Proceedings of DETC/CIE 2004:
Design Engineering Technology Conferences & Computers in Information and Engineering Conference
September 28-October 2, 2004, Salt Lake City, Utah

DETC2004-57475

SEGMENTATION OF NOISY LASER-SCANNER GENERATED MESHES WITH PIECEWISE POLYNOMIAL APPROXIMATIONS

Miguel Vieira[1] **Kenji Shimada**[2]

Department of Mechanical Engineering
Carnegie Mellon University

ABSTRACT

Laser scanners offer a fast and simple way of collecting large amounts of geometric data from real-world objects. Although this aspect makes them attractive for design and reverse engineering, the laser-scanner data is often noisy and not partitioned into meaningful surfaces. A good partitioning, or segmentation, of the scanner data has uses including feature detection, surface boundary generation, surface fitting, and surface reconstruction. This paper presents a method for segmenting noisy three-dimensional surface meshes created from laser-scanned data into distinct regions closely approximated by explicit surfaces. The algorithm first estimates mesh curvatures and noise levels and then uses the curvature data to construct seed regions around each vertex. If a seed region meets certain criteria, it is assigned a region number and is grown into a set of connected vertices approximated by a bicubic polynomial surface. All the vertices in a region are within known distance and surface normal tolerances from their underlying surface approximations. The algorithm works on noisy or smooth data and requires little or no user interaction. We demonstrate the effectiveness of the segmentation on real-world examples.

1. INTRODUCTION

The speed, ease, and accuracy with which laser scanners can acquire digital geometric data of real world objects promises substantial potential for their applicability in different aspects of industrial design. They produce dense point clouds, often triangulated by built-in software, representing the surface geometry of the scanned object. Their practicality makes them desirable for tasks such as quality control, product development, reverse engineering, and rapid prototyping.

However, aside from being meshed into a triangulated surface, the data points are not organized into meaningful sets that could be useful for later processing. A desirable partitioning of laser-scanner data for engineering design would identify surfaces of constant or slowly varying curvature, either blended at their boundaries or connected along sharp edges.

Such a segmentation of laser-scanner data is highly desirable for several reasons. The segmented surfaces and their boundaries could be used to define boundary curves for fitting surface patches to the scanner data. They could also be used to detect feature/character lines on the scanned surface. Finally, the segmented surfaces could be used to automatically create a piecewise-smooth reconstruction of the scanned object.

There are numerous difficulties in segmenting an unstructured noisy surface mesh. Existing segmentation algorithms either are designed for structured, height-field data or rely on curvature estimates and edge detection. However, triangles of widely varying size and shape and a high level of scanner noise make such algorithms unusable for laser-scanner generated meshes.

The goal of this work is to describe a method for segmenting laser-scanner data into distinct surfaces and to generate analytical representations for them. Our algorithm segments the mesh into several regions, each one described by a bicubic polynomial surface. Upon the completion of the algorithm, every vertex in each region lies within a certain distance from the polynomial approximating the region.

The first step of the algorithm is to estimate the curvatures and noise level of the mesh (Section 3). The noise is measured both in terms of distance and vertex normal deviation from an underlying smooth surface. The vertex curvatures are then used to select seed vertices (Section 4), and we attempt to grow regions from each seed vertex (Section 5). If a seed region and its approximating surface satisfy the requirements for a region, it becomes a new region and begins growing (Section 6). Vertices adjacent to a region are added as long as they are compatible with the surface approximating the region. The requisite data structures are as follows. For each vertex, we record the number of the region it belongs to, 0 indicating that a vertex is unsegmented. For each region number, we store the translation and rotation matrices for its local coordinate system and the coefficients of its polynomial approximation. A region is a list of vertices.

[1] mcv@andrew.cmu.edu
[2] shimada@cmu.edu
Correspondence to: Kenji Shimada
The Department of Mechanical Engineering, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh PA 15213

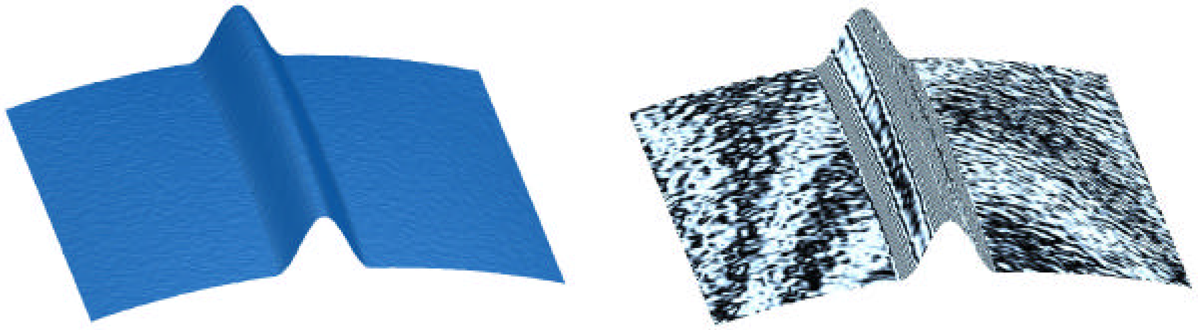


Figure 1. A noisy synthetic surface mesh with simulated reflection lines. This mesh will be used to demonstrate the different steps of the algorithm.

2. RELATED WORK

The mesh segmentation algorithm presented in this paper extends the region growing method first presented by Besl and Jain [1] and further developed in [2],[3],[4] and applies it to noisy, unstructured surface meshes. In the original paper [1], noisy image and range data are modeled as piecewise smooth surfaces. The region growing algorithm then segments the data into regions that can be accurately represented by bivariate polynomials. The algorithm works by first estimating image noise by calculating the distance from each pixel to a locally fit least-squares plane. Then the image is partitioned into areas with constant-sign Gauss and mean curvatures. These areas are contracted to create seed regions containing a few pixels. Bivariate polynomials are fit to the seed regions and they are grown by adding adjacent pixels compatible with the polynomials. Bilinear polynomials are used at first, but the polynomial order can increase up to biquartic when a low order surface can no longer accurately approximate the pixels in a region.

