Zuerst ersch. in: IEEE International Conference on Shape Modeling and Applications, 2007 : SMI '07 ; 13 - 15 June 2007, Lyon, France / in cooperation with ACM SIGGRAPH... . - Los Alamitos, Calif. [u.a.] : IEEE Computer Society, 2007. - S. 123-136. - ISBN 978-0-7695-2815-1

Robust Smooth Feature Extraction from Point Clouds

Joel Daniels II[1] Linh K. Ha[1] Tilo Ochotta[2] Cláudio T. Silva[1]

[1] *University of Utah*
[2] *University of Konstanz*

Abstract

Defining sharp features in a given 3D model facilitates a better understanding of the surface and aids visualizations, reverse engineering, filtering, simplification, non-photo realism, reconstruction and other geometric processing applications. We present a robust method that identifies sharp features in a point cloud by returning a set of smooth curves aligned along the edges. Our feature extraction is a multi-step refinement method that leverages the concept of Robust Moving Least Squares to locally fit surfaces to potential features. Using Newton's method, we project points to the intersections of multiple surfaces then grow polylines through the projected cloud. After resolving gaps, connecting corners, and relaxing the results, the algorithm returns a set of complete and smooth curves that define the features. We demonstrate the benefits of our method with two applications: surface meshing and point-based geometry compression.

1. Introduction

Digital scanner technology has become more affordable and accurate, increasing its popularity and utility. These scanners collect a dense sampling of points, with mechanical probes or lasers, to generate a virtual representation of a physical form. Consequently, geometric processing of point clouds is becoming increasingly important.

The preservation of sharp features is a primary concern for many geometric computations and modeling applications. In this context, a feature is described as the discontinuity in the surface normals evaluated on the model. In such regions the G^1 continuity is not maintained. Identification of these edges facilitates a better understanding of the model improving filtering [6, 20], reconstruction [17], refinement or resampling [4], simplification [8], smoothing and visualization [2] methods.

Features can be computed through the investigation of changes in the normals of neighboring discrete surface data, i.e. polygonal facets or vertices. The most simple method is to implement a threshold test that identifies potential feature edges where normals differ above some tolerance level across adjacent samples. However, the implementation of such detection techniques is not straightforward, since in most cases, normals and connectivity are not provided in point data sets and their reconstruction is a nontrivial problem, especially when sharp features must be preserved.

The noise inherent in scanned models presents a second major challenge to feature edge extraction in point clouds. The identification of feature lines is complicated in the presence of noise due to the fact that the large surface gradients appearing in regions with sharp features are similar to those found in noisy areas.

In this paper, we present an algorithm to define a set of curves that are aligned along the feature edges of a point cloud. Based upon the framework of *Robust Moving Least Squares* (RMLS) surfaces [7], we first select a number of potential feature points that are identified by the RMLS operator to be near to possible features. One issue with the RMLS is the inability to reconstruct smooth features. The projection method produces jagged edges that cannot be used without further processing to construct smooth feature curves. Therefore, we propose to apply a smoothing method on the RMLS projected feature points using a technique based on principal component analysis (PCA), followed by a feature growing strategy that constructs a set of polylines. Finally, we account for poor sampling quality by proposing a technique to connect the multiple extracted polylines across gaps to create a complete set of feature curves. The pipeline is illustrated in Fig. 1.

The advantage of our approach is that in contrast to previous work [10], our input data consists of a set of unorganized points (possibly affected by noise) that approximate the original surface. We also assume that no attribute information, e.g., normal vectors, are given a priori. The output feature curves are described as smooth and complete, which, in the presence of closed

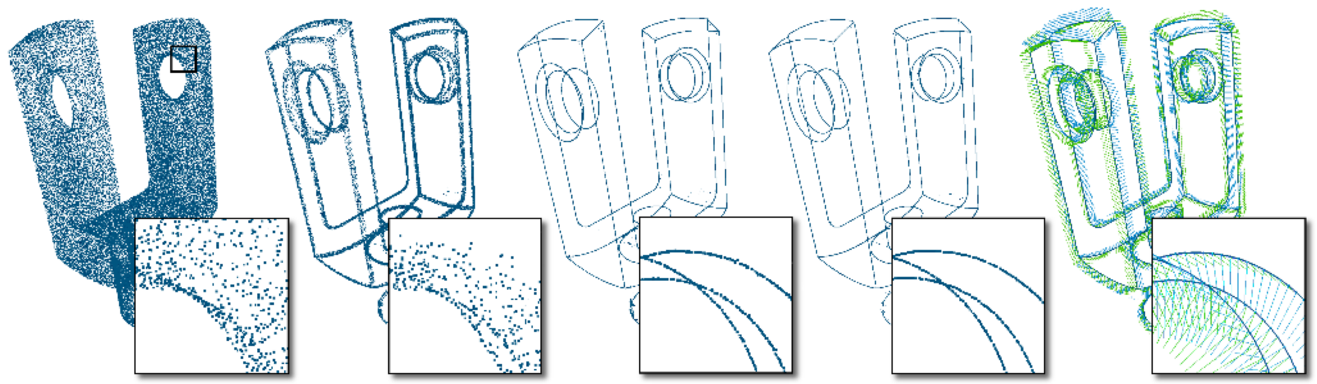


Figure 1. Overview of our feature extraction pipeline; from left to right: original point cloud; candidate points identified to be close to possible features; candidate points projected onto the intersection of RMLS surfaces; smoothed projected points; reconstructed polylines that identify the sharp features (with normal vectors for the defining surfaces).

loops, allows us to extract surface patches that are enclosed by the polylines. Furthermore, we show that this information can be used to improve previously developed techniques that operate on point-sampled geometry, e.g., surface meshing and compression.

In Section 2, we review related work in the field of feature line extraction on mesh- and point-based models. Section 3 describes our feature extraction pipeline. In Section 4, we apply our results to improve surface reconstruction and the compression of point-based models. Section 5 presents experimental results, and we conclude the paper with final remarks and future work in Section 6.

