Noname manuscript No.
(will be inserted by the editor)

Joel Daniels II · Tilo Ochotta · Linh K. Ha · Cláudio T. Silva

Spline-Based Feature Curves from Point-Sampled Geometry

the date of receipt and acceptance should be inserted later

Abstract Defining sharp features in a 3D model facilitates a better understanding of the surface and aids geometric processing and graphics applications, such as reconstruction, filtering, simplification, reverse engineering, visualization, and non-photo realism. We present a robust method that identifies sharp features in a point-based model by returning a set of smooth spline curves aligned along the edges. Our feature extraction leverages the concepts of Robust Moving Least Squares to locally project points to potential features. The algorithm processes these points to construct arc length parameterized spline curves fit using an iterative refinement method, aligning smooth and continuous curves through the feature points. We demonstrate the benefits of our method with three applications: surface segmentation, surface meshing and point-based compression.

Keywords Feature extraction · Moving least squares · Point-based modeling · Robust statistics · B-splines

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, where the G^1 continuity is not maintained. Identification of these edges facilitates a better understanding of the model improving filtering [7, 26], reconstruction [22], resampling [4], simplification [9, 12], smoothing [8, 15] 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. The pipeline, illustrated in Figure 1, is an extension of our earlier feature extraction method for point-based models described in Daniels et al. [5]. However, algorithmic modifications are proposed to improve accuracy and reduce noise within the extracted curves near acute feature edges and on low quality point clouds. Furthermore, the altered approach uses an iterative least square fitting of B-spline curves to perform smoothing in place of the previous principle component analysis (PCA) based relaxation.

Our algorithm is based upon the framework of *Robust Moving Least Squares* (RMLS) surfaces [8]. A number of potential feature points are selected, identified by the RMLS operator to be near to possible features. One issue with the RMLS is the inability to reconstruct smooth

Scientific Computing and Imaging Institute
University of Utah
72 S Central Campus Drive, 3750 WEB
Salt Lake City, UT 84112 USA
Tel.: +1-801-585-1867
Fax: +1-801-585-6513
E-mail: {jdaniels ochotta lha csilva}@sci.utah.edu

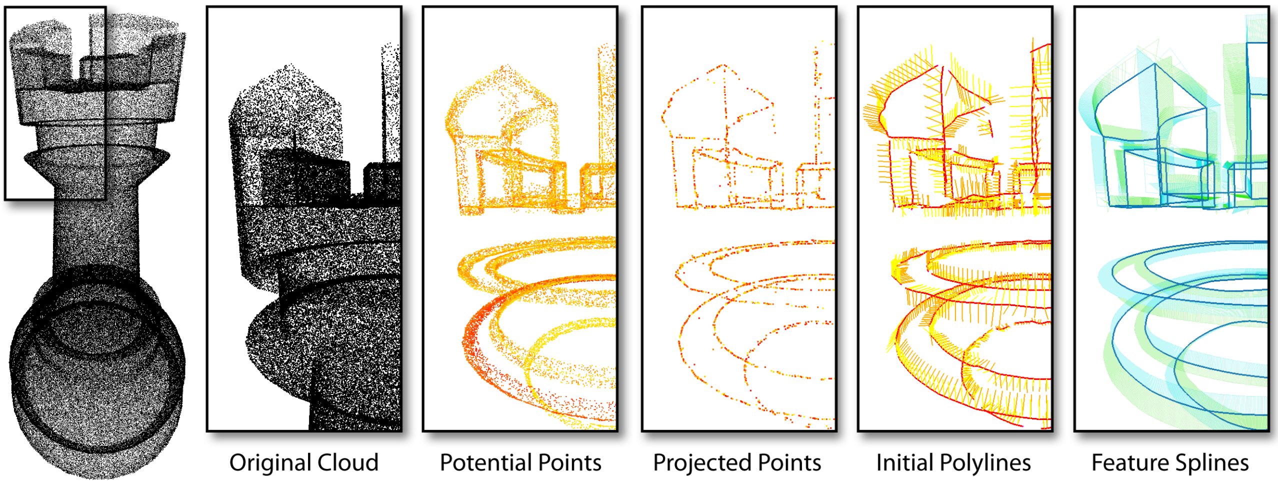

Fig. 1 The feature extraction pipeline (left to right) for a given point cloud, projects candidate points, identified near potential features, to the intersection of locally fit RMLS surfaces. Approximating polylines are grown through the projected data and guide the creation of arc length parameterized B-spline feature curves.

features. The projection method produces jagged edges that cannot be used without further processing to construct smooth feature curves. Therefore, a feature growing strategy constructs polylines used to parameterize and guide the creation of spline curves smoothly fit to the RMLS projected edge points. Our algorithm accounts for poor sampling quality, due to noise or low point densities, by connecting the created polylines across gaps to construct a set of complete feature splines.