More recently, Mangan and Whitaker [5] adapted the morphological watershed algorithm from image analysis to surface meshes. The idea is to calculate the curvature on the mesh and then find local curvature minima. These minima are labeled and become catchment basins. Then, for each unlabeled vertex, a steepest descent (based on curvature) path is followed until a labeled vertex is reached. The unlabeled vertex is then assigned this label. Once all vertices are labeled, regions are merged with a preference for regions with a larger difference between the highest and lowest curvature vertices they contain. Improvements to the original algorithm in [5] are given in [6] and [7]. Taking a physics-based approach, Wu and Levine [8] segment a surface mesh by simulating the electrical charge density distribution on the surface and then segmenting it along lines of charge density minima. Lessage, et al, [9] segment surface meshes by first detecting sharp edges on the mesh and then growing regions made of vertices of similar curvatures, bounded by sharp edges.

The existing region growing papers [1][2][3][4] are based on gridded height data and their algorithms are not applicable to general three-dimensional surfaces. The recent algorithms for three-dimensional mesh segmentation partition a mesh into distinct regions, but the segments cannot in general be approximated accurately by simple surfaces. The new algorithms are mostly based on curvature estimates or edge detection, which can be unreliable on noisy meshes. Furthermore, as mentioned in [4], such algorithms will likely fail in industrial design applications where part surfaces can consist of smoothly blended surfaces with no sharp edges. Finally, the results of recent algorithms do not give any information for a smooth reconstruction of the part.

In this paper, we extend the region growing algorithm to unstructured three-dimensional surface meshes. We introduce new methods for estimating mesh noise and for region growing on three-dimensional surfaces. Our algorithm works on noisy data with little or no user interaction. Upon completion, the algorithm produces both a mesh segmentation and a set of surfaces that approximate every point in each segment to known tolerances. Figure 1 shows a synthetic surface with noise artificially added. This surface will be used to demonstrate the different steps of our algorithm.

3. NOISE ESTIMATION

The algorithm starts by estimating the noise variance of the mesh. Later, during region growing, we will use the noise level estimates as thresholds for adding vertices to regions. Because there is no ground truth available for comparison with the laser-scanner data, we assume the mesh vertex positions represent noisy samples of a smooth surface. Then, for some small area on the mesh, a biquadratic polynomial approximation should provide a reasonable estimate of the position and shape of the original surface near that area. We choose a biquadratic surface because it is of low order and a planar surface would be unable to differentiate between noise and a smooth, curved surface. Our algorithm approximates the submesh around each vertex with a polynomial surface and calculates errors by measuring the deviation of the mesh vertices from their approximating surfaces.

3.1 Vertex Neighborhood Construction

The algorithm visits each vertex $\mathbf{x}_i$ of the mesh and constructs a small neighborhood of nearby vertices. This neighborhood consists of all vertices topologically connected to $\mathbf{x}_i$ and lying within a certain distance from it. The average length of the N edges incident to the vertex determines this distance:

$$l_{avg,i} = \frac{1}{N}\sum_{j=0}^{N-1}\left\|\mathbf{x}_j - \mathbf{x}_i\right\|. \tag{1}$$

This local calculation of average edge length is more useful than a global calculation because overlapping scans of a part can lead to widely varying vertex densities and edge lengths. To form the neighborhood, we begin by adding vertex $\mathbf{x}_i$ to it and then all connected vertices $\mathbf{x}_j$ whose Euclidean distances from $\mathbf{x}_i$ satisfy

$$\left\|\mathbf{x}_j - \mathbf{x}_i\right\| < w\ l_{avg,i}\ , \tag{2}$$

where w scales the radius of the neighborhood. Increasing w makes the approximation more robust and less sensitive to noise, but also makes it more susceptible to feature edges and discontinuities in the original part. In our experiments, we found that $w = 3.0$ works well even for noisy meshes.

3.2 Biquadratic Surface Fitting

We approximate the mesh near vertex $\mathbf{x}_i$ by least squares fitting a biquadratic polynomial to the vertices in its neighborhood. We perform a linear least squares fit in a local (u,v,w) coordinate system with $\mathbf{x}_i$ at the origin and its vertex normal $\mathbf{n}_i$ as the w axis. The normal of the vertex is defined by

$$\mathbf{n}_i = \sum_{j\in F(i)} q_j \mathbf{n}(F_j) \Big/ \left\| \sum_{j\in F(i)} q_j \mathbf{n}(F_j) \right\|, \tag{3}$$

where $\mathbf{n}(F_j)$ is the normal of incident face F_j, and q_j is the angle formed by the edges of F_j incident to $\mathbf{x}_i$. Although there are various ways of estimating the surface normal, we choose this one because it is fast, local, and achieves good results. The u,v axes of the local coordinate system are found by taking cross products with $\mathbf{n}_i$. We find the positions $\mathbf{u}_j$ of the neighborhood vertices using the rotation and translation

$$\mathbf{Q} = \begin{bmatrix} \mathbf{u}_i^T \\ \mathbf{v}_i^T \\ \mathbf{w}_i^T \end{bmatrix}, \quad \mathbf{T} = -\mathbf{x}_i, \tag{4}$$

in

$$\mathbf{u}_j = \mathbf{Q}\mathbf{x}_j + \mathbf{T}. \tag{5}$$

Then we determine its coefficients of the full biquadratic polynomial

$$h(u,v) = a_{00} + a_{01}u + a_{10}v + a_{20}u^2 + a_{11}uv + a_{02}v^2$$

by least-squares fitting it to the neighborhood. We solve the least squares problem by generating the normal equations and then finding the polynomial coefficients with Cholesky factorization. See a reference such as [10] or [11] for details. Finally, we use $h(u,v)$ to define a patch

$$\mathbf{x} = (u, v, h(u,v))$$

that approximates the vertex neighborhood.

