure 5. The creation of simplicial surfaces of this type from arbitrary point data has become a topic of considerable interest in recent years.[8,9,10,11,12] Simplicial surfaces are not only important as input for RP processes, but they also have application in the fields of graphical rendering, virtual reality, and finite element modelling.

The criteria for triangular mesh generation is the ability to create the mesh with a minimal set of input parameters and reasonable execution times for all data sets. Additional requirements are: efficient, minimal user-interaction methods for mesh verification such as residual repairs of the generated mesh; normal orientation; and surface projection of incomplete meshes to create closed model volumes. There is currently no algorithm available in the literature that meets all of these criteria. The closest is perhaps the Rapid Prototyping Module available in recent versions of Surfacer. However, we have found this mesh generation algorithm and its accompanying utility algorithms for mesh verification to be poorly documented in operation and insufficient in practice.

Methods

The method we have chosen for mesh generation is a modified Delauney triangulation algorithm with elements of the marching cubes algorithm. Each of these algorithms are discussed extensively in the literature.[8,13] In our approach the point cloud is treated as a contiguous data set. The underlying image matrix information for each view in the point cloud is presently not stored as part of the point cloud.

With a contiguous point cloud it is convenient to store the point data as a set of ordered voxels to optimize processing during visualization and other operations. Descriptions of voxel structures can be found in the literature.[8,14,15] Currently we use two criteria for storing the point cloud as a voxel representation. In both cases voxels void of point cloud data are not stored. The simplest method is to create voxels of fixed size containing a maximum number of data points. The other method is to create voxels by recursive subdivision using the criteria that each voxel will contain a set of points with a minimal error to a best fit plane. This effectively segregates the point cloud into voxels of different sizes that contain nearly planar elements. One advantage of this last method is it tends to decrease the number of voxels, thus reducing processing overhead.

Our algorithm then uses a Delauney method to triangulate the data in each voxel. Delauney triangulation is defined for points in 2D space. Therefore, the algorithm creates a best fit plane of the voxel data onto which the voxel data points are projected. The plane is clipped by the voxel boundaries. The projected points on the best fit plane are sorted according to their distance from the center of the plane, thus optimizing execution. The criteria for Delauney triangulation is based on the idea that given a set a three points describing a triangle, a circumscribed circle of the triangle should not contain any additional points. If the circle contains points then the triangle must be split. However, to split a triangle properly one must check for adjoining triangles in order to avoid introducing topological errors. In our implementation, the method tends to grow a patch of triangles from the center of the best fit plane. When all triangles of the patch meet the Delauney criteria then triangles connecting the patch to the boundary points of the clipping plane are removed and a final post-processing check is done to remove boundary triangles with large angles. As long as all points in a voxel can be projected onto the best fit plane and the topology is not complex, the Delauney method will yield a valid triangulation devoid of redundant triangles and will not be self-intersecting.

After each voxel is triangulated it is merely a matter of stitching neighboring voxel patches together to obtain a contiguous mesh. Presently this step can introduce topologically incorrect triangles in areas of rapidly changing topology. These spurious triangles can be easily removed by a post-processing step to yield small gaps in the mesh. Gaps in the mesh can also occur due to sparse or missing data, excessive noise, or misalignment of the individual views. Because of the mesh generation method it will be necessary to harmonize surface normals. We have incorporated additional algorithms to handle these problems in both automatic and semi-automatic manners. For example, proper orientation of normals is an automatic feature. One merely determines the largest z-coordinate and propagates normals by traversing the minimal spanning tree that is inherent in the mesh structure. Other algorithms such as redundant triangle and mesh self-intersecting detection can also be automated. Gap closure can also be automated but we provide the opportunity for the user to inspect the gap before proceeding with the closure. We also provide the capability to create an offset surface of a triangular mesh as a means of creating a solid suitable for RP fabrication. This is necessary as we often encounter thin freeform surfaces.