2. Related Work

2.1 Mesh-Based Feature Extraction

Multiple techniques have investigated the identification of feature edges on polygonal models. Hubeli et al. [11] create a multi-resolution framework and normal-based classification operators to define a set of edges on which a thinning process extracts feature lines. Watanabe et al. [19] use discrete differential geometry operators to compute approximations of the mean and Gaussian curvatures. Whereas, Hildebrandt et al. [10] use anisotropic filtering on discrete differential geometric approximations of third order derivatives of the surface mesh. Both techniques [10, 19] build a set of feature edges from the extrema triangles. Attene et al. [4] identify chamfer triangles to reconstruct sharp features on meshes. They insert new vertices on the edges and faces of these triangles and project the points to the intersection of planes fit to the surrounding surfaces. The underlying assumption of connectivity and normals associated with the vertices of the mesh is not available for point-based models. In order to extract feature lines from point clouds using these techniques, a connectivity construction method (surface reconstruction) must be applied in a preprocessing step. The construction of connectivity is non-trivial, computationally expensive, and moreover, the success of feature extraction relies on the ability of the polygonal meshing procedure to accurately build the sharp edges.

2.2 Point-Based Feature Extraction

Feature line extraction from point-based models is not straightforward in the absence of connectivity and normal information. Pauly et al. [16] use covariance analysis of the distance-driven local neighborhoods to flag potential feature points. By varying the radius of the neighborhoods, they develop a multi-resolution scheme capable of processing noisy input data. Gumhold et al. [9] construct a Riemann graph over local neighborhoods and use covariance analysis to compute weights that flag points as potential creases, boundaries, or corners. Both techniques [9,16] connect the flagged points using a minimum spanning tree and fit curves to approximate sharp edges. Demarsin et al. [5] compute point normals using principal component analysis and segment the points into groups based on the normal variation in local neighborhoods. A minimum spanning tree is constructed between the boundary points of the assorted clusters, which is used to build the final feature curves.

These techniques are capable of extracting features on point clouds by connecting existing points; however, their accuracy depends on the sampling quality of the input model. In contrast to their work, our method relies on projecting points onto sharp edges, yielding more accurate approximations of features in the original surface.

Jenke et al. [12] identify sharp features to improve their Bayesian statistic based reconstruction algorithm. While their method uses a similar curvature test to identify potential edge regions, their classification method limits identification to complete edges. We are able to define edges that dissipate as the two defining surfaces smooth towards a single surface. Additionally, our classification method is computed on the original data set; whereas, [12] performs a optimized smoothing on the geometry.

3. Feature Extraction Algorithm

3.1. Preliminaries

Our method operates on an unorganized set of points $\mathcal{P} = \{(x, y, z)^T\} \subset \mathbb{R}^3$ that sample some original surface $\mathcal{S}$. Note that we do not rely on attributes, such as associated point normal vectors or connectivity information. Given $\mathcal{P}$, we set $\mathcal{S}$ to be the *Moving Least Squares* (MLS) surface defined by a projection operator Ψ that is applied to an arbitrary point $p \in \mathbb{R}^3$ in order to project it onto the surface, $\Psi(p) \in \mathcal{S}$. Each point on the surface projects onto itself, $\mathcal{S} = \{p \mid p = \Psi(p)\}$.

There are several options of selecting the projection operator Ψ. Most variants implement the projection of a point $p \in \mathbb{R}^3$ through finding a plane H that approximates a local neighborhood around p in the set $\mathcal{P}$. Specifically, the plane H is defined by a point $q \in \mathbb{R}^3$ and a normal n, $H = H(n, q) = \{x \in \mathbb{R}^3 \mid \langle x - q, n\rangle = 0, \|n\| = 1\}$, and found, such that the sum of weighted squared distances of points in $\mathcal{P}$ to the plane H, is minimized:

$$(n, q) = \arg \min_{(n,q)} \sum_{p \in \mathcal{P}} \langle n, p - q\rangle^2 \theta(\|p - q\|), \qquad (1)$$

where $\theta(\cdot)$ is a exponentially decreasing weighting function, e.g., $\theta(s) = e^{-s^2/h^2}$, which assigns larger weights to points near q. The parameter h defines the spatial scale of θ. Once H is found, it serves as a reference for a local Cartesian coordinate system with origin in q and two span vectors that are orthogonal to n. Then, a bivariate polynomial g of a given degree is fit in order to find the projected point $\Psi(x)$, which is above q and on the surface $\mathcal{S}$. This polynomial fitting corresponds to finding

$$\arg \min_{g} \sum_{p \in \mathcal{P}} \|p - p_g\|^2 \theta(\|p - q\|), \qquad (2)$$

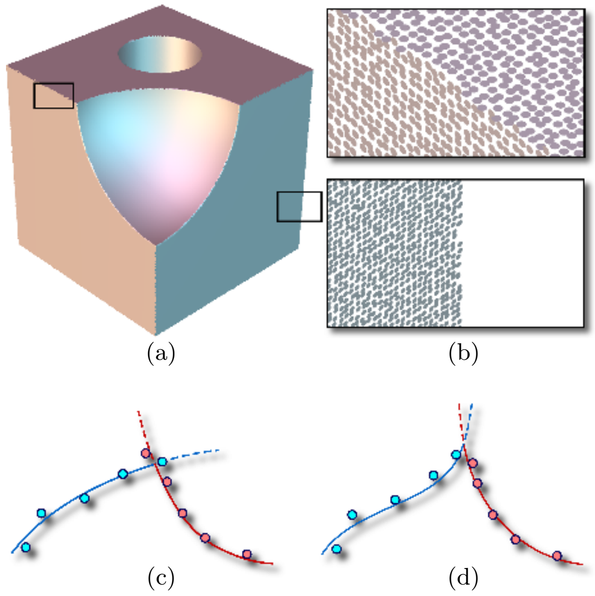


Figure 2. RMLS sub-neighborhoods are computed independently for each point, which creates jagged edges, (a) and (b), due to the fact that nearby points may construct different neighborhoods; this leads to a disagreement on the configuration of the intersecting surfaces, (c) and (d).

where p_g denotes the projection of p onto the polynomial g in direction of the plane normal n. The function $\theta(\cdot)$ again is the exponentially decreasing weighting function. Originally proposed by Levin [14], MLS surfaces have been improved in many ways [1–3, 7].

The MLS projection relies on the minimization of (1) and (2), which includes Gaussian weighting of points in the local neighborhoods. This weighting leads to a smoothing effect that corrects noise and outliers in the set $\mathcal{P}$, but simultaneously removes sharp features. The RMLS variant [7] was developed to solve this problem by considering iteratively constructed neighborhoods based on statistical analyzes of the corresponding point distributions. In particular, they use least median of squares, which is a regression method to minimize the median of absolute residuals between the point set and the fit. Their forward search algorithm grows multiple MLS neighborhoods in order to find smooth, flat, and outlier free regions (*sub-neighborhoods*) for the final MLS projection.