3.3 Curvature Calculation

With a polynomial approximation for the neighborhood in hand, we can calculate robust estimates of the mean and Gaussian curvatures at the vertex. For other methods of calculating curvature on meshes, see [12]. The curvature estimates are then used to compute the principal and absolute curvatures at the vertex, which will be useful for seed vertex selection, as described in Section 4. We begin by calculating the coefficients of the first and second fundamental forms of the surface at the origin of the local coordinate system. They are

$$\begin{aligned} E &= \mathbf{x}_u \cdot \mathbf{x}_u & \qquad L &= \mathbf{n}\cdot \mathbf{x}_{uu} \\ F &= \mathbf{x}_u \cdot \mathbf{x}_v \quad \text{and} & M &= \mathbf{n}\cdot \mathbf{x}_{uv} \\ G &= \mathbf{x}_v \cdot \mathbf{x}_v & N &= \mathbf{n}\cdot \mathbf{x}_{vv}, \end{aligned}$$

where $\mathbf{n}$ is given by

$$\mathbf{n} = \frac{\mathbf{x}_u \times \mathbf{x}_v}{\left\| \mathbf{x}_u \times \mathbf{x}_v \right\|} \tag{6}$$

and all partial derivatives are evaluated at the origin. The Gaussian curvature is then

$$K = \frac{LN - M^2}{EG - F^2}$$

and the mean curvature is

$$H = \frac{GL + EN - 2FM}{2\left(EG - F^2\right)}.$$

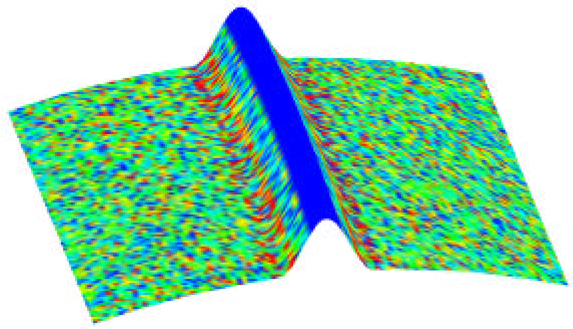


Figure 2. The estimated noise on the mesh. The noise is zero where Eq. (8) holds, shown in blue.

Finally, the principal curvatures are

$$k_1 = H + \sqrt{H^2 - K}$$

and (7)

$$k_2 = H - \sqrt{H^2 - K}.$$

The absolute curvature, which will be used for seed vertex selection, is defined as the square root of the summed squares of the principal curvatures

$$k_{abs} = \sqrt{k_1^2 + k_2^2}.$$

3.4 Feature Edge Detection

If the neighborhood of a vertex includes a sharp edge, then a biquadratic polynomial will not in general be able to model it correctly. It is unwise to trust the polynomial approximation for noise estimation in such a case, since a poor fit will likely over- or underestimate the noise at a vertex. Therefore, we want to use the biquadratic surface for noise estimation only when the surface is relatively flat - in places where the original surface has small, slowly varying curvatures. The quantity $k_{1,i}$ is the largest curvature (alternatively, the smallest radius of curvature) in any direction at vertex $\mathbf{x}_i$. A small radius of curvature indicates a feature edge or discontinuity in the original surface. Through various experiments, we determined that when the radius of curvature is less that 10 times the average edge length, then a feature is probable near a vertex. Therefore, if

$$\frac{1}{\left|k_{1,i}\right|} < 10\, l_{avg,i} \tag{8}$$

for the polynomial approximating the neighborhood of vertex $\mathbf{x}_i$, then the vertex error is not included in the estimate of the total noise for the mesh.

3.5 Global Noise Estimation

We can now estimate the mesh noise. We quantify the noise in both a G^0 and G^1 sense. That is, in addition to measuring the difference between each mesh vertex and its approximating polynomial (the G^0 error estimate), we also measure the difference between each mesh vertex normal and the normal of its approximating polynomial (the G^1 error estimate). By definition, the vertex whose noise we are measuring lies at the origin of its local coordinate system. Therefore, the G^0 noise for vertex $\mathbf{x}_i$ is simply

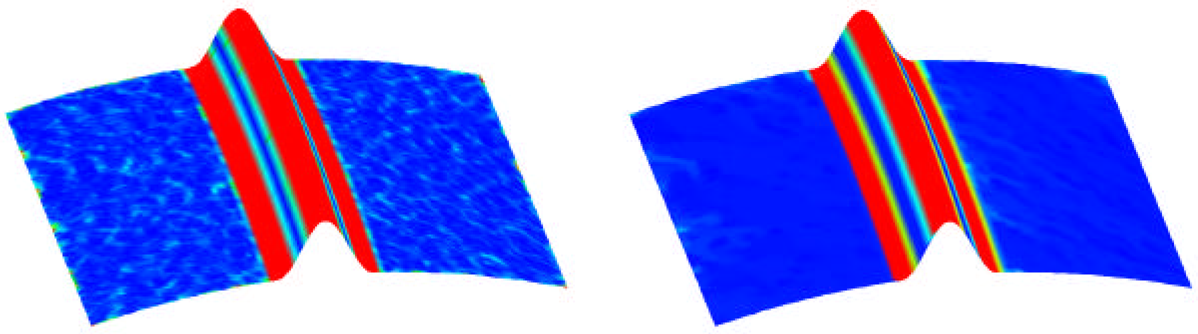


Figure 3. The absolute curvature on the mesh. The unfiltered curvature is shown on the left and the filtered curvature is on the right. Note how the absolute curvature strongly differentiates between flat and curved areas.

