Schuhmann, Norbert
Fraunhofer-Institut fuer
Produktionstechnik und Automatisierung
Nobelstr. 12
D-70569 Stuttgart, Germany
Phone: +49 711 970 1851
Fax: +49 711 970 1004
e-mail: nos@ipa.fhg.de

New Methods for Feature-Line Extraction from Point Clouds
Méthodes d’extraction de lignes à partir d’un nuage de points

New Methods for Feature-Line Extraction from Point Clouds
Méthodes d'extraction de lignes à partir d'un nuage de points

Abstract
Feature lines are substantial means for the segmentation of measured point clouds, especially if a surface model has to be generated from the data. In this article, methods for the automated extraction of feature lines from measured or scanned point clouds will be described. These methods can be applied to points from any sensor since they do not require sorted or edited data sets. The mathematical basis for the feature line extraction, experimental results and the limits of the methods will be presented.

Keywords: Information technology, CAD, reverse engineering, feature lines.

Introduction
To date a 3D CAD model from an existing physical part will be derived mostly by taking data from a 3D scanner, e.g. a laser triangulation scanner or a coordinate measuring machine (CMM), then measuring and analyzing the dense discrete point data to derive the 3D CAD model. Typically the measured data are large point clouds up to millions of points in the case of laser scanners with varying densities.

One class of reverse engineering methods is based on the segmentation of the point cloud into polygonal regions to fit an individual surface to each region. To avoid undesired smooth surfaces in regions with edges, edges of the scanned part (together with additional boundaries) should form the boundaries of each polygonal region.

Since in current CAD systems this segmentation has to be done manually, the subject of the paper are methods for the automated extraction of feature lines - fillet end lines and theoretical edges. To be independent from special 3D scanners, no order information of the point cloud will be considered - the point cloud will be regarded as totally unsorted scattered data.

Classification of Feature Lines
Figure 1 shows three different types of feature lines for reverse engineering applications:

- Step Edges: They appear due to scanner insufficiencies. Optical scanners only can measure, what can be seen from outside the part. In the case of holes and channels, normally only a few points at the beginning of these geometrical features can be recorded and step edges appear.
- Fold Edges: They are characterized by discontinuities in the first derivative across the edge and will be generated with separate surface patches in CAD modeling.
- Smooth Edges: In CAD modeling of parts fold edges do not appear as often as commonly expected. Especially in the case of design surfaces, edges will be often modeled as smooth edges and the surface and edge modeling will be done with fillet surfaces with a small fillet radius. The smooth edge itself will be represented by four entities then: Two fillet end lines, the edge line on the part and additionally a theoretical edge line "above" the part (see Figure 2).

Automatic and Semi-Automatic Methods for Feature Line Extraction
Current work at the Fraunhofer-Institute Manufacturing Engineering and Automation (FhG-IPA) concentrates on two different types of algorithms for feature line extraction:

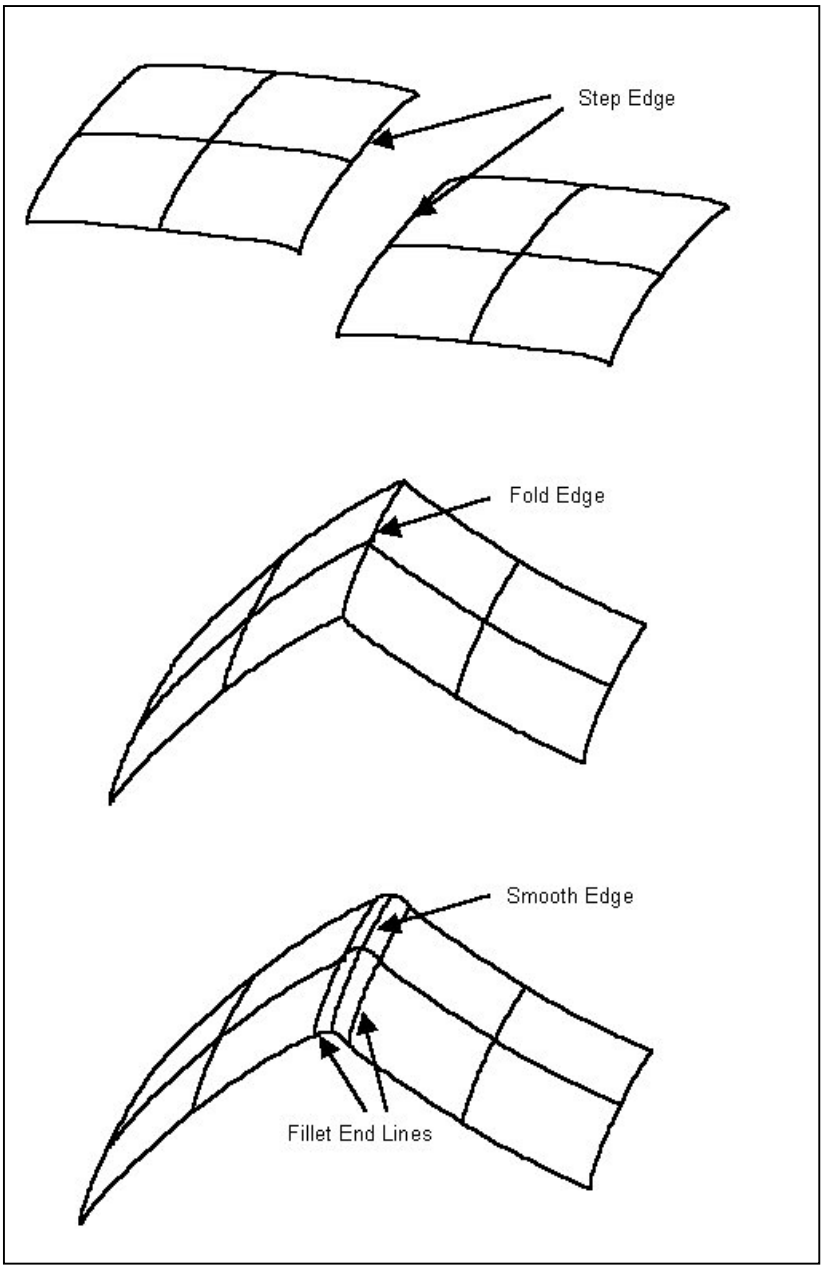

Figure 1: Classification of feature lines

1. Automatic extraction of sharp edges based on robust curvature analysis of the whole point cloud: This algorithm is based on the results of Knorpp (s. [7]) and was further optimized in the meantime.
2. Interactive extraction of sharp edges and/or smooth edges (semi-automatic method). In the latter case fillet end lines, the edge on the part and the theoretical edge line can be computed (see figure 2). The user will be requested to mark a region on the point cloud, where the algorithm will extract the feature lines. Here, the benefit of user interactions will be, that with a pre-segmentation of the point cloud the feature extraction can be done in (or near) real-time. Besides, the (raw) location of feature lines can mostly be seen by the user in the CAD visualization of the point cloud. Optimized user interactions through rough marking of the edge to be detected, allow accurate detection of the fillet end lines, the edge on the part and the theoretical edge, even if the accuracy of the cloud would be rather poor (noisy clouds).

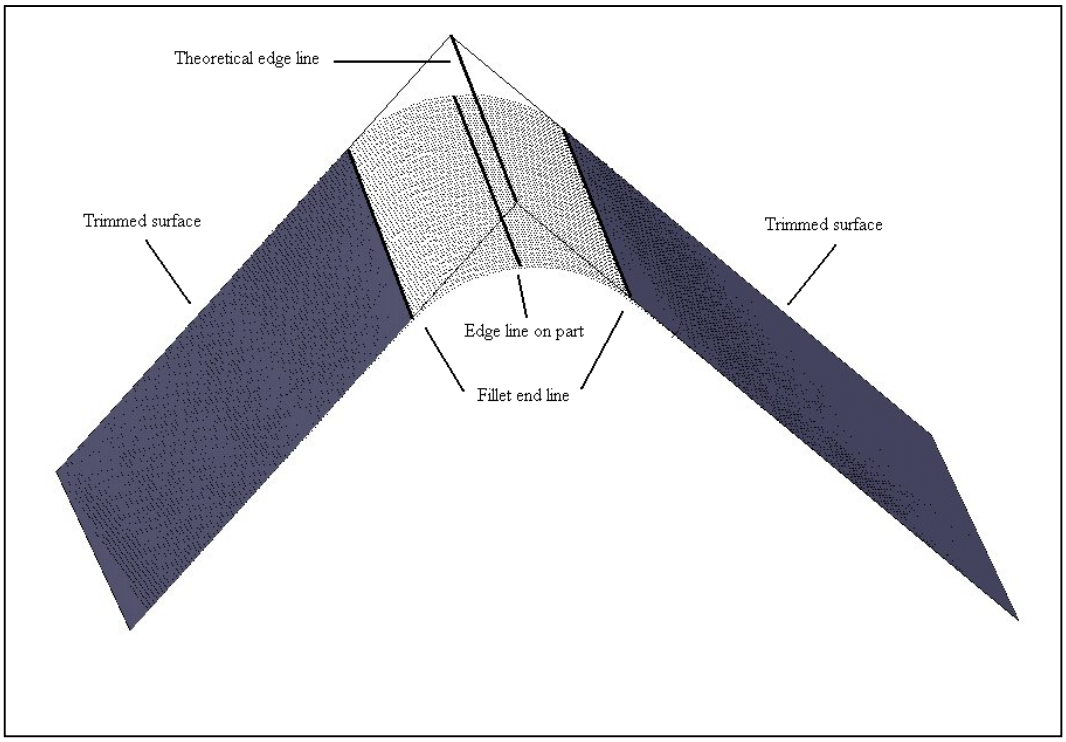

Figure 2: Feature lines and adjacent surface patches

Since the method for automated extraction of sharp edges were already published, the current paper will concentrate on the second method.

Conventional Construction of Fillet Surfaces in Reverse Engineering

Figure 6 shows a reverse engineered car rim based on 3D scanner data. In several regions fillet surfaces were constructed. To date, in the most available reverse engineering CAD-systems there exist powerful tools, to generate even different kinds of fillet surfaces (e.g. circular, parabolic fillets) by a few user interactions, once the adjacent surface patches had been constructed. But it turned out, that accurate generation of these surrounding surfaces is hard manually work, which can be considered as very expensive - expensive in time and consequently in costs. The problem is even bigger, if the fillet surface should be near to the measured points, means if the distances from the measured points in the fillet region to the fillet surface and the surrounding patches is intended to be rather small. In that case the CAD engineer normally does the following 10 steps:

1. Manual selection of a path curve, which follow the path of the desired fillet to be extracted.
2. Multiple cross-sectioning of the measured points in the fillet region perpendicular to the selected path curve.
3. Analyzing each of the cross section point sets for changes in curvature analysis by eye.
4. In each cross section point set: Manual determination of two fillet end points: a start and an end point.
5. Adding all end points to 2 point sets (polylines) and computing two best fit freeform curve, which will form the boundaries of the fillet and the adjacent surface patches, which will be called fillet end curves here.
6. The fillet end curves will be used for manually segmentation of the cloud near the fillet region into 3 parts: a point set to fit the fillet surface and two point sets to fit the adjacent surface patches.