This approach is inherently capable of detecting noise by disregarding outliers during the surface fitting phase, and defines sharp features. Moreover, it reduces the smoothing effect of the traditional MLS projection and improves numerical stability. However, the drawback is that the clustering is computed in-

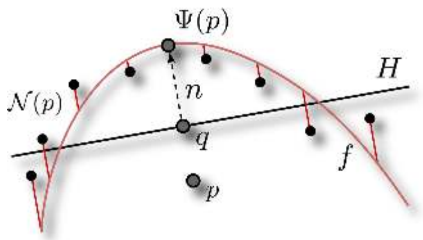


Figure 3. The residual $r(p)$ is computed as the maximum distance between the points in the neighborhood around p and the polynomial fit f, whereas the distances are considered in direction of the plane normal n.

dependently for all points, which may lead to situations where nearby points do not grow to identical sub-neighborhoods, resulting in jagged edges. Figures 2a and 2b illustrate the occurrence of jagged edges, created by the RMLS projection. Figures 2c and 2d show two different sub-neighborhood constructions for one set of points that leads to different configurations of the feature region.

To overcome this problem, we propose to apply a smoothing procedure to the RMLS projected points. The goal of the following five-step algorithm is to extract smooth and complete curves that approximate the features well:

(1) Extract points from $\mathcal{P}$ that are near potential features,

(2) Use RMLS to fit multiple surfaces to the neighborhoods of these points and project each point to its nearest intersection between the surfaces,

(3) Smooth the projected points using covariance analysis of adaptively grown neighborhoods,

(4) Create an initial set of feature lines by growing polylines that approximate the smoothed projected points,

(5) Analyze the end points of the feature polylines to complete gaps and connect to corner points.

The following subsections discuss the algorithmic details of each of the five steps.

3.2. Identifying Potential Edge Regions

To identify potential edge regions, we adopt the method of Fleishman et al. [7]. Specifically, given a point $p \in \mathcal{P}$, they consider the neighborhood $\mathcal{N}(p) \subset \mathcal{P}$ that is used for the polynomial fit of the MLS projection. Considering the polynomial f over the reference plane H that is defined by point q and normal n,

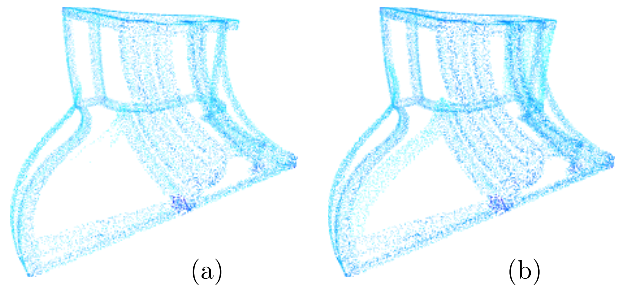


Figure 4. Identification of potential feature points for the Fandisk model; the threshold τ is used to adjust the number of points retained near potential features; the automatically computed threshold $\tau = 0.057$ (a) and a manually adjusted threshold $\tau = 0.014$ (b).

they evaluate the maximum polynomial residual r in a neighborhood around p,

$$r(p) = \max_{x \in \mathcal{N}(p)} \|x - x_f\|,$$

where x_f corresponds to the projection of x onto f in direction of the plane normal n, see Fig. 3.

For sharp features, r becomes reasonably large because many point sampled along the sharp edge lift away from the polynomial. Based on this observation, they straightforwardly define any point $p \in \mathcal{P}$ to be a potential feature point, if $r(p) > \tau$ for some user defined threshold τ. Note that strong outliers will also record large r; however, they are eliminated during the RMLS projection stage.

This approach relies on an appropriate selection of the threshold τ. We extend the pipeline in [7] by using an automatic computation method, which produces an adaptive threshold. In particular, we found that setting τ to the mean residual value taken over all points in $\mathcal{P}$, $\tau = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} r(p)$, leads to satisfying results. Moreover, we allow the user to modify τ manually, using an interactive tool. Figure 4 shows the Fandisk model after identifying potential feature points using different values for the threshold τ. The output of the identification procedure is a set of potential feature points, i.e. $\mathcal{F} = \{p \in \mathcal{P} \mid r(p) > \tau\}$.

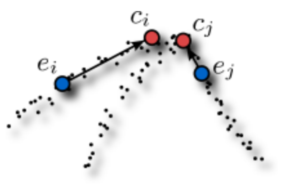
3.3. Projecting Points to Edges

We use the RMLS procedure to project the points $p \in \mathcal{F}$ towards the features. The output of the procedure is the definition of a number of surfaces fit to the points in the neighborhood around p. This number describes the feature types in the region.

If RMLS returns a single surface, then this implies that no feature was detected and the high residual was produced by outliers. Two surfaces indicate the presence of a sharp feature. We now use Newton's method, as in [16], to project p onto the intersection of the two surfaces closest to p, yielding an *edge point* e. Note that e does not correspond to the RMLS projected point, it is rather the result of the Newton projection seeded at the RMLS projection of p. If three or more surfaces are found, then there are multiple sharp features and a *corner point* c is produced in addition to an edge point e. At this stage, the output of the projection procedure is a feature point cloud $\mathcal{F}^{(p)}$ that we compose of edge points $\mathcal{E}^{(p)}$ and corner points $\mathcal{C}^{(p)}$, $\mathcal{F}^{(p)} = \mathcal{E}^{(p)} \cup \mathcal{C}^{(p)}$.

The RMLS projection stage results in multiple corner points projected to each region formed by three or more intersecting surfaces. This is due to the occurrence of jagged edges as described in section 3.1. The goal is to group close corner points to retrieve a single point identifying the feature corner. We achieve this by weighting the corner points to prioritize their importance during the projection stage.

Specifically, for each corner point $c_i \in \mathcal{C}^{(p)}$ that was projected from $p_i \in \mathcal{F}^{(p)}$, we consider the corresponding projection to the two closest surfaces, $e_i \in \mathcal{E}^{(p)}$. We now assign a weight w_i to the point c_i being inversely proportional to the distance between c_i and e_i, $w_i = 1/\|c_i - e_i\|$. The figure shows two close corner points c_i and c_j with different weights. We now group the corner points $\mathcal{C}_k^{(p)} \subset \mathcal{C}^{(p)}$ by considering neighborhoods of a predefined feature radius ϵ and select $\hat{c} \in \mathcal{C}_k^{(p)}$ that is the closest to the weighted average of the corner points in the group. All other points in $\mathcal{C}_k^{(p)}$ are defined to be edge points and are moved from $\mathcal{C}^{(p)}$ to $\mathcal{E}^{(p)}$.