$$e_i^0 = |a_{00}| . \qquad (9)$$

The G^1 error measures the difference between the vertex normal and the surface normal of the approximating surface and is given by

$$e_i^1 = \cos^{-1}(\mathbf{n} \cdot \mathbf{n}_i) , \qquad (10)$$

where $\mathbf{n}$ and $\mathbf{n}_i$ are as defined in Eqs. (3) and (6). Both of these error estimates will be used for testing vertex compatibility during region growing. Note that if Eq. (8) holds for a vertex $\mathbf{x}_i$, then we set $e_i^0 = e_i^1 = 0$. The error e_i^0 is shown in Figure 2.

We must remark here that our mesh noise calculation makes some notable assumptions. First, Eq. (9) assumes the error of each vertex occurs only in the vertex normal direction. In reality, the error of the vertex is truly three-dimensional and may likely be related to laser scanner orientation. Furthermore, we do not calculate the shortest distance from each vertex to its approximating polynomial. Our estimate, however, requires virtually no computation and never underestimates the noise of a vertex. Second, Eq. (10) assumes that our calculation of the mesh vertex normal in Eq. (3) is in some sense correct for the vertex, when this clearly may not be the case in the presence of noise. Finally, the polynomial approximation itself is a function of the mesh vertex normal calculation, since the shape of the surface approximating the vertex neighborhood will change with the local coordinate system. We clearly simplify the problem a great deal. Nevertheless, our calculated noise values correspond well with those quoted by laser scanner manufacturers and laser scanner operators and we believe our noise estimation is sound.

To calculate the noise level for the entire mesh, we calculate the sum of squares of the G^0 and G^1 errors for all vertices not lying near feature edges. Dividing the sum by the number of vertices for which the errors were computed and taking the square root, we arrive at an estimate for the root-mean-square position and normal errors,

$$s^j = \left(\frac{1}{M'} \sum_{i=0}^{M} \left(e_i^j\right)^2 \right)^{1/2} , \qquad (11)$$

where $j = 0,1$ and M and M' are the total number of mesh vertices and the number of mesh vertices not on feature edges according to Eq. (8), respectively. During segmentation, the noise estimates will be crucial in deciding if vertices should be added to growing regions.

4. SEED VERTEX SELECTION AND CURVATURE FILTERING

4.1 Seed Vertex Selection

We use a simple scheme for seed vertex selection, sorting the vertices by their filtered absolute curvature and then growing regions from each vertex in order of ascending curvature. We choose absolute curvature because it can detect flat areas more reliably than Gaussian curvature and is more sensitive to surface bending than mean curvature.

4.2 Curvature Filtering

Although the size of the neighborhoods used for polynomial approximation mollifies some of the effects of noise on the estimated curvature, the curvature can still appear speckled. Therefore, we suggest smoothing the curvature with a single pass of a median filter. The implementation of a median filter on mesh curvature is parallel to its implementation on an image. For each vertex, we sort the absolute curvature values of its adjacent vertices and then change its curvature value to the median absolute curvature:

$$\left(k_{abs,i}\right)_{new} = median\left[k_{abs,j}\right] .$$

A single iteration of this approach produces good results. Note that the mesh vertices are not moved, but only the curvature of the vertices is changed. A median filter is appropriate for smoothing the curvature because it is a robust statistic uninfluenced by outlier noise and because it does not create new curvature values. The effects of median filtering on the absolute curvature are shown in Figure 3.

We assume that, at any point during the segmentation process, the best seed vertex is the one with lowest absolute curvature not already belonging to any region. Therefore, to find seed vertices, the algorithm first sorts all the vertices by their filtered absolute curvatures and then attempts to grow regions from all the vertices in the mesh in order of increasing absolute curvature, skipping vertices already assigned to regions.

5. SEED REGION CONSTRUCTION

Once a seed vertex has been selected, the algorithm checks if the vertex neighborhood is suitable for region growing and, if this is the case, uses the neighborhood to find a surface approximation for region growing.

We choose to segment meshes into regions that can be accurately represented by bicubic polynomials because such functions are easy to deal with and prove the feasibility of our approach. Clearly, however, there are surfaces common in engineering, such as cylinders and spheres, which cannot be properly modeled with such functions. Nevertheless, many classes of surfaces could be used within the framework presented here. The requirements are that one can readily fit the surfaces to an arbitrary mesh, calculate surface normals, and calculate distances from the surfaces to points in space.

Because the algorithm uses a full bicubic polynomial for region growing, at least ten vertices are needed for least-squares fitting. In our implementation, we use at least twenty vertices to ensure algorithmic stability. The initial neighborhood is constructed by adding the seed vertex to it, and then adding more vertices based on their topological distance from the seed vertex until the neighborhood contains more than twenty vertices.