7. Best fit of the adjacent surface patches after additional boundaries were fixed (e.g. by pre-constructed outer surface patches).
8. Tangent or curvature continuously extension of the two adjacent surface patches over the fillet end curves for intersectioning.
9. Intersectioning of the extended adjacent surface patches and possibly trimming of the two surface patches with the intersection curve, since some fillet generation tools require surface patches, which end in a common curve. Of course, this depends on the used CAD system.
10. Construction of the fillet with special attention to meet the fillet end lines and the segmented point set near the fillet.

The following main problems arise in the steps of the presented workflow:

- As already mentioned above, lots of steps need to be performed by the user, which was identified as very expensive. Furthermore, in almost each of the steps, there exists a more or less high probability for errors or inaccuracies, which results in a substantial high probability for an undesired result regarding the whole process chain.
- Curvature analysis of the cross section points by eye cannot be desirable, since normally lots of attempts will be needed to obtain accurate and un-noisy fillet end point sets.
- It turned out, that the most critical issue is the smooth extension (tangent or curvature continuous) of the adjacent surfaces, since smooth extensions of freeform surfaces often result in oscillating surfaces. The succeeding intersection of these extended patches does not lead to a suitable (smooth and accurate) intersection curve, which is needed for the computation of the fillet. Normally only ruled surfaces like full or parts of planes, spheres, cylinders and cones can be regarded as suitable surfaces for smooth and non-oscillating extensions.

Construction of Fillet Surfaces based on Feature Lines

Regarding the presented conventional workflow, the need for an automation of the described steps is evident. The method described below is intended to be as automated as possible with special attention for producing results of high quality in means of accuracy to the measured point cloud and smoothness of the resulting feature lines.

The method computes the following CAD elements:

- Two fillet end lines in NURBS representation for each region, where a fillet should be reverse engineered.
- A theoretical edge line (NURBS representation), which specifies the position of the intersection curve of the extended surfaces, adjacent to the fillet to construct. Of course the input for the method consists of the measured point cloud only; the adjacent surfaces to the fillet need not to be generated before.
- An edge line on the part (NURBS representation). This line normally would not be needed for the construction of the fillet. The edge line on the part can be used, if no fillet, but a sharp edge should be modeled, especially in the case of fillet with very small radii.

Figure 2 shows each of the feature lines to be extracted. To avoid any confusion at this early stage, a very simple part, consisting of plane adjacent surfaces and a cylindrical fillet is demonstrated, here.

After the fillet end lines and the theoretical edge line were computed, two or more adjacent surfaces can be constructed using the theoretical edge line as a common boundary. Analogously the computation of the fillet can be done using the two fillet end lines as boundaries with existing fillet generation tools. It's also possible, to use existing functions to

compute normal blending surfaces instead of a special fillet generator then, since the boundaries of the blending surfaces are determined by the fillet end lines. Another possibility will be to use constrained surface fitting algorithms; then the segmented points will be fitted and the fillet end lines can be considered as the constraints. Algorithms for each of the possibilities are more or less standard methods in reverse engineering and not subject of this paper; they can be found elsewhere (s. [10], [13], [16]).

Optimized Workflow for Automated Feature Line Extraction

Not too surprisingly, a workflow for an automated feature line extraction can be set up similar to the conventional manual workflow described above. The main problem is, that no user interactions to check and optimize the quality of each intermediate result were aimed during the main computation. Consequently very stable algorithms during the main computation are required. The workflow consists of the following consecutive steps:

1. Rough selection of a path curve in polyline mode (with user interactions): The pass curve need not follow the fillet to be extracted very accurately – it is only needed as a starting value for the computation and the result will not depend on the exact location of the curve. The path curve is expected to be located "somewhere" in the region of the fillet.
2. Rough pre-segmentation of the point cloud based on distances to the path curve (with user interactions): again only a very rough pre-segmentation is intended.
3. Input of number of sections perpendicular to the path curve (with user interactions).
4. Computation of cross section points (fully automated).
5. Fitting of spline curves, one for each of the cross section points (fully automated).
6. Curvature analysis of each of the fit curves to get the two fillet end points in each of the cross section points (fully automated).
7. Subdivision of the fit curve in the computed fillet end points and tangential extension of the subdivided fit curves (fully automated).
8. Intersectioning of the tangential extensions to compute the theoretical edge point (fully automated).
9. Collection and ordering of the theoretical edge point, the fillet end points and the projected theoretical edge points (projection on the input point cloud) into four polylines: theoretical edge polyline, two fillet end polylines and edge polylines on the part (fully automated).
10. Spline fitting of the four polylines to get the feature lines in NURBS representation (fully automated).

Steps 1-3 need not to be explained in detail, since they are standard tools in modern CAD implementations and were realized just by applying an optimized data-structure to get the intermediate results in or near real-time (see for example [10]). The most problematical step is the multiple spline fitting of the cross section points and the curvature analysis of the resulting spline curves, which will be explained in the following chapter.

Introduction of a Criteria for the Calculation of Fillet End Points

The image below shows the run of the curvature values of a curve, simply obtained by intersectioning of the constructed surfaces for the following base example: The surfaces to be reverse engineered consist of two orthogonal planes, trimmed and connected with a cylindrical fillet with radius $R=1$. Consequently the intersection curve (intersection perpendicular to the axis of the cylinder) consists of two lines connected with a quarter circle. The curvature values of the lines are zero and of the circle constantly equal 1. The run of the curvature values is characterized by a discontinuity exactly in the fillet end points.

This leads to the following search criterion determining the location of the fillet end points:

As fillet end points in a perpendicular cross section shall be considered the location of discontinuities in the curvature values with a substantial sudden increase in the curvature values of the intersection curve.