The advantage of our approach is that, in contrast to other feature extraction work [11], our input data consists of a set of unorganized points (possibly affected by noise) that approximate the original surface. We assume that no attribute information, e.g., normal vectors, are given a priori. In the presence of closed loops, we are able to segment the point-based model into multiple surface regions that are enclosed by the feature curves. Additionally, our segmentation scheme improves the performance of previously developed point-based algorithms, including surface meshing and compression.

In Section 2, we review related research in feature identification. Section 3 describes our feature extraction pipeline. In Section 4, we apply our results to improve surface segmentation, mesh reconstruction and the point-based compression. Section 5 analyzes experimental results, and lastly, final conclusions are presented 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. [13] 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. [24] use discrete differential geometry operators to compute approximations of the mean and Gaussian curvatures. Whereas, Hildebrandt et al. [11] use anisotropic filtering on discrete differential geometric approximations of third order derivatives of the surface mesh. Both techniques [11, 24] 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. [21] 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. [10] 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 [10, 21] connect the flagged points using a minimum spanning tree and fit curves to approximate sharp edges. Demarsin et al. [6] compute point normals using principal component analysis (PCA) 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 point cloud features by connecting existing points; thus, their accuracy depends on the sampling quality of the input model. In contrast, our method relies on projecting points onto sharp edges, yielding more accurate approximations of features in the original surface.

Jenke et al. [14] 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, [14] performs an optimized smoothing on the geometry. A related technique is proposed by Lipman et al. [18], who use a data-dependent MLS technique to segment the neighborhoods based on regularity detectors.

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 compute $\mathcal{S}$ 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, originally proposed by Levin [17] and more recently modified by [1–3, 8]. 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

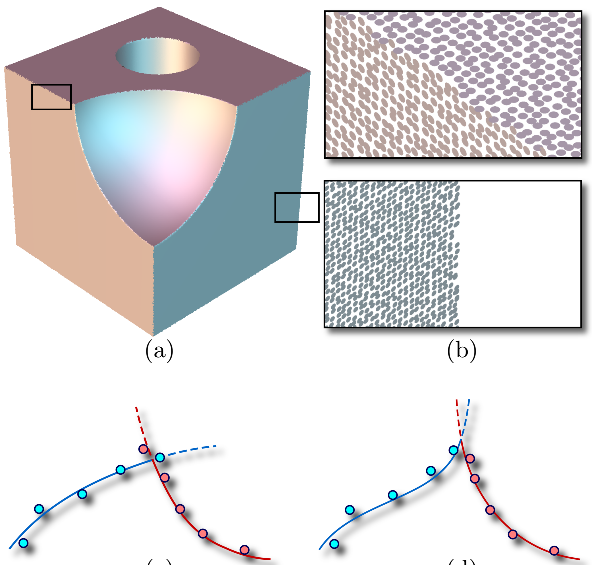

Fig. 2 RMLS sub-neighborhoods are computed independently for each point, which creates jagged edges, (a) and (b), due to the fact that nearby points may construct different neighborhoods; this leads to a disagreement on the configuration of the intersecting surfaces, (c) and (d).

weighted squared distances of points in $\mathcal{P}$ to the plane H, is minimized:

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

where $\theta(\cdot)$ is an 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

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

where p_g denotes the projection of p onto the polynomial g in direction of the plane normal n, and $\theta(\cdot)$ is the previously defined weighting function.

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 [8] 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

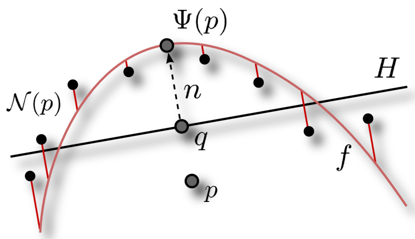

Fig. 3 The residual $r(p)$ is computed as the maximum distance between the points in the neighborhood around p and the polynomial fit f, whereas the distances are considered in direction of the plane normal n.