Once the initial neighborhood has been constructed, the seed vertex normal is used to determine a local coordinate system and a bicubic polynomial surface is fit to all the vertices in the vertex neighborhood in this coordinate system (see Eqs. (4) and (5)). The full bicubic polynomial is

$$\begin{aligned} h(u,v) &= a_{00} + a_{01}u + a_{10}v \\ &+ a_{20}u^2 + a_{11}uv + a_{02}v^2 \\ &+ a_{30}u^3 + a_{21}u^2v + a_{12}uv^2 + a_{03}v^3. \end{aligned}$$

Because the initial neighborhood is grown rather arbitrarily and may, for example, contain a feature edge, we must test its compatibility with the approximating surface. We therefore test the G^0 and G^1 compatibility of each vertex in the initial neighborhood with the bicubic polynomial. If any vertices in the neighborhood do not satisfy the compatibility requirements, the region is rejected. This means none of the vertices in the seed region are assigned to a region. Then the algorithm moves on to the next seed vertex. Compatibility is verified by first checking if

$$\left|w_j - h(u_i, v_i)\right| < w_0 s^0, \tag{12}$$

where w_j is the height in the local coordinate system of vertex $\mathbf{x}_j$ and h is evaluated at (u_i, v_i). Then, we calculate the G^1 compatibility. Letting

$$\mathbf{x} = (u, v, h(u,v))$$

again, we check the condition

$$\cos^{-1}(\mathbf{n}_i \cdot \mathbf{n}) < w_1 s^1 \tag{13}$$

where $\mathbf{n}_i$ is the vertex normal calculated by Eq. (3) and $\mathbf{n}$ is found with Eq. (6) evaluated at (u_i, v_i). The coefficients w_0 and w_1 are empirically determined and, in general, scanner dependent. If we assume the noise is normally distributed, then the s^i are standard deviations and, for example, 95% of vertices can be expected to lie within $2s^0$ of a good polynomial approximation.

6. REGION GROWING

If the surface approximating the initial neighborhood of the seed vertex passes the compatibility tests, Eqs. (12) and (13), then the neighborhood becomes a seed region and we begin region growing. This is a simple process in which vertices adjacent to the region are added to it if they do not yet belong to a region and if they satisfy the compatibility inequalities (12) and (13). This is continued until all vertices adjacent to the region are either incompatible with its polynomial approximation or are already assigned to other regions.

Once a region finishes growing, we try to enlarge it by fitting a new bicubic polynomial surface to it. Because the region can be quite large, however, the seed vertex normal does not necessarily provide a good basis for a local coordinate system. Therefore, we perform a linear least squares fit of a plane in the existing local coordinate system and then use the plane normal to compute a new local coordinate system. That is, if the plane is given by

$$h(u,v) = a_{00} + a_{01}u + a_{10}v,$$

then its normal is

$$\mathbf{n} = \frac{\left[-a_{01}, -a_{10}, 1\right]^T}{\sqrt{a_{01}^2 + a_{10}^2 + 1}},$$

and we fit the new bicubic polynomial surface to the region in the coordinate system defined by $\mathbf{n}$ and the seed vertex position. Note

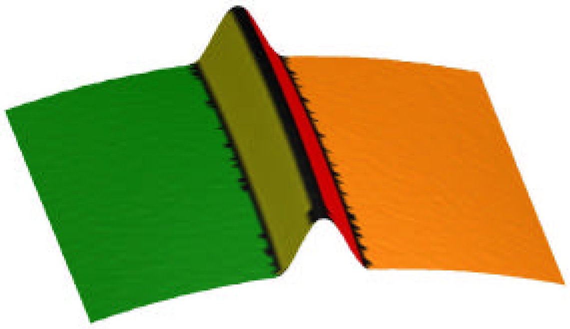


Figure 4. Final segmentation. Black lines represent unsegmented areas.

that $\mathbf{n}$ must be cast into the global coordinate system. The vertices in the region might no longer be compatible with the new surface representation, so the region is cleared and all its vertices are reset to not belong to any region. Then the algorithm begins growing the region from the seed vertex again. The goal of this approach is to obtain the largest possible region with the best possible surface representation.

We repeat the process of region growing and surface refitting until the size of the region stops increasing from one iteration to the next. In our implementation, we stop the algorithm when the size of the current region is less than five vertices larger than the size of the previous region. That is, we stop when

$$\left|R_{new}\right| - \left|R_{old}\right| < 5. \tag{14}$$

For the first iteration, $\left|R_{old}\right|$ is set equal to the size of the initial seed vertex neighborhood. When a region meets this termination criterion, it is saved and a new seed vertex is selected. A segmented mesh is shown in Figure 4.

Once a region finishes growing, we have a group of connected vertices that are all approximated to known tolerances $w_i s^i$ by a bicubic polynomial surface in a local coordinate system. When all seed vertices have been checked, if the compatibility conditions are properly set, then the mesh is partitioned into connected sets of vertices representing meaningfully different surfaces.

Speckle Removal

Once the mesh segmentation is complete, the algorithm makes one pass through the mesh to remove small holes in the regions caused by outlier vertices. Outlier vertices are caused by spike noise in the laser-scanner data and lie far outside the bounds assumed by a normal distribution and would generally not pass the compatibility tests for a region. Therefore, vertices obviously corrupted by spike noise are assigned to the region surrounding them. Specifically, after segmentation is complete, all vertices not assigned to a region, but surrounded by vertices belonging to a single region, are assigned to their surrounding region.

7. RESULTS

In Figures 6 and 7, we demonstrate the various steps of our algorithm on two real data sets. Figure 6 shows an automobile C-pillar and Figure 7 shows part of an engine. Figures 6(a) and 7(a) merely demonstrate the noise level of the meshes by simulating reflection lines on them.

In Figures 6(b) and 7(b), the mesh vertices are colored based on the noise value calculated in Eq. (9). Vertices where Eq. (8) holds and the error is set to zero, are shown in blue. Note that strong feature edges are correctly avoided. The calculated normal errors are

$s^0 = 0.24$ mm and $s^1 = 6.5°$ for the automobile panel and $s^0 = 0.075$ mm and $s^1 = 8.0°$ for the engine.

Figures 6(c) and 7(c) show the median-filtered absolute curvature of the meshes, and Figures 6(d) and 7(d) show the final segmentation. Note the robust curvature estimation in the presence of noise. For segmentation, we used $w_0 = 1.0$ and $w_1 = 0.6$ for the automobile panel and $w_0 = 2.5$ and $w_1 = 2.0$ for the engine. In both cases, the segmentation produces large segments with clear boundaries.