3.4. Smoothing the Feature Points

The output of the projection procedure is a set of points $\mathcal{F}^{(p)}$ that are expected to identify the sharp features. However, the points in $\mathcal{F}^{(p)}$ are affected by noise, since they were obtained using the RMLS operator, which tends to produce jagged edges. Inspired by [13], we implement a smoothing filter based on the covariance analysis of points in local neighborhoods.

Specifically, given $p \in \mathcal{F}^{(p)}$, we consider the set of k feature points in the δ-neighborhood of p identified by $\mathcal{N}(p) = \{p_i \in \mathcal{F}^{(p)} \mid \|p_i - p\| \leq \delta\}$, $i = 1, \ldots, k$. We now compute the 3×3 covariance matrix

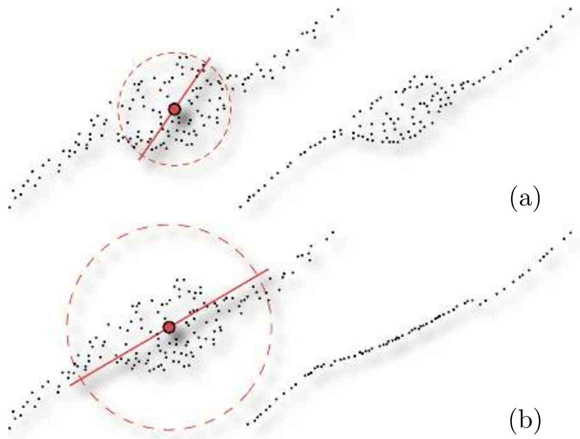


Figure 5. Feature cloud smoothing using different neighborhood sizes δ.

$$\mathbf{C} = (p_1 - \bar{p}, \ldots, p_k - \bar{p}) \cdot (p_1 - \bar{p}, \ldots, p_k - \bar{p})^T,$$

where $\bar{p}$ is the mean, $\bar{p} = \frac{1}{k}\sum_{i=1}^{k} p_i$. We solve the eigensystem $\mathbf{C}\mathbf{x}_i = \lambda_i \mathbf{x}_i$, $i \in \{0, 1, 2\}$ and assume (without loss of generality) that $\lambda_0 \geq \lambda_1 \geq \lambda_2$. The eigenvector $\mathbf{x}_0$ that corresponds to the largest eigenvalue λ_0 is expected to be aligned with the feature line in the neighborhood $\mathcal{N}(p)$. We achieve a smoothing effect by projecting feature points p onto the line defined by the mean $\bar{p}$ and the eigenvector $\mathbf{x}_0$, yielding p'.

The overall smoothing performance is determined by the neighborhood size δ, see Fig. 5. In order to obtain stable smoothing in the presence of noise and high curvature, we adopt the adaptive neighborhood growth scheme in [13]. This method is based on a statistical analysis of point distributions on the plane.

In particular, given a point p, the algorithm fits a local plane to points p_i in the neighborhood $\mathcal{N}(p)$ and projects them onto the plane that is defined by the two eigenvectors, which correspond to the two largest eigenvalues. The resulting point set (in $\mathbb{R}^2$) is then interpreted in terms of a distribution of two *random variables*, X and Y, which can be analyzed by statistical methods. The cross-correlation coefficient of X and Y is defined by $\rho(X,Y) = \frac{cov(X,Y)}{\sigma(X)\sigma(Y)}$, where $cov(X,Y)$ is the covariance of X and Y, and $\sigma(\cdot)$ is the standard deviation. Large absolute values for ρ indicate that there is a linear relationship between X and Y, meaning that the points in $\mathcal{N}(p)$ are distributed along a feature line.

Given p, the adaptive neighborhood growth algorithm increases the neighborhood $\mathcal{N}(p)$ by iteratively adding points closest to p from $\mathcal{F}^{(p)}$ until a sufficiently large correlation coefficient ρ is found. In practice we found

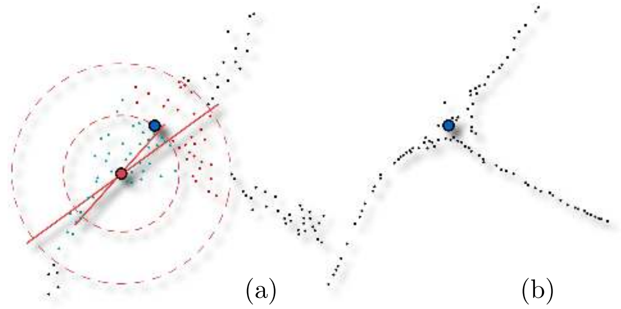


Figure 6. The distance to the nearest corner point also caps the neighborhood size for the smoothing to reduce the number of points considered that belong to another feature (shown in red).

that $|\rho| \geq 0.7$ leads to adequate results. The neighborhood growing procedure is terminated when a user-defined correlation is reached. Moreover, neighborhoods are not grown beyond identified corner points (Fig. 6), since the neighborhood is likely to start capturing multiple features, which complicates the identification of the feature lines.

Given the set of projected feature points $\mathcal{F}^{(p)}$, the output of the smoothing procedure is the set of smoothed feature points denoted by $\mathcal{F}^{(s)}$ that is again decomposed into edge points $\mathcal{E}^{(s)}$ and corner points $\mathcal{C}^{(s)}$.

3.5. Feature Polyline Propagation

At this stage of the pipeline, the set of feature points $\mathcal{F}^{(s)}$ is unorganized and contains no topological information. In many practical applications, however, continuous feature curves are desired. The goal of the feature polyline propagation technique is to approximate the feature points $\mathcal{F}^{(s)}$ with a set of polylines. The user defines the maximum allowable segment length s_{max} as an input parameter to control the coarseness of the feature polylines.

We build a polyline construction algorithm based on Lee's method [13], which works as follows. The feature polyline is initialized at a seed point p, where the PCA for points in the neighborhood $\mathcal{N}(p) \subset \mathcal{F}^{(s)}$ with radius s_{max} is computed, Fig. 7(a). All points $p_i \in \mathcal{N}(p)$ are projected onto the line defined by p and the eigenvector that corresponds to the largest eigenvalue of $\mathcal{N}(p)$, yielding points p'_i. The two points p_j, $p_k \in \mathcal{N}(p)$ with the furthest corresponding projections p'_j and p'_k in opposite directions from p are considered to be endpoints of two new segments in the feature, Fig. 7(b,c). The procedure is repeated with p_j and p_k, Fig. 7d, until no more points are found for new connections.