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 independently for all points, which may lead to situations where nearby points do not grow to identical sub-neighborhoods, resulting in jagged edges. Figure 2a and 2b illustrate the feature edge noise due to RMLS projections. Figure 2c and 2d show two different sub-neighborhood segmentations for one point set that leads to different configurations of the feature region.

To overcome this problem, we propose to construct feature curves that *accurately* approximate the RMLS projected points. Our goal is to extract continuous and complete spline curves that smoothly annotate features on the model despite RMLS and sample related noises. The following is the five-step algorithm:

1. (Section 3.2) Extract points from $\mathcal{P}$ that are near potential features,
2. (Section 3.3) Use RMLS to fit multiple surfaces to the neighborhoods of these points and project each point to its nearest intersection between the surfaces,
3. (Section 3.4) Generate an initial set of feature polylines that approximate the projected points,
4. (Section 3.5) Reconstruct corners and complete gaps in the initial polyline approximations,
5. (Section 3.6) Define an arc-length parameterization over the projected points based on the initial feature polylines to which spline curves can be fit.

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

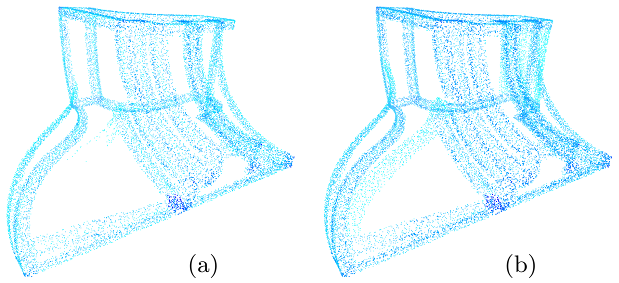

Fig. 4 Identification of potential feature points for the Fandisk model; the threshold τ is used to adjust the number of points retained near potential features; the automatically computed threshold $\tau = 0.057$ (a) and a manually adjusted threshold $\tau = 0.014$ (b).

3.2 Identifying Potential Edge Regions

To identify potential edge regions, we adopt the method of Fleishman et al. [8]. 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, 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 Figure 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 (discussed in Section 3.3).

The identification of potential feature points relies on an appropriate selection of the threshold τ. We extend the pipeline in [8] 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 good 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, generated a projected feature edge cloud $\mathcal{E}$. The output of the procedure is the definition of several surfaces fit to the points in the neighborhood around p. The number of surfaces 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 or more surface indicate the presence of a sharp feature. We now use Newton's method, as in [21], to project p onto the intersection of the two surfaces closest to p, yielding an *edge point* e that may be inserted into the projected edge cloud $\mathcal{E}$.

If the distance between the original point p and its candidate projection point e, $\|p - e\|$, is less than the radius of the RMLS neighborhood, then e is added to $\mathcal{E}$. The culling process reduces noise within the set of projected points $\mathcal{E}$, especially near acute features or on noisy models. The purpose is to keep only the projected points whose feature is well defined by the neighborhood of the original point.

However, in practice we found that regions of $\mathcal{E}$ have low sample densities due to the culling of projected points. This becomes problematic during the feature approximation stage, subsection 3.4, by creating large gaps between extracted polylines. To accommodate for the lost points, we visit the projected points $e \in \mathcal{E}$ and upsample their neighborhoods. The upsampling process adds points along the eigenvector v_0 that corresponds to the largest eigenvalue of the neighborhood of e, $\mathcal{N} \in \mathcal{E}$, $e \pm \alpha \cdot v_0$. The scalar value α is based on the growth step size defined in Section 3.4. Upsampled points are only computed if the computed eigenvectors v_0, v_1 and v_2, with associated eigenvalues $\lambda_0 \geq \lambda_1 \geq \lambda_2$, are highly correlated, $\lambda_0/(\lambda_0 + \lambda_1 + \lambda_2) > \beta$. This computation is adapted from the surface variation computation used by [20] to analyze the variation of a curve. The models and results presented in this paper experienced good results with $\alpha = 0.75$ and $\beta = 0.7$.