Of course, the intersection curve is not available at this stage, since it is the goal to compute the surfaces and they cannot be analyzed, before they were computed. But the intersection curve will be approximated by a fitting curve of the points in perpendicular cross sections. In the case of spline curves with single inner knots and polynomial degree equal or more than 3, the spline curves will be at least curvature continuos (C^2 or G^2 continuos) in the knots and C^∞-continuos elsewhere. Even if discontinuities can be modeled by using multiple knots in spline theory, this cannot performed here, since the location of the fillet end point is not known yet. In the case of C^2 or G^2 continuos fitting curves, the criterion determining the fillet end points will be generalized by skipping the discontinuity condition and the search for locations with substantial sudden increase in the curvature values of the fitting curve.

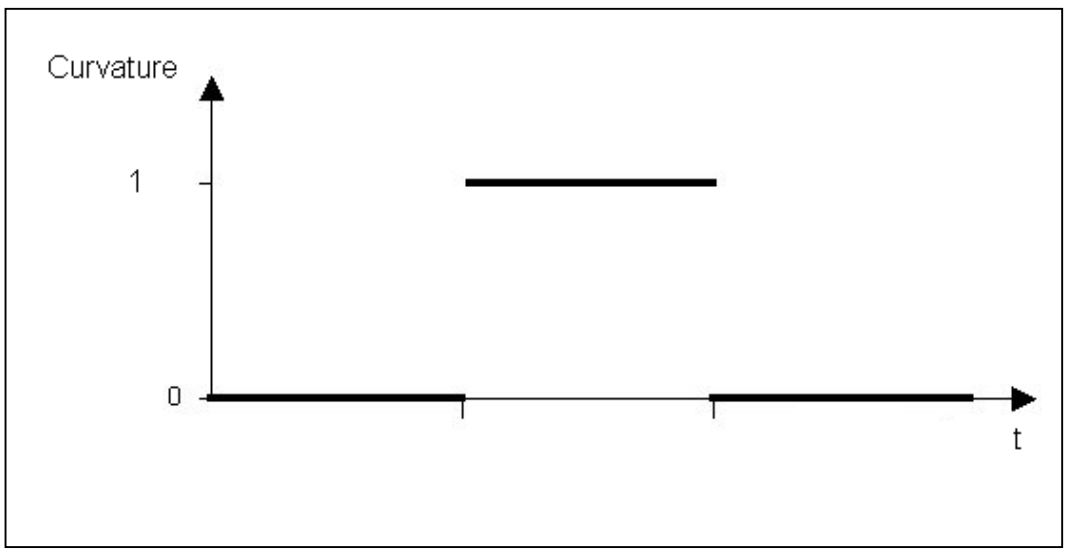

Figure 3: Curvature values for vase example to obtain the criterion for the location of fillet end points

Algorithms for Automated Multiple Spline Fitting for Curvature Analysis of Cross Section Points

For spline fitting, lots of algorithms were proposed by different authors in mathematical CAD bibliography, and at first sight at least some of them seem to be appropriate for the intended curvature analysis of the points in each of the cross section. But especially for automated multiple spline fitting, a well-known problem arises: On the one hand, the resulting spline curve should be accurate respectively to the cross section points, means the distances of the points to the fit curve need to be checked and should be rather small. On the other hand, if the resulting spline curve would follow the cross section points too closely, oscillations appear in the spline curve, if the scan data would be noisy (and it always is). The amount of oscillations of the spline curve normally will increase, if the distance to the points should decrease, but oscillating spline curves can be used for curvature analysis only with great difficulties. There is no real solution out of this classical reverse engineering paradox, but it can be (approximately) solved as good as possible.

The total amount of oscillations S of an arbitrary curve $c(t)$ can be measured by the infinitesimal sum of the squares of the second order derivative $c''(t)$ of the curve:

$$S = \int_0^1 \left| c''(t) \right|^2 dt$$

The total deviation D of $c(t)$ to the $N+1$ cross section points $P_0 \ldots P_N$ can be measured by the sum of the square distances $c(t)$ to the point set $\{P_i\}$:

$$D = \sum_{i=0}^{N-1} \left| c(t_i) - P_i \right|^2,$$

where $t_0 \ldots t_N$ specify pre-computed parameter values, $t_i \; e \; [0,1]$, determined e.g. by chordal parameterization (see [10]). The compromise between accurate curve fitting and a smooth, non-oscillating curve can be formulated mathematically by a convex combination of S and D by introducing the smoothing control parameter σ:

$$E_\sigma(c) := (1-\sigma)D + \sigma S = (1-\sigma)\sum_{i=0}^{N-1} \left| c(t_i) - P_i \right|^2 + \sigma \int_0^1 \left| c''(t) \right|^2 dt,$$

where the smoothing control parameter should be chosen $\sigma \; e \; (0,1)$, where increasing $\sigma \to 1$ will result in an increased smoothing of the spline curve c(t) and decreasing $\sigma \to 0$ will result in a more accurate fit curve.

This method, originally proposed by C. Reinsch (s. [11]) and I.J. Schoenberg (s. [15]), is not very new, but seldom applied in current CAD systems for reverse engineering. The reason is, that the number of control points increases with the number of the points to be fitted and most CAD systems have memory or runtime problems with a substantial high number of control points for a spline curve, but the method can be used for efficient curvature analysis of the cross section point sets, here. The method has the following two interesting properties:

1. Among all at least twice differentiable curves $g(t)$, where $g(t)$ could be an arbitrary curve - not necessarily in spline representation – the unique minimizer of $E_\sigma(g)$ for arbitrary $\sigma \; e \; (0,1)$ is a cubic B-spline with interior knots $t_1 \ldots t_{N-1}$, called smoothing spline curve c_σ.
2. For $\sigma \to 0$ the resulting sequence of cubic spline curve converges to an interpolation curve, means the total deviation D vanishes: $D \to 0$.