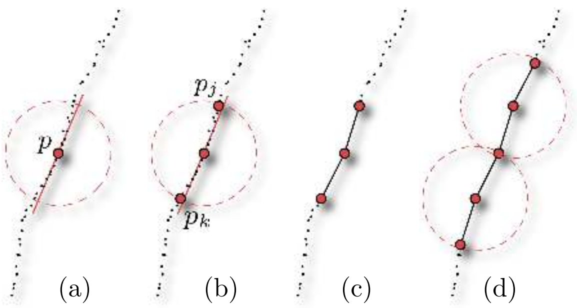


Figure 7. Curve growing starts at a seed point, finds all points in a user specified radius, projects them to the major eigen axis, and connects to the furthest points. The processed points are removed and the method is repeated.

For our models we found that best results are achieved by starting the feature growing procedure at edge points that are preferably far away from corner points. Our implementation maintains a priority queue that sorts all edge points $p \in \mathcal{E}^{(s)}$ according to their distance to the closest corner point $q \in \mathcal{C}^{(s)}$. As long as the queue is not empty, there are unvisited feature points remaining. To create a new feature line, we remove the first element from the queue, followed by applying the propagation algorithm as described above. Once the feature points are connected, we remove the elements from the queue that correspond to points within the neighborhoods $\mathcal{N}(p)$ of processed points p.

Although, the polyline growing scheme replaces the points with curve approximations, it is not yet complete due to gaps that occur in regions with poor sample quality. We observed that in these regions there are multiple separated polylines that describe single features. The next stage of our pipeline addresses this problem by connecting close feature curves to produce complete polylines.

3.6. Completing Feature Curves

The final stage cleans the feature polyline representations in order to produce complete and smooth curves along the sharp edges of the model. This procedure visits all end points of feature polylines to search for gaps between multiple polylines and regions where new corners have to be constructed. Moreover, polylines are smoothed in order to create complete and smooth feature representations.

Our feature completion method is driven by a directional search approach. As illustrated in Fig. 8, we consider the tangent vector that is evaluated for an endpoint by fitting a cubic polynomial to the last four

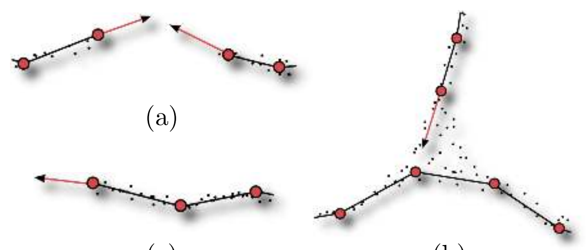


Figure 8. Feature completion requires an analysis of the endpoints of each feature curve to determine if it must fill a gap between multiple features (a), split a single feature into two thus forming a new corner point (b), or be left alone unattached (c).

points of each end of the feature polyline. An alternative approach is to evaluate the direction of the last segment of the feature curve; however, we found that the cubic fitting leads to better results.

For each end point p, we search for other feature points that are within the cone formed by the tangent vector at p and a predefined aperture angle. We search for the three cases in Fig. 8: *gap completion* (a), *corner creation* (b), and *endpoint identification* (c). The search algorithm finds the closest vertex of the feature polylines that exists within the volume of the search cone that is within the distance αs_{max}. The scalar α should be adjusted based on the sampling quality along the features of the cloud. In practice we found that $\alpha = 7$ accommodates well for the sampling of the models in this paper.

Figure 8(a) shows an example for the case of gap completion, which occurs due to poor sampling quality, i.e. the distance between two end points is larger than s_{max}. To resolve this problem, we merge any two polylines with endpoints that are within αs_{max} distance to each other and corresponding tangent vectors that point to opposing directions within the aperture angle.

Corner creation occurs if the end point of a polyline projects to an interior point p of another feature line, Fig. 8(b). To prevent the production of two corner points, we split the corresponding line segment, insert a new corner point at p, and reconnect the involved line segments accordingly. This case resolves situations where a single polyline missed a corner during the growing process.

The last case, Fig. 8c, occurs if a feature line dissipates as detected on the Fandisk model. If no vertices of feature lines are found during the directional search, then we declare the end point to belong to a dissipating feature edge, and no connection operation is performed.

After the endpoints have been resolved the feature curves are considered complete. Perturbations are smoothed from the feature polylines by fixing the endpoints and applying Laplacian smoothing on the internal vertices weighted by inverse distances to the target point. This process is limited to 1 or 2 passes to minimize shrinking on the features. Finally, we consider the feature polylines to be complete and smooth.

4. Applications

The output of our method is a set of connected and smoothed polylines that identify the sharp features of a point-sampled surface. We are also able to extract surface patches that are bounded by the feature lines. In this section, we will show the benefits of applying the feature line information as input data for methods that rely on point sets. We focus on surface reconstruction and shape compression.

We demonstrate the effectiveness of our feature extraction technique using point models that represent machine parts with sharp edges, such as the Quarter Piston model, the Ra model, the Scallop model, and the well-known Fandisk model. We emphasize that all data sets consist only of a set of 3D points. We do *not* assume associated normal vectors for the geometry.

4.1. Surface Reconstruction

Triangle meshes are a popular representational form for models that are commonly used in geometry processing. The algorithms leverage additional information inherent in mesh models versus point set surfaces, mainly connectivity, to compute discrete differential operations. Often, the robustness and efficiency of the algorithms are dependent on the quality of the input mesh. This demand for high quality mesh structures has motivated much of the development in surface construction and remeshing algorithms.

Schreiner et al. [18] extend the approach in [17] and proposed an advancing-front algorithm that produces high quality meshes from many different types of source models, including point-set surfaces. Their approach is based on defining a *guidance field*, based on local curvature information to determine the size of the triangles. Precomputing a specific guidance field allows the triangulation scheme to adapt to areas of high curvature, shrinking the allowable triangle size as the advancing fronts grow towards such regions.