Experiments

In this section we show some results of producing triangular meshes from reverse engineering data. First, as a comparison let us consider the point cloud data of the familiar IMS-T1 part[16] shown

previously in Figure 4. This data set was converted to a surface model using algorithms developed at Daimler-Benz, Figure 7. This process required approximately eight hours and resulted in a surface model that could be used to generate a triangular mesh, Figure 8. However, mesh generation was not without problems. This is because the correctness of the reconstructed surface model depended heavily on the order in which fitted surfaces were extracted from the point cloud. Surface extraction order effects both how clipping planes are generated and how much data remains available to describe the original topology. Therefore, the generated mesh contained many errors including overlapping sections, non-joined sections, and secluded volumes. To repair these errors required about eight hours of intensive interaction. In contrast, Figure 9 shows a mesh generated directly from the same point cloud data. Time to complete the initial triangulation was only a few minutes on an SGI Indo Workstation. An additional one hour was required to complete and validate the mesh structure. Correctness of the resulting topologies can be seen in Figure 10. Here we show contours extracted from the given generated meshes compared to a contour extracted from the original CAD model. The contours shown are from the topologically complex stair region of the IMS-T1 part. The contour from the mesh generated from the surface reconstruction is inaccurate while the contour from the mesh generated from the point cloud data is more true to the original data.

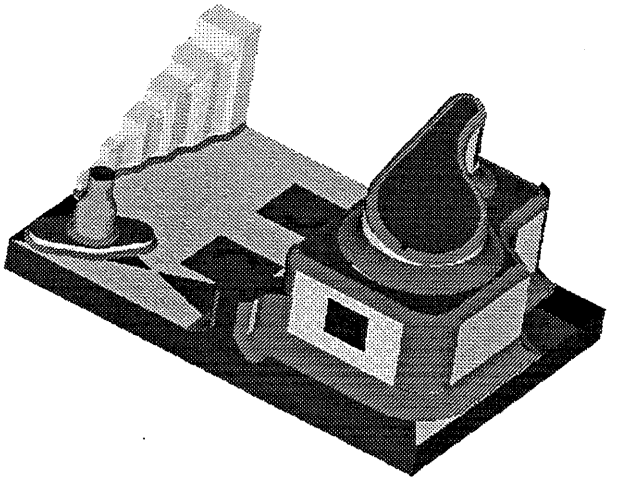

Figure 7. Surface model obtained from the point cloud data in Figure 4.

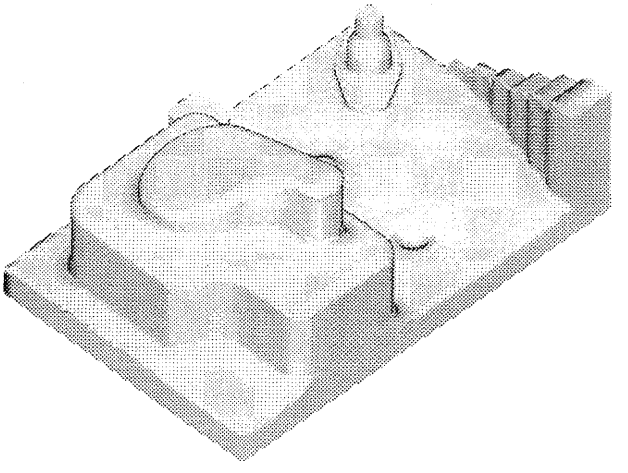

Figure 8. Triangular mesh obtained from the surface model in Figure 7.

Lastly, we show the familiar Mercedes-Benz star, Figure 11. This object is relatively small (~10cm diameter and 0.5cm in thickness) and, at first glance, seemingly simple in geometry. However, the star has many small faceted features of widely varying angles. To get an adequate description of this object required twelve views resulting in ~100,000 data points. Generating a mesh from these data was slightly more difficult resulting from the many fine features because many of the surfaces of the object meet to form small acute angles. The point cloud data in these fine regions become indistinguishable as the planes converge because of the small distances and inaccuracies in the measurement. As a result, the meshing algorithm generates erroneous triangles in these regions, thus leading to a invalid mesh. Repairing and closing this mesh required about one hour. The closed mesh is seen in Figure 12.