The proof is simple and based on the well-known oscillating-minimization property for spline interpolation and a generalization to it (s. [2]). The first property can be interpreted as a really nice (at least) approximating solution for the paradox mentioned above: Even if it is not possible to solve the paradox, it is possible to find the best compromise to minimize the functional E_σ.

The construction of the smoothing spline curve c_σ can be formulated mathematically straight forward as a quadratic form, which will be skipped here and can be found in [2]. The computation results in a linear system of equations for the control points of the spline curve. But it should be emphasized here, that because of the local support of the B-spline basis functions, the matrix of the linear system will be sparse (with three subdiagonals and three superdiagonals) and symmetric and positive definite (the solution is unique) and therefore can be solved with numerical methods very efficiently, for example with the conjugate gradient method of Heestens and Stievens. This pointed out to be an important issue, since the size of the matrix of the linear system increases with the number of points to be fitted and therefore could be very large.

The second property enables the implementation of a best fit to tolerance algorithm for the smoothing spline: Since the sequence of spline curves converges to an interpolating spline curve or more exactly, since the total deviation D decreases monotonously for $\sigma \rightarrow 0$ with the limit $D=0$ in the case of $\sigma = 0$, an iterative algorithm was set up to compute a smooth fit to tolerance curve (tolerance τ):

- Start with $\sigma = 0.5$, compute the fitting spline curve and the total deviation D to the points P_i
- While not $0.9\,\tau < D < 1.1\,\tau$
  { Decrease or increase slightly the smoothing control parameter σ, depended of the fact, whether D will be higher or lower than τ. }

This algorithm will converge to a solution with a total deviation, not only better than the specified tolerance, but even with a total deviation within the "neighborhood" (+/- 10%) of the tolerance. This is extremely important for the curvature analysis, since all spline curves to be fitted for the different sets of cross section points will be forced to have a similar fitting quality in the sense of the total deviation to the points to be fitted and therefore also in the sense of smoothness. In the implementation (see below) the tolerance was set individually to each cross section, depending on the maximum curvature value of the points, which can be computed before with a rough spline fit calculation.

Another subtask to handle should be mentioned here: The described fitting method can be considered as one representative within the class of parametrical curve fitting algorithms. Common to all algorithms in that class are the subtasks:

- Sorting and parameterization of the input point set
- Establish and solve the system of equations (called normal equations) for the unknown control points coordinate-wise, which means e.g. for 3D: three systems of equations with the same matrix must be solved.
- In the case of best fit to tolerance curves: Check the distances to the resulting curve and iterate, until the tolerance will be reached.

The sorting subtask is of great importance for getting the desired results, because the resulting fit curve always "follows" the points as they were sorted in the input point set. Here, sorting can be done by a next neighbor search. Even if sorting algorithms must be considered as runtime critical for arbitrary point distributions, in practice and in the special case of point sets derived by cross sectioning of 3D scanner data, sorting can run very efficiently, once a suitable data-structure for next neighbor search was established (s. [7], [17]). Applying the pre-processing steps space sampling and filtering of the input cloud, the runtime and the quality of the resulting curve for parametrical fitting can be further improved.

The advantage of the chosen smoothing spline curve for curvature analysis can be proven mathematically, but can also be seen by eye. Figure 4 compares the smoothing spline curve with a conventional best fit to tolerance spline curve, where the same fitting tolerance was used: The less amount of oscillations for the smoothing spline is evident.

Curvature Analysis of Point Clouds Using Fit Curves

The curvature analysis of the point clouds can be done straight forward, once the smoothing spline was computed just by analyzing the resulting smoothing spline.

The curvature $\kappa(t)$ of an arbitrary curve is defined through its first and second order derivatives:

$$\kappa(t) = \frac{\left| c'(t) \times c''(t) \right|}{\left| c'(t) \right|^3}$$