The algorithm uses MLS for projection and curvature computations on point clouds. Consequently,

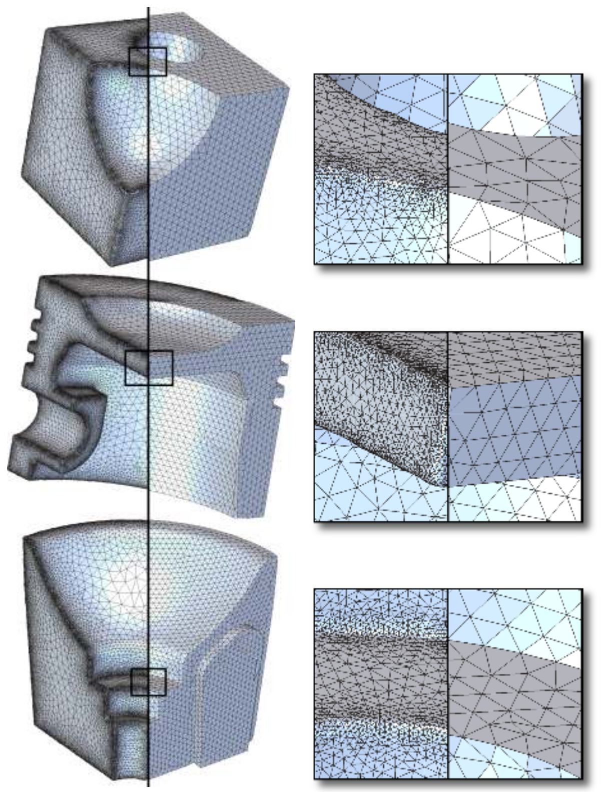


Figure 9. Results obtained by applying the meshing technique in [18] to Scallop (top), Quarter Piston (middle), and Ra (bottom); the left half of each closeup view shows meshes obtained by the traditional method; the corresponding right halves show the meshes after initializing the input of [18] with feature lines extracted with our method.

their method adaptively shrinks the triangle sizes near sharp features and produces rounded edges. Extending their projection and guidance field computations by using RMLS enables reconstructions with sharp features; however, as previously discussed, this creates jagged edges and additional computational costs.

In response to this problem, we guide the method by including the extracted feature curves with evaluated normals for the two meeting surfaces. Using these feature lines as the initial polylines of the advancing front, the remeshing system is able to generate a triangle mesh with sharp features without changing the MLS methods. Consequently, the projection computation is not effected, while the algorithm automatically produces models with smooth sharp features, Fig. 9.

Moreover, the feature lines aid the computation of a better guidance field. Previously, sharp features compute large curvature values such that the guidance field

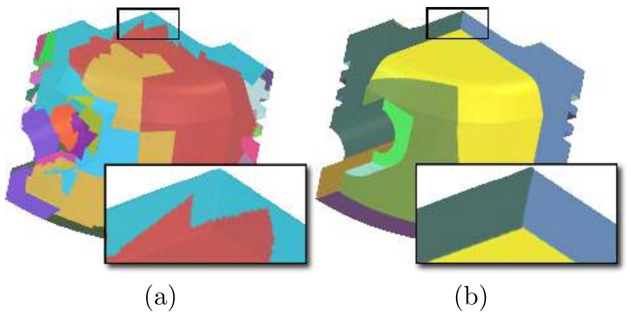


Figure 10. The partition of Quarter Piston using the traditional patch composition method in [15] (a); partition with initial segmentation (b).

scales the triangles to a very small size to capture an MLS smoothed region. By initializing the advancing front to the feature edges, it is known that curvature values computed here are misleading; therefore, larger triangles can be used to approximate the surface grown from the sharp edges. The mesh reconstructed using the feature polylines has a significantly reduced final triangle count, Fig. 9.

4.2. Surface Compression

Surface compression is an elementary problem in geometry processing, the goal of which is to represent 3D models compactly for the purpose of space efficient storage and fast transmission. The task of the encoder is to transform a specific surface representation into a compact bit stream, which can be decoded at the receiver side in order to reobtain the original model or an approximation of it, if the encoding is lossy.

In [15] it was shown that a point model can be compressed by decomposing the input surface into a number of patches, which are parameterized as height fields over planar domains and resampled on regular grids. The resulting images with arbitrary regions of support are encoded using state-of-the-art wavelet compression of shaped images. One key feature in their method is the surface parameterization that is based on a split-merge partitioning procedure. In the first step, the surface is recursively subdivided until all patches meet some prescribed surface flatness constraint. In the consequent phase, adjacent patches are merged as long as the resulting patches show height field properties.

Although their method yields a high rate-distortion performance, the approximation quality depends heavily on the structure of the surface partition. Especially for models with sharp features, the constructed partitions show patches that contain the features, rather than identifying the features as patch boundaries. The

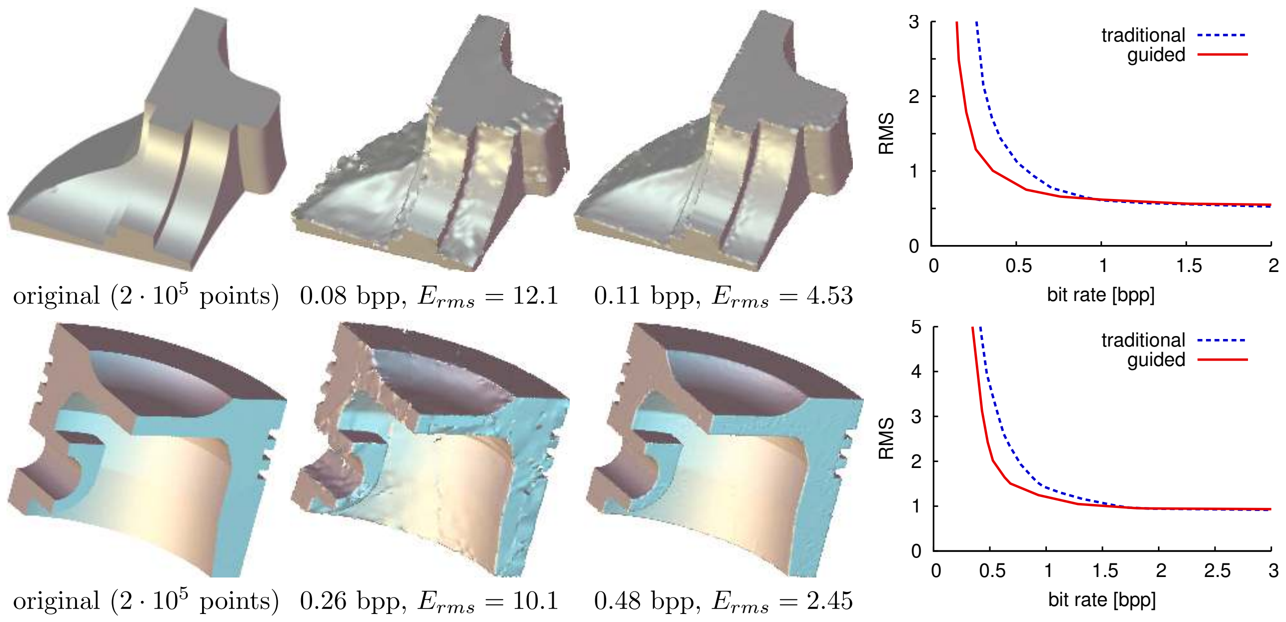

original ($2 \cdot 10^5$ points)
0.08 bpp, $E_{rms} = 12.1$
0.11 bpp, $E_{rms} = 4.53$
original ($2 \cdot 10^5$ points)
0.26 bpp, $E_{rms} = 10.1$
0.48 bpp, $E_{rms} = 2.45$
traditional
guided
RMS
bit rate [bpp]
traditional
guided
RMS
bit rate [bpp]