3.4 Feature Polyline Propagation

At this stage of the pipeline, the set of feature points $\mathcal{E}$ 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 with a set of polylines. Because these polylines will guide the future parameterization of $\mathcal{E}$ and iterative spline curve fitting, they are not expected to be smooth; instead, we rely on the spline curves to smooth noise in the point set $\mathcal{E}$.

We build a polyline construction algorithm based on Lee's method [16], which works as follows. The user defines the maximum allowable segment length s_{max} as an input parameter to control the coarseness of the feature polylines. The feature polyline is initialized at a random seed point p, where the PCA for points in the neighborhood $\mathcal{N}(p) \subset \mathcal{E}$ with radius s_{max} is computed, Figure 5(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. Lee's growing scheme attaches 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.

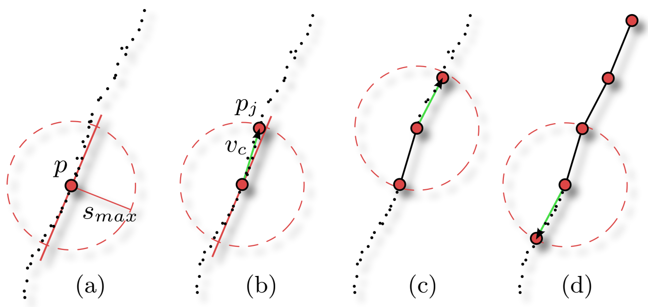

Fig. 5 Curve growing starts at a seed point, finds all points in a user specified radius, projects them to the major eigenvector, computes a weighted average step direction, and adds a new point to the approximating feature polyline. The processed points are removed and the method is repeated.

However, to grow smoother approximating polylines, we opt to compute a current step vector v_c that is a weighted average of the neighborhood points:

$$v_c = \sum_{i=0} \theta(\|p'_i - p\|) \cdot \frac{p_i - p}{\|p_i - p\|}.$$

The points are weighted inversely based on their projected distance along the eigenvector, $\theta(s) = e^{-s^2/s^2_{max}}$, and the final vector v_c is normalized. A new polyline vertex is added to the feature at $p_j = p + s_{max} \cdot v_c$, illustrated in Figure 5(b). The procedure is repeated at the p_j, Figure 5(c), until no points remain in the growth direction. The process is re-seeded at p and repeated in the opposite growth direction 5(d).

To prevent rounding corner regions by growing onto multiple feature edges and to reduce the noise in the approximating feature polylines, an additional termination condition is required. A growth vector v_g stores an average direction based on previous step vectors. To commit a new point p_j, the angle between the current step vector v_c and the growth vector v_g must be below a defined threshold, $\langle v_g, v_c \rangle \leq cos(\alpha)$. For our purposes, $\alpha = 30°$ provided an adequate angle threshold. After the point is committed to the feature polyline, the growth vector v_g is updated, accumulating the step vectors $v_g = (v_g + v_c)/2$.

For our models we found that best results are achieved by starting the feature growing procedure at edge points with highly correlated neighborhood eigenvectors. Our implementation maintains a priority queue that sorts all edge points $p \in \mathcal{E}$ according to their eigenvector correlation. Given eigenvalues $\lambda_0 \geq \lambda_1 \geq \lambda_2$, the correlation term for a point is computed as $\lambda_0/(\lambda_0+\lambda_1+\lambda_2)$. This sorting term seeds the feature growth at points furthest away from corner regions in $\mathcal{E}$ where the correlation terms become smaller.

To create a feature line, we remove the first element from the queue and apply 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. As long as the queue is not empty, there are unvisited feature points remaining thus continuing feature line creation.

The polyline growing scheme approximates the projected cloud $\mathcal{E}$; however, 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.

3.5 Completing Feature Curves

In this stage, the feature lines are merged to produce complete approximating polylines through the RMLS projected cloud $\mathcal{E}$. By inspecting the end points of each feature, the completion stage inspects gaps between multiple polylines and regions where new corners should be constructed. Connecting the polylines is important as these lines will guide the construction of the final feature spline curves.

Our feature completion method is driven by a directional search approach. As illustrated in Figure 6, we consider the tangent vector that is evaluated for an end point by fitting a cubic polynomial to the last four points at each end of the feature polyline. An alternative approach evaluates the direction of the last segment of the curve; however, we found that a cubic fit smooths perturbations for 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 Figure 6: *gap completion* (a), *corner creation* (b), and *edge splitting* (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 repeating the feature completion by iteratively increasing α over the range $[0, 1.5]$ accommodates for the projection based noise.