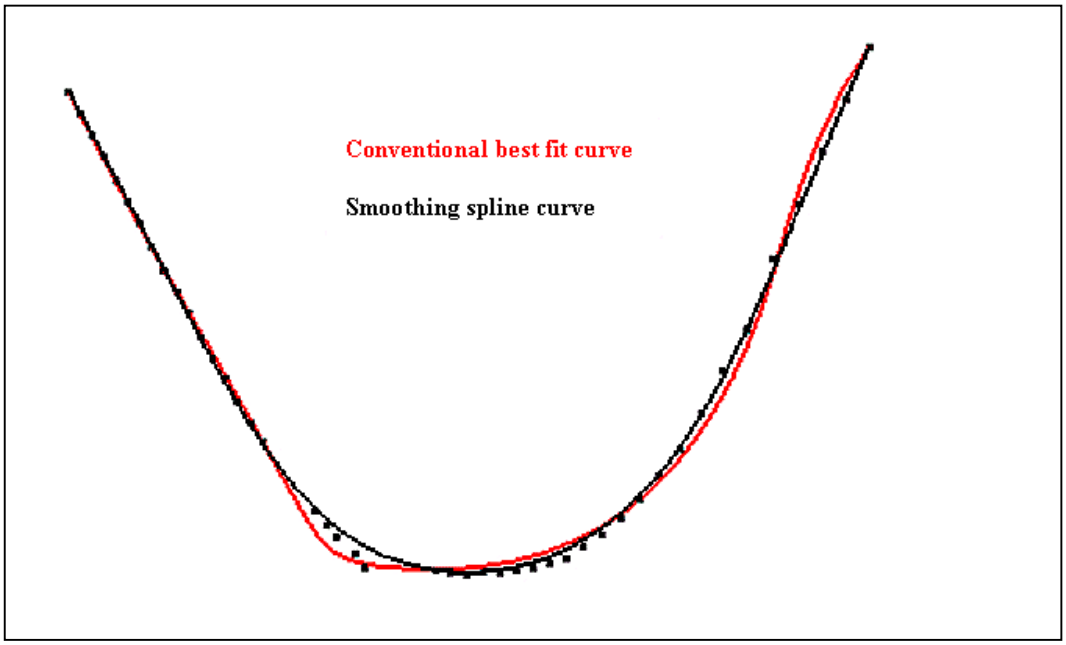

Figure 4: Comparison of a conventional best fit curve and smoothing spline curve

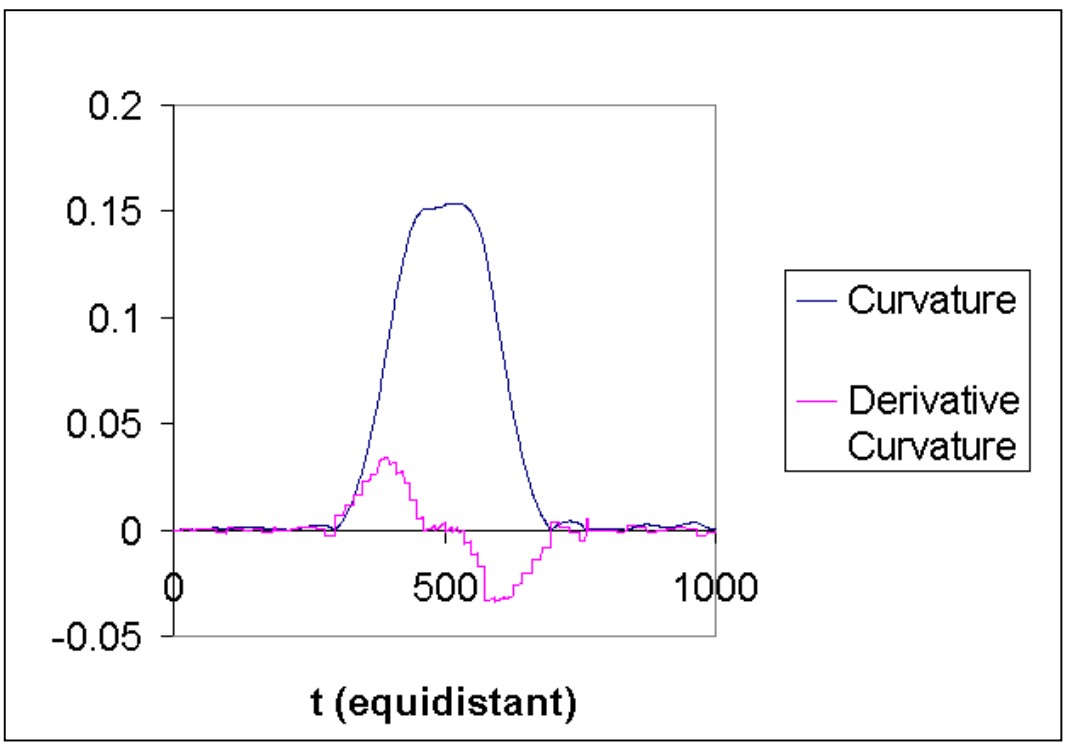

Figure 5: Curvature and derivative curvature values of the smoothing spline curve

As mentioned above, the fillet end points are determined by jumps or at least a strong increase of the scalar curvature function $\kappa(t)$. This implies, that two maxima of the derivative of the curvature function $d/ds\ \kappa(t(s))$ should be computed. The two maxima will be located left and

right from the user selected starting point, which is the center of the cross sectioning plane. To be independent of a special parameterization of the curve, arc length parameterization and the derivatives respectively to the arc length s of the curve should be preferred. The computation of $d/ds\ \kappa(t(s))$ can be done by (multiple) applying the chain rule. Since these arithmetic operations are rather boring and of lower interest, it will be skipped here and the interested reader should be invoked to use modern computer algebraic software like MAPLE or MATHEMATICA to compute the formula.

Analyzing the derivative of the curvature function $d/ds\ \kappa(t(s))$ to find the two maxima, which determine the location of the fillet end points, can be done simply by evaluating the derivative of the curvature function in a uniform dense distributed parameter vector. The maxima will be found by linear search in the resulting evaluation vector, starting somewhere in the middle of the curve, defined e.g. by the parameter value of the center of the cross sectioning plane. Runtime and accuracy can be optimized by an adaptive evaluation strategy: The evaluation and the search will start with a sparse parameter vector, this vector will be condensed subsequently only in the area near the roughly computed fillet end points.

Since the derivative of the curvature function $d/ds\ \kappa(t(s))$ consists of a combination of up to third order derivatives of the curve $c'''(t)$, and since the fact, that the cubic smoothing spline function is only curvature continuos in the knots, the derivative of the curvature function turns out to be discontinuous in the knots here. This can be seen in the image below, but it does not affect the computation of the maxima, which will determine the fillet end points.

Computation of the Theoretical Edge Line

Once the fillet end points were computed, the computation of the theoretical edge point will be done, simply by intersection of the tangents of the curve in the extracted fillet end points. 2D intersection of the tangent can be used, because the fitting curve and consequently the tangents are expected to be located within the cross sectioning plane.