Figure 11. Rate distortion performance for Fandisk (top row) and Quarter Piston (bottom row); the left figure shows the original model, followed by two decodings for different rates; the rate-distortion curves show that the compression performance is significantly increased by applying the partitioning in [15] based on feature aware segmentation in this work (right); the error is the root-mean-square error in [15].

major drawback of this behavior is that sharp features within patches correspond to high frequency components in the corresponding wavelet signal, which are consequently hard to encode.

To overcome this problem, we propose to guide the partitioning procedure in [15] by the sharp features that were constructed with our method. In particular, the encoder receives the input model with attached pre-partition, which is extracted from the closed curves, see section 3.6. The splitting operations in the partitioning step are performed on each individual patch in the pre-partition, while the merging is performed as in the usual setting.

The advantage of this approach is that the features give the encoder a first guess of the surface structure to build the patch layout, Fig. 10b. The individual patches in the resulting partition show an improved structure as well as more regular boundary shapes. In contrast, the traditional partitioning method does not recognize edges sufficiently, leading to patches that wrap around sharp features, Fig. 10a.

Figure 11 shows compression results for Fandisk (top row) and Quarter Piston (bottom row) at $2 \cdot 10^5$ points each. The left images show the original models followed by two decoded models at different bit rates. For the Fandisk model we observe heavy compression artifacts only at extremely low bit rates, e.g., 0.08 bits per point (bpp). Slightly increasing the bit rate leads to a significantly better geometric reconstruction quality. For the Quarter Piston model (Fig. 11 bottom row), however, we observe that more bits need are needed (0.26 bpp) to achieve an adequate reconstruction quality. We explain this behavior by the fact that this model shows more complex geometric properties that the Fandisk. Further increasing the bit rate, e.g., to 0.48 bpp, yields reconstructions that are close to the original model.

5. Experimental Results

We discuss the robustness of our method to noise and its run time efficiency. Figure 12 shows extracted feature lines for the Quarter Piston model at different noise levels. We applied uniform noise to the original model by shifting each point p by a random vector $\delta(p)$ of restricted length, $\|\delta(p)\| \leq l$. The length l is the noise level, which is expressed in units of the bounding box diagonal length d_B of the original model. The results in Fig. 12 show that our method is robust to noise and consequently extracts smooth feature lines. Only for high noise levels, we observe that the feature line completion step fails to connect all feature lines.

The total time required to construct the polylines is determined by the number of *potential* feature points

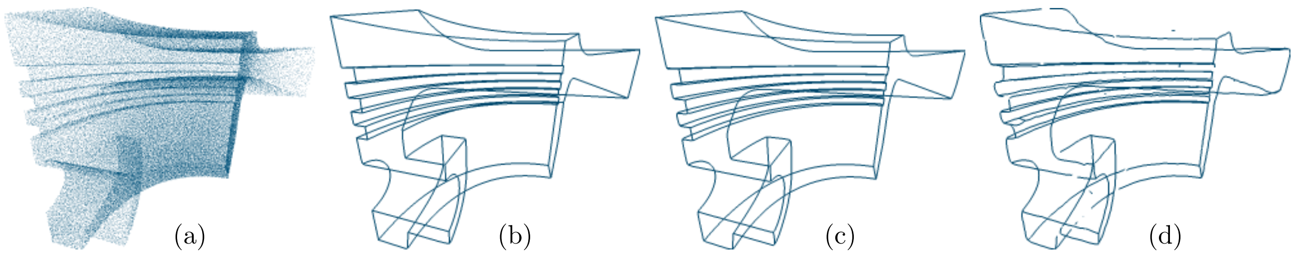


Figure 12. Feature extraction from the Quarter Piston model; original point cloud (a); features extracted from the original model (b); features extracted from the original model after applying uniform noise $\|\delta(p)\| \leq 0.01 d_B$ (c), and $\|\delta(p)\| \leq 0.05 d_B$ (d), where d_B is the length of the model bounding box diagonal.

in data set, since only the feature point identification stage operates on the complete point set. Table 1 shows run times for all of the five stages in our algorithm for different models at varying noise levels. We found that an increase in the noise level leads to a slight decrease in computational costs, which can be explained by the fact that the threshold τ becomes larger for noisy models, which consequently results in a reduced number of potential feature points. Carried out on a Pentium4 2 GHz platform, our implementation needs four to ten minutes to process models of half a million points, where the major part of computational costs is required for the RMLS projection (stage 2).

The robust feature extraction produces smooth results, but can be limited by an extremely poor sampling quality and by time constraints. High noise levels or sparse sampling worsens the performance of our method, since the distinction between sharp features and noisy regions becomes ambiguous. Because RMLS is computationally expensive, the algorithm is run as a pre-process and its output improves the performance of other interactive methods.

Figure 13 illustrates the algorithm output on several different models with varying feature types. The refinement and smoothing that occurs over the multiple stages removes perturbations, while extracting an accurate representation of the sharp features. Additionally, note that the extracted feature polylines are coupled with normal vectors. We couple the projected points with the RMLS evaluated normals during the projection phase that identifies the edges. The feature growing method groups normals of neighboring polyline vertices to establish consistent vector orientation.

6 Conclusions and Future Work

We presented a method for extraction of feature curves on point-sampled surfaces. Our algorithm leverages the robust statistical methods of RMLS to project points

model	noise	τ $[10^{-2}]$	$\lvert\mathcal{F}\rvert$	run time [seconds] (1)	(2)	(3)	(4)	(5)	total
Quarter Piston	0.0	0.5	66k	177	348	17	71	0.1	613
	0.01	1.2	62k	179	342	16	60	0.1	598
	0.05	4.7	30k	182	365	10	12	1	570
Fandisk	0.0	0.7	39k	178	198	10	22	0.1	407
	0.01	1.1	37k	180	226	10	19	0.1	435
	0.05	4.8	8k	186	83	2	0.6	2	274
Ra	0.0	0.8	31k	176	164	7	12	0.0	359
	0.01	1.4	28k	176	153	7	10	0.0	347
	0.05	4.7	11k	182	133	3	0.9	0.3	319

Table 1. Run time performance of our feature extraction implementation for models that contain 500000 **points each; the stages in our pipeline correspond to point identification (1), point projection (2), point smoothing (3), curve propagation (4), and curve completion (5).**

to all possible features such that they do not need to be exactly sampled by the input point cloud. The output curves are described as complete and smooth and prove valuable as inputs to existing geometric processing applications for point clouds. We were able to significantly improve the performance of a previously proposed surface reconstruction and a point-compression method without any modifications to their algorithms.