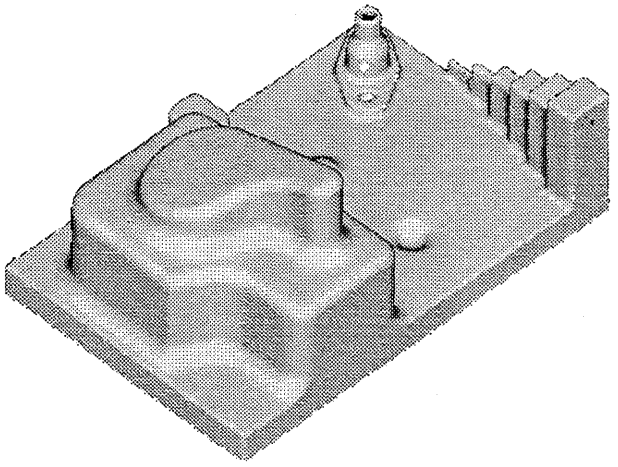

Figure 9. Triangular mesh generated directly from the point cloud data shown in Figure 4.

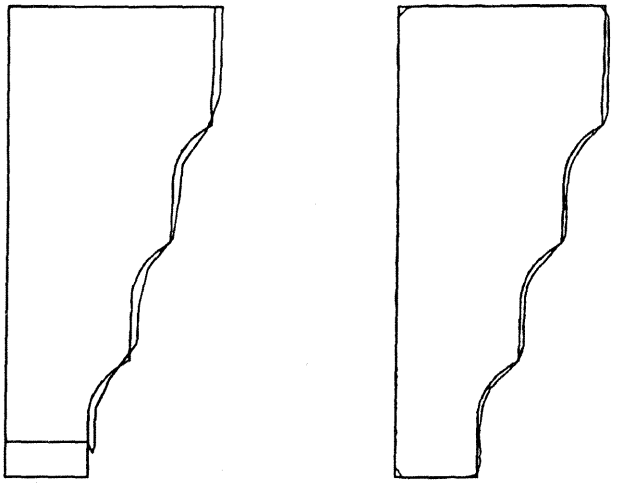

Figure 10. Comparison of contours extracted from the original CAD model and the meshes generated from the reconstructed surface model (left) and generated directly from the point cloud data (right).

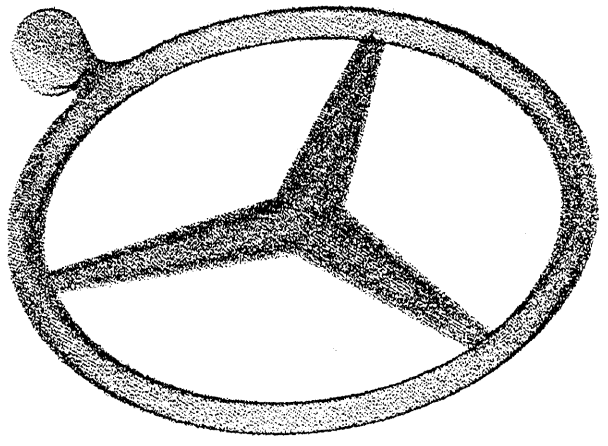

Figure 11. Point cloud data for the Mercedes-Benz star. Data is the result of twelve views yielding ~100,000 data points.

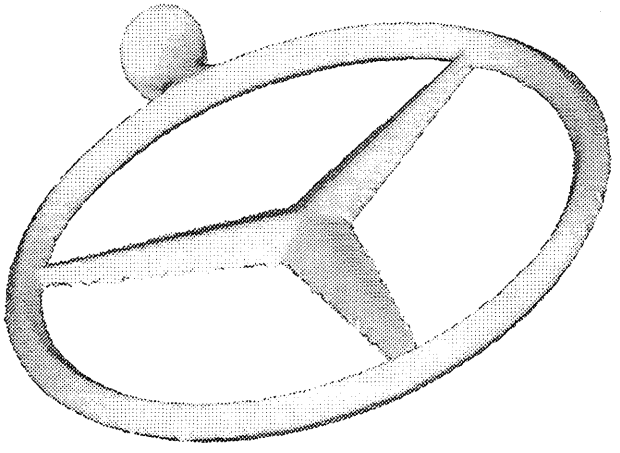

Figure 12. Triangular mesh of the Mercedes-Benz start obtained from the point cloud data in Figure 11.