An Example

The images below shows the extracted fillet end lines and the theoretical edge line for one fillet to be extracted. The input data was obtained by 3D scanning of an automobile rim (part of a rim) with a Hymarc 3D laser scanner (triangulation scanner). The pre-segmented cloud consists of 25.000 points, the computation time for the automated cross sectioning, spline fitting, curvature analysis and feature line extraction was only a few seconds on a Intel Pentium II processor with 450 MHz with the operating system MS-Windows. To obtain the feature lines in NURBS representation, the presented smoothing spline algorithm could be used, but here standard parametrical fitting algorithms could be preferred, to get spline curves with less control points.

Benefit of Automated Feature Line Extraction

The example can still be considered as very simple compared to complex reverse engineering jobs. To measure the whole operating time, a test showed, that an averaged experienced user needed about 15 hours to reverse engineer the rim data-set to get the CAD-model consisting of 17 surface patches (trimmed and untrimmed surfaces) by using Imageware's software Surfacer without using the new automated feature line extraction. Applying the new method, the operating time should be at least halved, because it turned out, that the segmentation of the cloud based on feature lines and computing a wireframe model first was the most time-consuming job for the whole project.

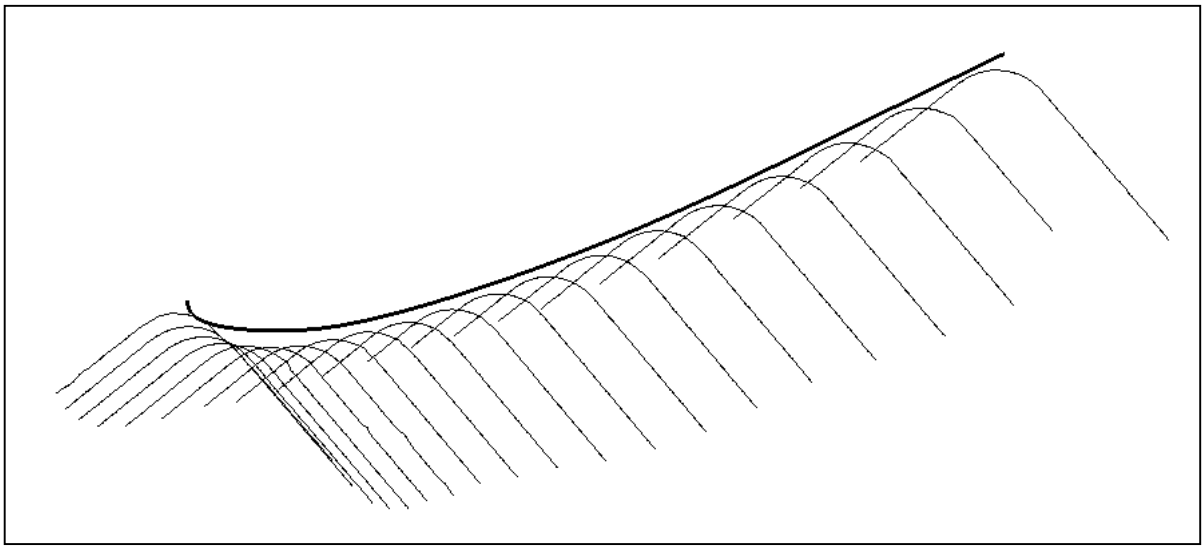

Figure 5: Automated detection of fillet end lines and theoretical edge lines

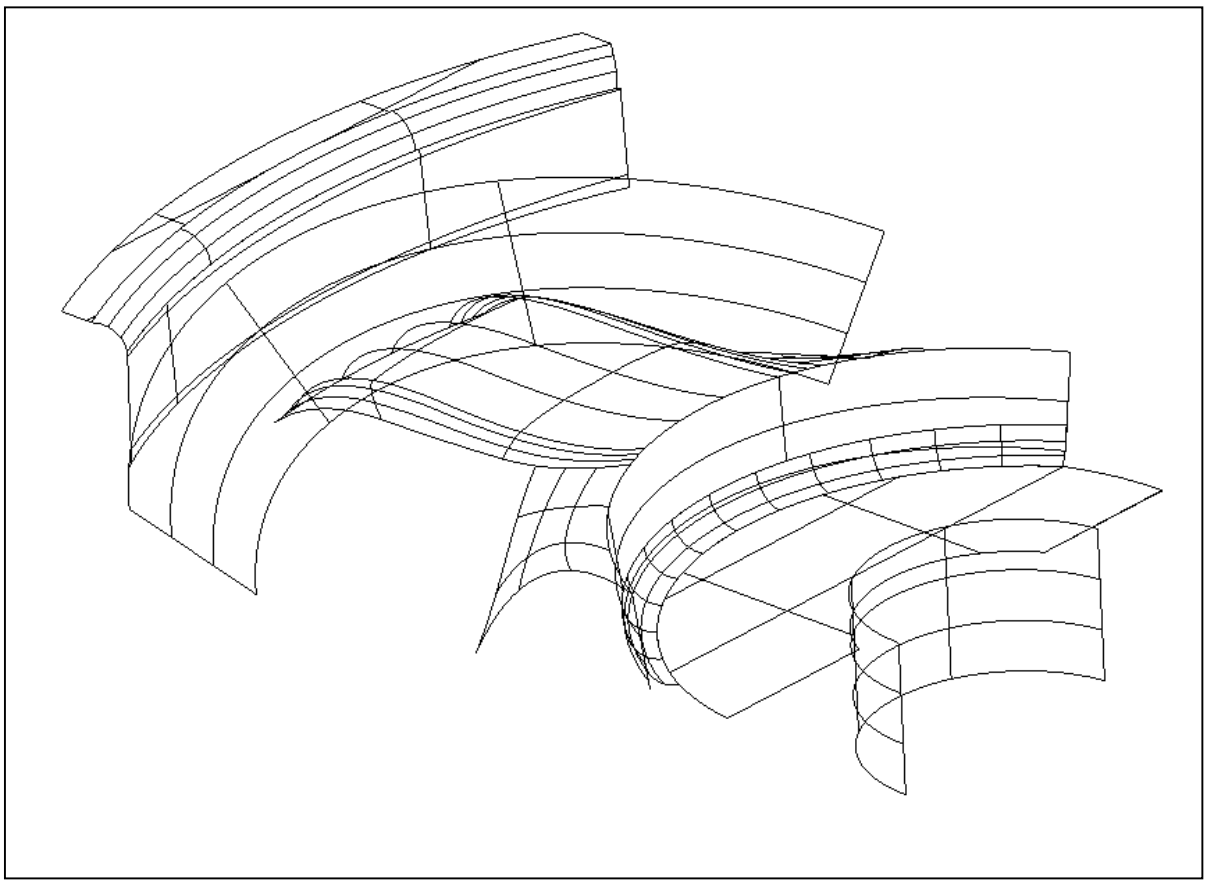

Figure 6: Reverse engineering of an automobile rim (part)

Feature Lines for Class A Surface Modeling

To date, even if there is no clear and common mathematical definition of class A surface modeling (sometimes also called class-1 surfaces), the topic is well established and there is at least a common understanding: Class A surfaces are surfaces of high accuracy and smoothness and are used for design purposes, e.g. for the modeling of the interior and exterior in automobile CAD design. Mostly curvature continuos (C^2 or G^2 continuity) transitions are

required for smooth transitions between adjacent surface patches, but only position-continuos transitions (discontinuities in first order derivatives across the patch borders) for the modeling of fold edges. To do so in reverse engineering, pre-segmentation of scanned point clouds is an essential prerequisite and here especially the detection of fillet locations for establishing the initial wireframe model. Only with an accurate detection of feature lines from the scanned point cloud, surface models of high accuracy and smoothness can be engineered.

Conclusion

Automatic feature line detection is an important tool for reverse engineering applications. With the described algorithms the workflow for reverse engineering of fillet surfaces in 3D scanner data will be automated. The method can handle even dense huge point clouds in real time. It can be expected to halve the operating time for reverse engineering jobs in the case of multiple fillets to be constructed.