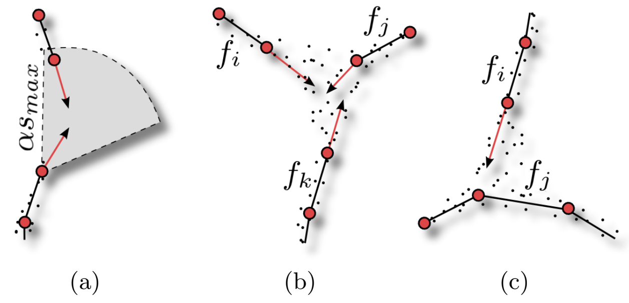

Fig. 6 Feature completion analyzes the end points of each feature curve to complete gaps between multiple features (a), merge multiple polylines into a corner (b), or split a single feature into two thus forming a new corner point (c).

Figure 6(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 end points that are within αs_{max} distance of each other and have corresponding tangent vectors that point to opposing directions within the aperture angle.

A corner is created, Figure 6(b), when multiple feature polylines are detected within the search cone. For three feature lines $f_{i,j,k}$ with end points $p_{i,j,k}$ within the search cone and converging tangents, a corner point is evaluated using the RMLS surfaces fit to the neighborhood of the midpoint, $(p_i+p_j+p_k)/3$. Similar to the edge projection procedure, Newton's method is used to evaluate the intersection of the three RMLS defined surfaces. Then the features $f_{i,j,k}$ are connected to the computed corner.

Edge splitting and corner creation occurs if the end point of a feature polyline f_i projects to an interior point p of another feature line f_j, Figure 6(c). The feature f_j is split into two feature polylines, f_k and f_l, at p and a corner point c is identified within the region using Newton's method on the RMLS defined surfaces. The three feature polylines f_i, f_k, and f_l are joined at the corner point c. This case resolves situations where a single polyline grew onto multiple features despite our threshold tests.

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 end points have been resolved the feature curves are considered complete. Note that the feature polylines are often very coarse, with many perturbations. In the following section we discuss how the feature polylines are used to estimate an appropriate arc-length parameterization for the feature projected curves. In this manner, spline-based fitting and refinement will smooth the final feature curves.

3.6 Spline-based Fitting and Smoothing

The output of the feature extraction algorithm is a set of continuous spline curves fit to $\mathcal{E}$ guided by the approximating feature polylines. To this point we have defined a set of feature polylines $\mathcal{F} = \{f_i\}$ that approximate RMLS feature edges through a projected point cloud $\mathcal{E}$. The spline fitting procedure segments $\mathcal{E}$ based on proximity of feature polylines, assigns parameter values to the segmented points, then uses a iterative least squares fitting procedure to define the spline curves.

The segmentation of $\mathcal{E}$ into multiple sub-clouds is based on the user defined neighborhood size s_{max}, used earlier as a step size for the polyline growth. A point $e \in \mathcal{E}$ is projected to each feature polyline f_i, yielding the point e'_i. The point e is inserted into the sub-cloud $\tilde{\mathcal{E}}_i$ if:

1. the distance between e and e'_i is less than the specified distance, $\|e - e'_i\| < s_{max}$,
2. e'_i is the nearest projected point to e, $\forall j, \|e'_j - e\| > \|e'_i - e\|$.

Next, the points within each segmented sub-cloud $\tilde{\mathcal{E}}_i$ are assigned parameter values in order to fit spline feature curves. A point $e \in \tilde{\mathcal{E}}_i$ is assigned a parameter value based on the ratio of the arc-length distance of its projection on f_i, e'_i, along the feature polyline and the overall length of f_i. In this manner, all computed parameter values are bounded between $[0, 1]$.

The spline curves are created for each segmented sub-cloud using an iterative approach that refines the defining knot vector until the curve is within bounded distance tolerances. For a sub-cloud $\tilde{\mathcal{E}}_i$ with associated parameter values $\mathcal{U}_i$, a spline curve $\mathcal{S}$ with the knot vector $k = \{k_i\}_{i=0}^{N}$ is defined using a least squares fit. The fitting procedure is initialized with the minimal uniform closed knot vector describing a cubic B-spline curve, $k = \{0, 0, 0, 0, 1, 1, 1, 1\}$.