In future work, we aim at improving initial neighborhood selection methods to speed up the RMLS method, improving data segmentation for CAD purposes, and investigating smoothing and resampling methods to maintain the sharp features.

Acknowledgments. This work was partially supported by NSF grants CCR-0310705, CCF-0528201, OISE-0405402, CCF-0401498, EIA-0323604, IIS-0513692, and OCE-0424602, NIH grant 573996, the Department of Energy, an IBM Faculty Award, the Army Research Office, and the DFG Graduiertenkolleg 1042 "Explorative Analysis and Visualization of Large Information Spaces". L. Ha is supported by a Vietnam Education Foundation fellowship.

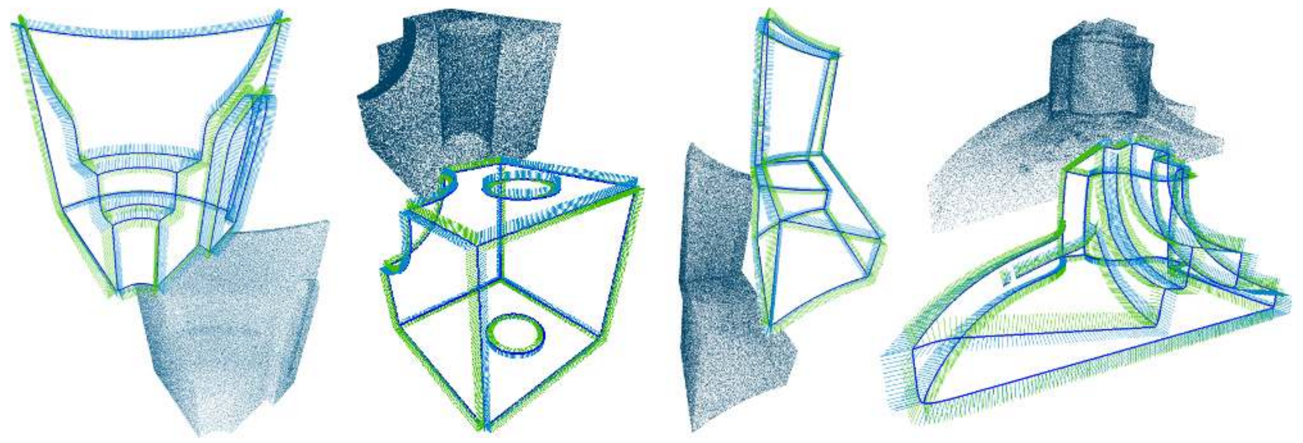


Figure 13. Features extracted with our method; displaying the original point sets without normal information does not show surface characteristics; our method identifies sharp features and can define oriented surface patches without relying on input normal vectors.

References

[1] M. Alexa and A. Adamson. On normals and projection operators for surfaces defined by point sets. In *Symposium on Point-Based Graphics*, pages 149–155, 2004.

[2] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, 2003.

[3] N. Amenta and Y. J. Kil. Defining point-set surfaces. *ACM Transactions on Graphics*, 23(3):264–270, 2004.

[4] M. Attene, B. Falcidieno, J. Rossignac, and M. Spagnuolo. Sharpen-bend: Recovering curved sharp edges in triangle meshes produced by feature-insensitive sampling. *IEEE Transactions on Visualization and Computer Graphics*, 11(2):181–192, March 2005.

[5] K. Demarsin, D. Vanderstraeten, T. Volodine, and D. Roose. Detection of closed sharp feature lines in point clouds for reverse engineering applications. Report TW 458, Department of Computer Science, K.U. Leuven, Belgium, 2006.

[6] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *ACM SIGGRAPH*, pages 317–324, 1999.

[7] S. Fleishman, D. Cohen-Or, and C. T. Silva. Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics*, 24(3):544–552, 2005.

[8] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *ACM SIGGRAPH*, pages 209–216, 1997.

[9] S. Gumhold, X. Wang, and R. McLeod. Feature extraction from point clouds. In *10th International Meshing Roundtable, Sandia National Laboratories*, 2001.

[10] K. Hildebrandt, K. Polthier, and M. Wardetzky. Smooth feature lines on surface meshes. In *Symposium on Geometry Processing*, pages 85–90, 2005.

[11] A. Hubeli and M. Gross. Multiresolution feature extraction for unstructured meshes. In *IEEE Visualization*, pages 287–294, 2001.

[12] P. Jenke, M. Wand, M. Bokeloh, A. Schilling, and W. Strasser. Bayesian point cloud reconstruction. *Computer Graphics Forum*, 25(3):379–388, 2006.

[13] I.-K. Lee. Curve reconstruction from unorganized points. *Computer Aided Geometric Design*, 17(2):161–177, 2000.

[14] D. Levin. *Geometric Modeling for Scientific Visualization*, chapter Mesh-independent surface interpolation, pages 37–49. Springer-Verlag, 2003.

[15] T. Ochotta and D. Saupe. Compression of point-based 3D models by shape-adaptive wavelet coding of multi-height fields. In *Symposium on Point-Based Graphics*, pages 103–112, 2004.

[16] M. Pauly, R. Keiser, and M. Gross. Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum*, 22(3):281–290, 2003.

[17] C. E. Scheidegger, S. Fleishman, and C. T. Silva. Triangulating point set surfaces with bounded error. In *Symposium on Geometry Processing*, pages 63–72, 2005.

[18] J. Schreiner, C. Scheidegger, S. Fleishman, and C. Silva. Direct (re)meshing for efficient surface processing. *Computer Graphics Forum*, 25:527–536, 2006.

[19] K. Watanabe and A. G. Belyaev. Detection of salient curvature features on polygonal surfaces. *Computer Graphics Forum*, 20(3):385–392, 2001.

[20] H. Yagou, Y. Ohtake, and A. Belyaev. Mesh smoothing via mean and median filtering applied to face normals. In *Geometric Modeling and Processing – Theory and Applications*, pages 124–135, 2002.