In Figures 6(e) and 7(e) we demonstrate the smoothness of the polynomial surfaces approximating each region by merely projecting each vertex onto the polynomial approximating the region to which it belongs. Discontinuities at region boundaries and unsegmented areas are visible.

8. CONCLUSIONS AND DISCUSSION

Our algorithm generates reasonable segmentations of noisy and smoothed laser-scanner data with little or no user interaction. The algorithm also works in a few seconds, even on large data sets. In addition to partitioning the mesh data, the algorithm also creates surfaces that approximate the data to within known tolerances. The segmentation and the surfaces associated with it have many applications, including feature detection, boundary generation for patch fitting, and surface reconstruction. The segmentation can often be improved if the laser-data is smoothed first using an algorithm that preserves sharp edges and feature lines, such as [13].

Although bicubic polynomial surfaces were used to demonstrate the feasibility of the algorithm, they may not be the best class of surfaces for segmenting general three-dimensional surface data. Clearly, cylindrical or spherical surfaces, which occur frequently in engineering design, cannot be modeled properly with piecewise bicubic surfaces. Furthermore, in this work we parameterize data points by projecting them onto a plane, but for a robust implementation a more general form of parameterization should be used.

Using the approximating surfaces and their corresponding regions to find feature lines and inflection lines is a promising direction of research. The segmentation can also be used for feature detection. The region boundaries and approximating surfaces might also be used to define patch boundaries for part design. Indeed, the region boundaries themselves can be segmented into sets of connected mesh edges represented by piecewise smooth space curves. This is described for images and range data in [2]. Furthermore, one might also use inflection lines calculated from the approximating surfaces to define patch boundaries.

Finally, the segmentation can be used to reconstruct piecewise smooth surfaces from noisy data. The segmentation creates several smooth surfaces that approximate the mesh with high accuracy. The region boundaries can then be examined to determine whether adjacent regions merge smoothly or along sharp edges. Adjacent surfaces with very similar surface normals along their boundaries could be blended and those with differing normals could be connected along their intersection, creating a sharp edge. These surfaces could then be used to remesh the part.

ACKNOWLEDGEMENTS

This work is based in part on work supported under an NSF CAREER Award (No. 9985288). We would like to thank Konica-Minolta and Honda for the data sets used in this work.

REFERENCES

[1] Besl, P. J. and Jain, R. C., 1988, "Segmentation Through Variable-Order Surface Fitting," IEEE Transactions on Pattern Analysis and Machine Intelligence, **10**, pp. 167-192.

[2] Besl, P. J., 1988, *Surfaces in Range Image Understanding*, Springer -Verlag, New York.

[3] Djebali, M., Melkemi, M. and Sapidis, N., 2002, "Range-Image Segmentation and Model Reconstruction Based on a Fit-and-Merge Strategy," *ACM Symposium on Solid Modeling and Applications*, pp.

[4] Sapidis, N. S. and Besl, P. J., 1995, "Direct Construction of Polynomial Surfaces from Dense Range Images through Region Growing," ACM Transactions on Graphics, **14**, pp. 171-200.

[5] Mangan, A. P. and Whitaker, R. T., 1999, "Partitioning 3D Surface Meshes Using Watershed Segmentation," IEEE Transactions on Visualization and Computer Graphics, **5**, pp. 308-321.

[6] Pulla, S., Razdan, A. and Farin, G., 2001, "Improved Curvature Estimation for Watershed Segmentation of 3-Dimensional Meshes," Arizona State University, Tech. Rep.

[7] Page, D. L., 2003, "Part Decomposition of 3D Surfaces," Ph. D., University of Tennessee, Knoxville.

[8] Wu, K. and Levine, D., 1997, "3D Part Segmentation Using Simulated Electrical Charge Distributions," IEEE Transactions on Pattern Analysis and Machine Intelligence, **19**, pp. 1223-1235.

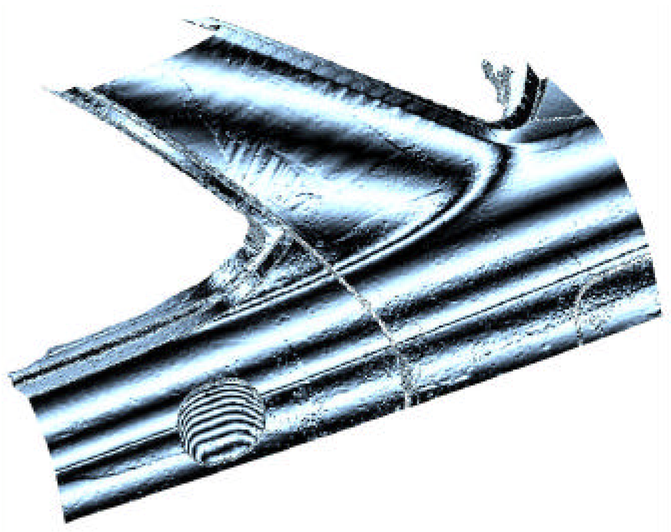
[9] Lesage, D., Leon, J.-C. and Véron, P., 2001, "Discrete Curvature Approximations for the Segmentation of Polyhedral Surfaces," *DETC 2001*, Pittsburgh, PA, pp.

[10] Press, W., Teukolsky, S., Vetterling, W. and Flannery, B., 1997, *Numerical Recipes in C, Second Edition*, Cambridge University Press, New York, NY, USA.