Iterative refinement of the final curve is determined by comparing measured distances between $\tilde{\mathcal{E}}_i$ and $\mathcal{S}$. The refined knot vector k_{new} is initially equivalent to k. Distance comparisons are made by evaluating the two points p_e and p_s at the parameter values between each consecutive non-equal pair of knot values, $u = (k_i + k_{i+1})/2$ on $\tilde{\mathcal{E}}_i$ and $\mathcal{S}$ respectively. While evaluation of p_s is well defined through B-spline evaluation, we further describe the approximation of $\tilde{\mathcal{E}}_i(u)$ as a weighted average of the sub-cloud points,

$$p_e = \frac{\sum_j \theta(|u_{ij} - u|) \cdot e_{ij}}{\sum_j \theta(|u_{ij} - u|)},$$

where, the weighting function is $\theta(x) = e^{-x^2/\alpha^2}$ and α is a user defined fall-off term. If the distance between p_e and p_s is greater than a defined tolerance, $\|p_e - p_s\| \geq s_{max}$ (the step size previously used for the polyline growth), then a new knot u is inserted into the knot vector k_{new}.

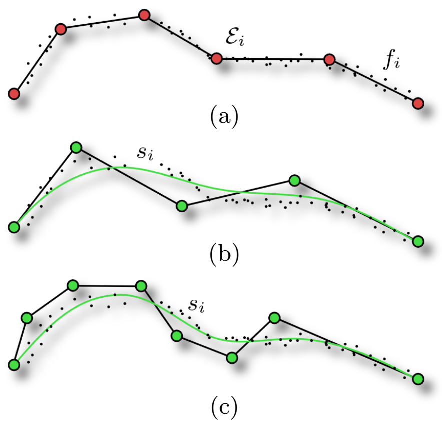

Fig. 7 The feature polyline f_i guides the construction of the spline curve fit to the edge projected sub-cloud $\mathcal{E}_i$ (a). An initial spline curve is fit to the arc-length parameterized cloud (b), and iteratively refined until the curve describes the points within a distance threshold (c).

A new spline curve is fit with the knot vector k_{new}, thus refining the feature curve definition. The distance comparisons are iteratively repeated until no new parameters are inserted into the knot vector k_{new}. The refinement of the spline curve converges towards a locally weighted average of the RMLS projected point cloud $\mathcal{E}$; which, in practice accurately annotates the feature edges.

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 segmentation, reconstruction and shape compression.

We demonstrate the effectiveness of our feature extraction technique using point models that represent machine parts with sharp edges where feature extraction can be well-defined. We emphasize that all data sets consist only of a set of 3D points, and do not assume associated normal vectors for the geometry.

4.1 Surface Segmentation

Surface segmentation algorithms dissect a model into multiple regions that each describe a common component section. While several techniques exist, specifically for point-based models [25, 27], our approach differs in that we do not compute surface approximations, normals, or other differential properties. By leveraging extracted edge curves, this segmentation algorithm defines multiple feature aligned surface regions by comparing only Euclidean distances between neighbor points.

Our point-based segmentation method operates directly on 3D point data and does not require surface normals nor connectivity. From an input cloud $\mathcal{P}$, the algorithm begins by flagging all points $p \in \mathcal{P}$ within a specified distance r of a feature polyline. Consequently, the points are classified as *boundary* points and *unvisited* points. The algorithm operates in two steps: first, segmenting the unvisited points into multiple regions, second merging boundary points into the nearest point group.

The segmentation is seeded at a random unvisited point $p \in \mathcal{P}$ then grown using an advancing wavefront approach. A priority queue $\mathcal{W}$, representing the wavefront, is initialized with the neighborhood points of p within a distance r, sorted by their distance from p. The growth of the region occurs by processing the top point of $\mathcal{W}$, p_t, until no more points remain within the wavefront.