References

[1] Bhandarkar, S. M., Siebert, A.: Integrating edge and surface information for range image segmentation. In: Pattern Recognition 25 (1992) No. 9, pp. 947-961.

[2] Boor de, C.: A practical guide to splines. New York u.a: Springer 1978.

[3] Geise, G., Schippke, S.: Ausgleichsgerade, -kreis, -ebene, -kugel im Raum. In: Mathematische Nachrichten 62 (1974) S. 65-75.

[4] Hoffmann, R., Jain, A.K.: Segmentation and classification of range images. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 9 (1987) No. 5, pp. 608-620.

[5] Hoppe, H.: Surface Reconstruction from Unorganized Points. Seattle, Washington, USA, Univ., PhD Thesis., 1994.

[6] Hoschek, J. , Dietz, U.: Smooth B-Spline Surface Approximation to Scattered Data. In: Reverse Engineering / Hoschek, J.; Dankwort, W. (ed.). Stuttgart: Teubner 1996, pp. 143-152.

[7] Knorpp, R.: Formleitlinien fuer die Flaechenrueckfuehrung – Extraktion von Kanten und Radiusauslauflinien aus unstrukturierten 3D-Messpunktmengen. Springer, Stuttgart 1998.

[8] Lange, M.: Segmentierung von Konturen auf der Basis von Kruemmungsberechnungen. In: 13. DAGM-Symposium Muster-erkennung 1991. Radig, B. (Ed.). Berlin: Springer, 1991.

[9] Nievergelt, J., Hinrichs, K.: Programmierung und Datenstrukturen. Berlin u.a.: Springer, 1986.

[10] Piegl, L., Tiller, W.: The NURBS Book. Berlin u.a.: Springer 1997.

[11] Reinsch, C.H.: Smoothing by Spline Functions. In: Numerische Mathematik 10 (1967), pp. 177-183.

[12] Roth, G., Levine, D.: Extracting geometric primitives. Computer Vision, Graphics Image Processing: Image Understanding 58 (1993) No. 1, pp. 1-22.

[13] Roth-Koch, S.: Merkmalsbasierte Definition von Freiformgeometrien auf der Basis räumlicher Punktwolken. Springer, Stuttgart 1995.

[14] Santarelli, P.: Glaettungskriterien und Algorithmen zum Modellieren von Kurven und Oberflaechen. PhD thesis, Universitaet Kaiserslautern 1994.

[15] Schoenberg, I.J.: Spline functions and the problem of graduation. Proc. Nat. Acad. Sci. 52 (1964), pp. 947-950.

[16] Schuhmann, N.: Konstruktion von Freiformflaechen aus unregelmaessig verteilten Messpunkten. Diploma thesis, Universitaet Stuttgart 1997.

[17] Sedgewick, R.: Algorithms. Addison-Wesley, New York 1991.

[18] Trucco, E., Fisher, R. B.: Experiments in Curvature-Based Segmentation of Range Data. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (1995) No. 2, pp. 177-182.
[19] Varadi, T., Martin R., Cox J.: Reverse engineering of geometric models – an introduction. In: Computer-Aided Design 29 (1997) No.4, pp.255-268.
[20] Wang. W., Iyengar, S. S.: Efficient Data Structures for Model-Based 3-D Object Recognition and Localization from Range Images. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (1992) No. 10, pp. 1035-1045.
[21] Wani, M. A., Batchelor, B.G.: Edge-Region-Based Segmentation of Range Images. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (1994) No. 3, pp. 314-319.