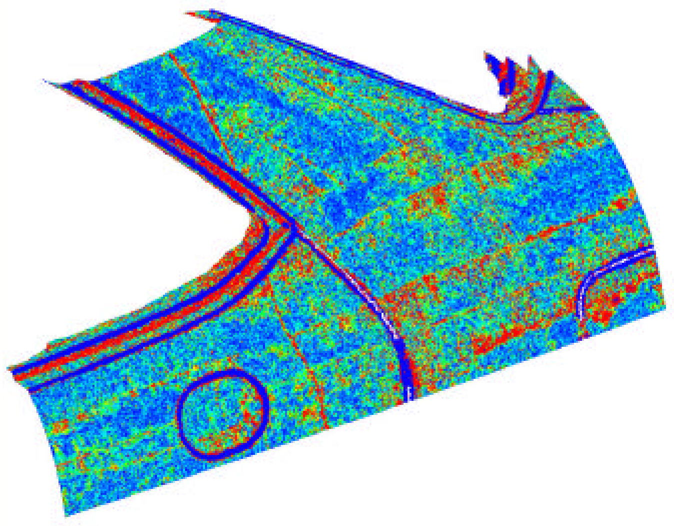
[11] Golub, G. and Loan, C. V., 1996, *Matrix Computations, Third Edition*, Johns Hopkins University Press, Baltimore, MD.

[12] McIvor, A. and Valkenburg, R., 1997, "A Comparison of Local Surface Geometry Estimation Methods," Machine Vision and Applications, **10**, pp. 17-26.

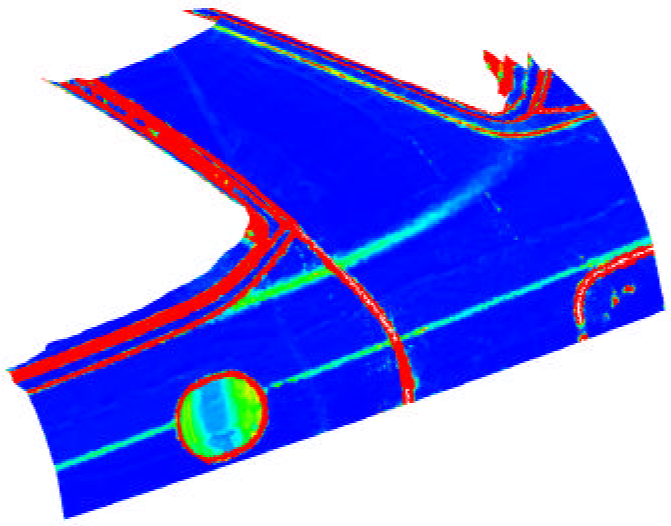
[13] Vieira, M., Shimada, K. and Furuhata, T., 2004, "Smoothing of Noisy Laser Scanner Generated Meshes Using Polynomial Fitting and Neighborhood Erosion," Journal of Mechanical Design (to appear).



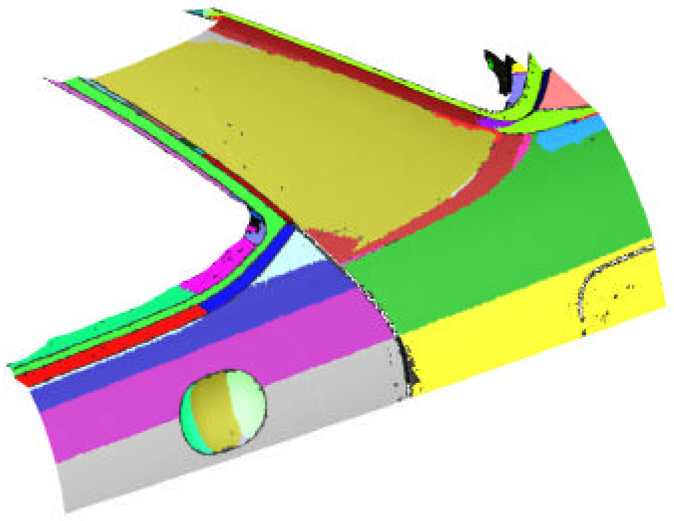
(a) Reflection lines on noisy mesh.



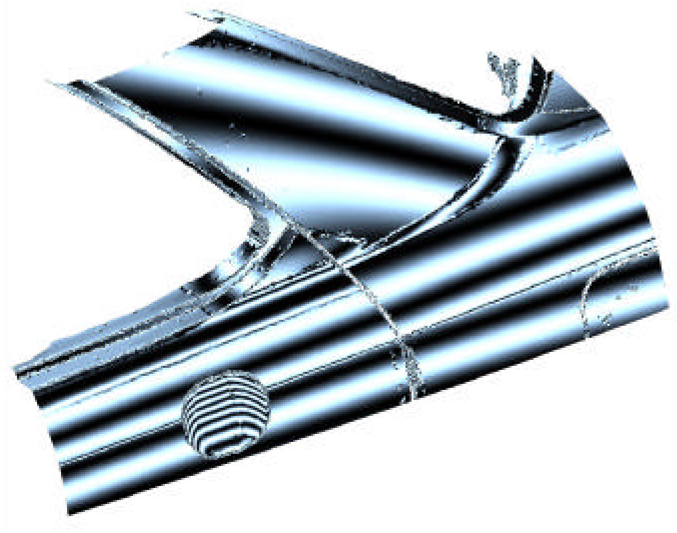
(b) Estimated noise.



(c) Filtered absolute curvature.