If the point p_t, with a priority distance d, is classified as unvisited, then we further propagate the wavefront. The unvisited neighbors of p_t are inserted into $\mathcal{W}$; where, the distance between p_t and a neighbor p_n is summed with p_t's priority distance d, $\|p_n - p_t\| + d$, to provide the sorting metric within $\mathcal{W}$. The top point p_t is inserted into the growing region, and classified as visited. Additionally, references are saved to p_t's neighborhood points that are flagged as boundary points to improve the boundary resolution performance, later described. When the wavefront $\mathcal{W}$ is exhausted, no new unvisited neighbors exist for a segmenting region, the process is repeated by seeding at a new unvisited point until the entire cloud is visited (or a boundary point).

After the unvisited points in $\mathcal{P}$ have been segmented into multiple regions, we resolve the boundary points. A boundary point is inserted into the region that contains the closest visited point. Note that these distance computations are saved from computations performed during the previously described growth phase.

Segmented models, using the described algorithm, are illustrated in Figure 8 with the splats oriented based on MLS projected normals for each of the segmented regions. The MLS projections no longer smooths the sharp edges, instead reconstructs the sharp feature normals. We further motivate our feature extraction and surface segmentation results in the following subsections with surface reconstruction and compression applications.

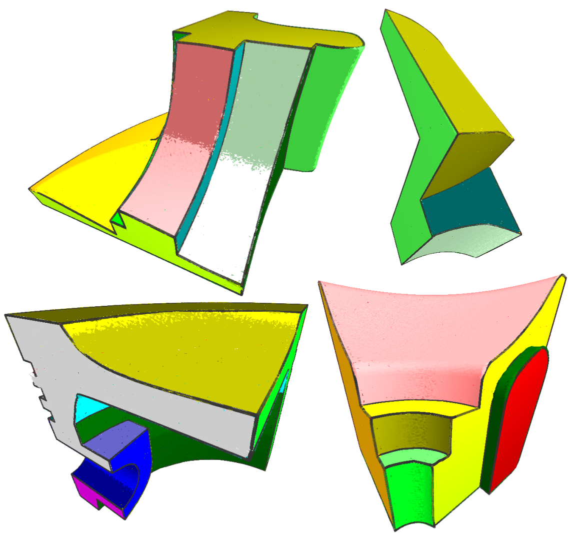

Fig. 8 The segmentation algorithm dissects the cloud into multiple feature aligned regions, where the MLS projected splats for each region reconstruct sharp feature normals on the model.

4.2 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 the development of surface construction and remeshing algorithms.

Schreiner et al. [23] extend the approach in [22] 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, 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

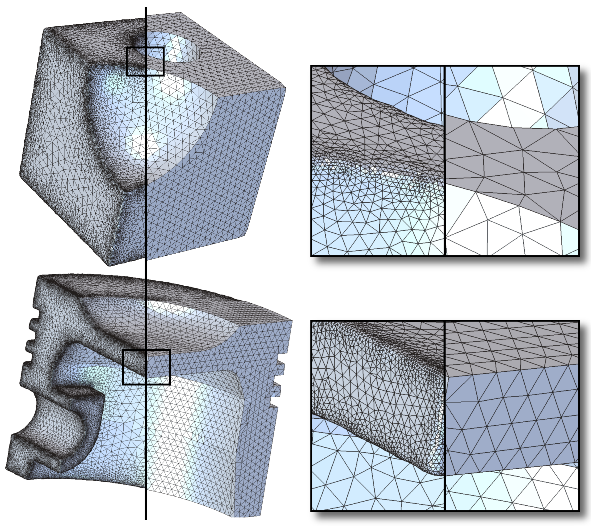
Fig. 9 Results obtained by applying the meshing technique in [23] to Scalloped Cube (top) and Quarter Piston (bottom); the left half of each closeup view shows meshes obtained by the traditional method; the corresponding right halves show the meshes after initializing the input of [23] with feature lines extracted with our method.

lines as the initial 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, Figure 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 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. Using feature polylines, the reconstructed mesh has a significantly reduced triangle count, Figure 9.

4.3 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 re-obtain the original model or an approximation of it, if the encoding is lossy.

In [19] 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.