Conclusions and Future Work

In this paper we have presented our approach to creating a triangular mesh suitable as input to solid freeform manufacturing process. This algorithm, along with a series of verification utilities, has been incorporated into the central software module of the reverse engineering process loop being developed by Daimler-Benz for use in product development and manufacturing applications. The mesh generation software in its present form adequately produces a representation that is faithful to the original point cloud data. The associated verification software has been used successfully to verify and repair meshes generated by our software as well as meshes generated by other software packages.

Currently our meshing algorithm does not handle certain types of data very well. Particularly troublesome is data with rapidly changing topology or data with excessive noise. Complex gaps in the mesh as well as the stitching of voxels still produce invalid triangles in areas of rapidly changing topology.

In the future we would like to perform as much data reduction as possible prior to mesh generation. One approach is to perform automatic geometric feature recognition to reduce the point cloud to a minimal representation of geometric objects and freeform surfaces described by the remaining point cloud. These elements are easily meshed using the algorithm described in this paper. In addition to data reduction, we are also investigating methods of mesh reduction.

Acknowledgements

The principle author would like to thank the Alexander von Humboldt Foundation for financial support to do this work.

References

1. Fürderer, K., "Projekt BDW: Beulen, Dellen, Welligkeiten. Der Kundenmaßstab", Internal Technical Report #TEP-94-016, pp. 18, (1994).
2. Post, D., Han, B., and Ifju, P., High Sensitivity Moiré, Springer-Verlag, New York, 1994.
3. Godin, G., Boulanger, P., and Rioux, M., "Direct Replication of Objects using 3-D Geometric Imaging and Rapid Prototyping", *The Fourth International Conference on Rapid Prototyping*, 159-168 (1993).
4. *Product Brochure*, Scientific Measurement Systems, Inc., 1995.
5. *Stereolithography Interface Specification*, 3D Systems, Inc., June, 1988.
6. *SLC File Specification*, Version 2.0, 3D Systems, Inc., November, 1994.
7. *Common Layer Interface (CLI)*, Version 2.0, Brite/EuRam Project #BE-5278 and #BE-5930, 1994.
8. Lorenen, W.E. and Cline, H.E., "Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm", *Computer Graphics: Siggraph '87 Conference Proceedings*, **21** [7], 163-169, 1987.

9. Hoppe, H., DeRose, T., Duchamp, T., Halstead, M., Jin, H., McDonald, J., Schweitzer, J., and Stuetzle, W., "Surface Reconstruction from Unorganized Data Points", *Computer Graphics: Siggraph '94 Conference Proceedings*, **26** [7], 71-78, 1992.
10. Chen, Y. and Medioni, G., "Surface Description of Complex Objects from Multiple Range Images", *Computer Society Conference on Computer Vision and Pattern Recognition*, June 1994.
11. Roth G. and Wibowo, E., "A Fast Algorithm for Making Mesh Models from Multi-View Range Data", In *Proceedings of the DND/CSA Robotics and Knowledge Based Systems Workshop*, St. Hubert, Quebec, October 1995.
12. *Surfacer V5.0 Manual*, ImageWare, Inc.,
13. Bern, M. and Eppstein, D., "Mesh Generation and Optimal Triangulation", in *Computing in Euclidean Geometry*, D.-Z. Du and F. Hwang, Eds., Lecture Note Series on Computing, **1**, 23-90 (1992).
14. Mäkelä, I. and Dolenc, A., "Some Efficient Procedures for Correcting Triangulated Models", *Solid Freeform Fabrication Symposium Proceedings*, **4**, 126-134 (1993).
15. Ashdown, I., "Octree Color Quantization", *C/C++ Users Journal*, **13** [3], 31-44 (1995).
16. IMS - Intelligent Manufacturing Systems, International Conference on Rapid Product Development, Stuttgart, Germany, 1994.