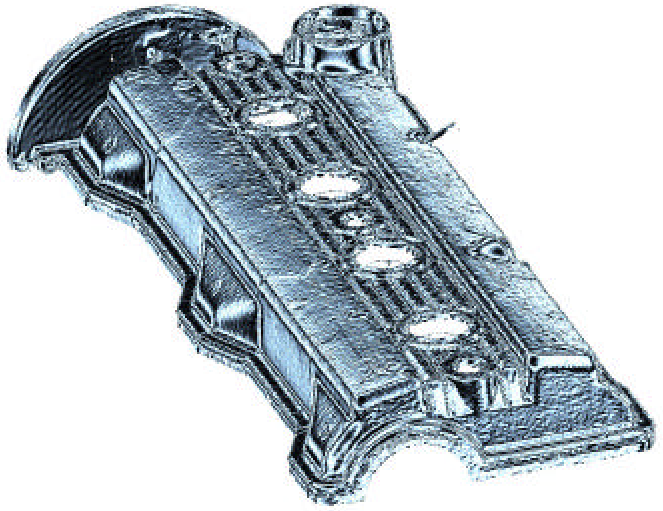


(d) Segmentation

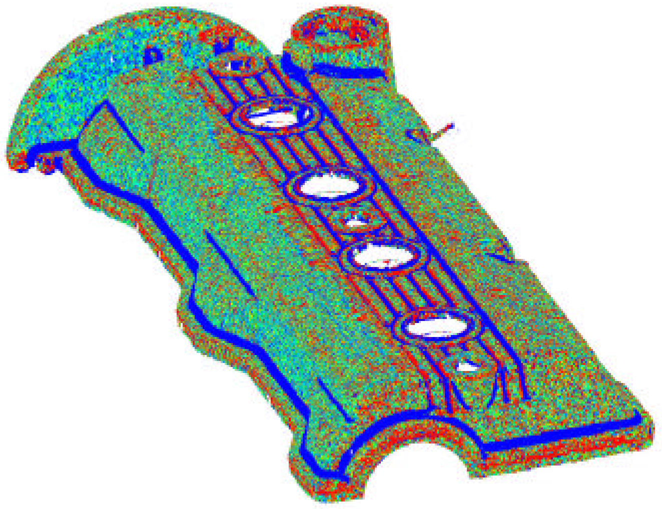


(f) Reflection lines on approximating surfaces

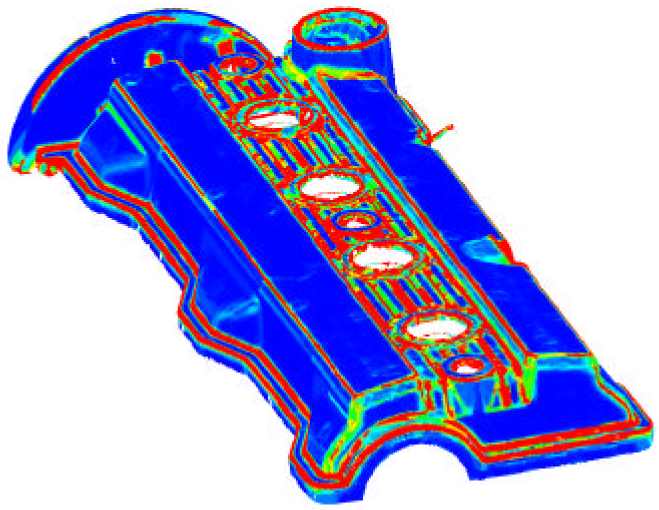
Figure 6. The different steps of our algorithm on a noisy laser scan of an automobile C-pillar. 99,970 vertices.



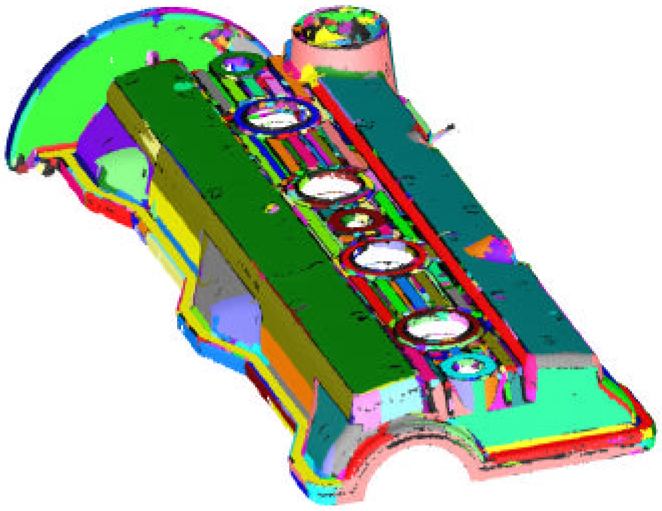
(a) Reflection lines on noisy mesh.



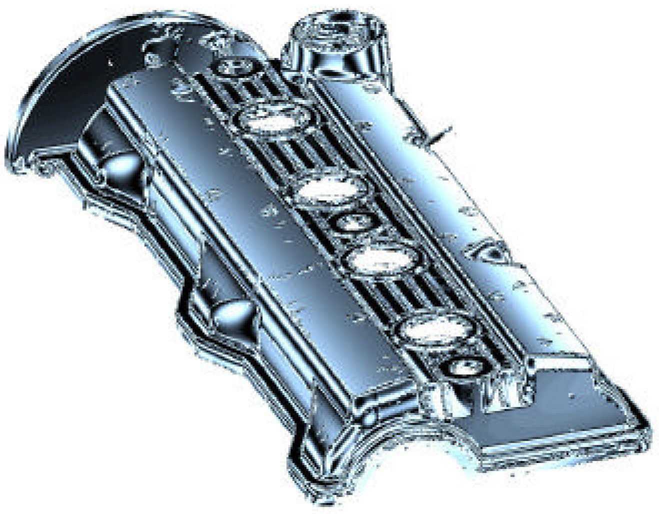
(b) Estimated noise.



(c) Filtered absolute curvature.



(d) Segmentation.



(f) Reflection lines on approximating surfaces.

Figure 7. The different steps of our algorithm on a noisy laser scan of an engine part. 278,667 vertices.