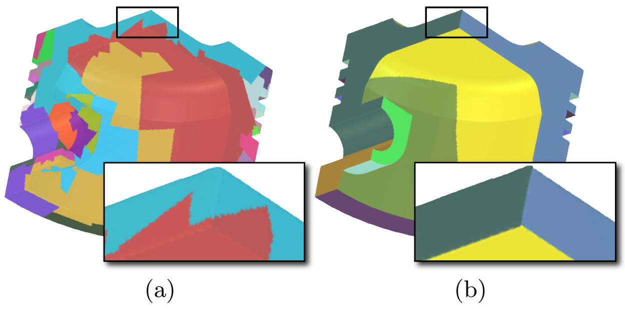
Fig. 10 The partition of Quarter Piston using the traditional patch composition method in [19] (a); partition with initial segmentation (b).

Although their method yields a high rate-distortion performance, the approximation quality largely depends on partition structure. Especially for models with sharp features, the constructed partitions show patches that contain the features, rather than identifying the features as patch boundaries. The 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 [19] 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 3.5. 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, Figure 10(b). 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, Figure 10(a).

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

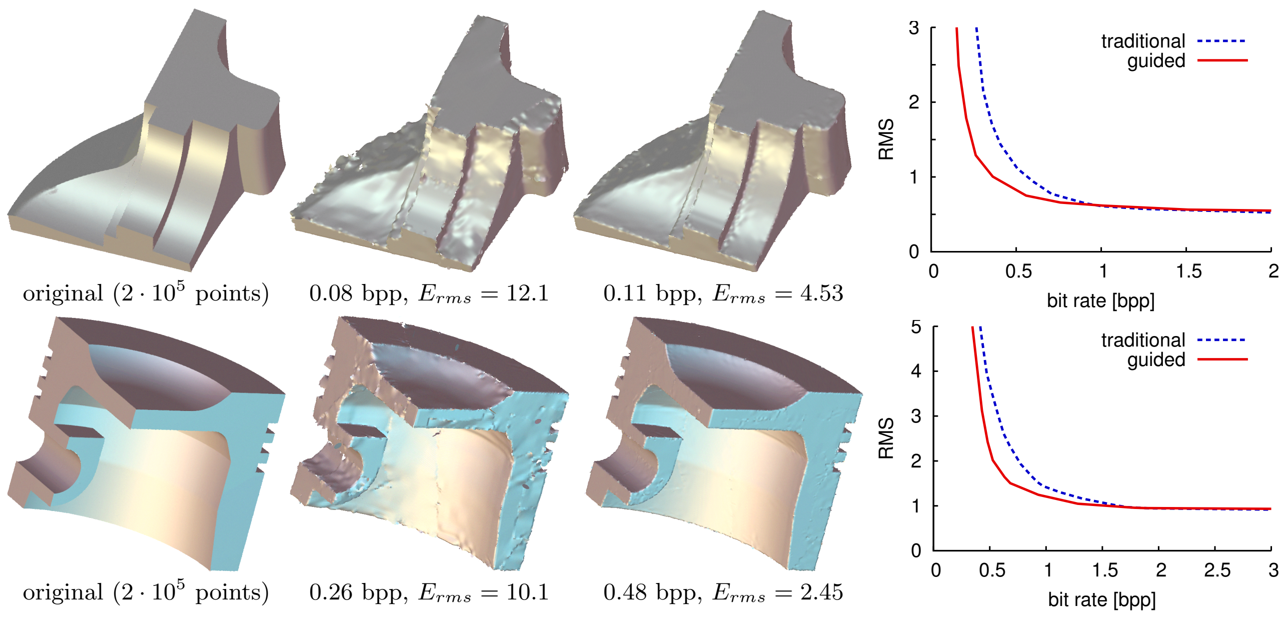

Fig. 11 Rate distortion performance for Fandisk (top row) and Quarter Piston (bottom row); the left figure shows the original model, followed by two decodings for different rates; the rate-distortion curves show that the compression performance is significantly increased by applying the partitioning in [19] based on feature aware segmentation in this work (right); the error is the root-mean-square error in [19].

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 (Figure 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 piston and chess piece models 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.

For low levels of uniform noise, the feature extraction procedure successfully defines smooth feature curves, as illustrated in Figure 12. Despite introducing a $0.02d_B$ uniform noise on the two models, the algorithm extracts a majority of feature curves within error tolerances of the actual sharp edges, tested against a hand built feature model extracted from a triangular mesh. Under levels of high noise, $0.05d_B$, differentiation between the uniform noise and feature edges becomes difficult. Consequently, weak features, with near planar defining surfaces, are not clearly extracted by the algorithm.

Further accuracy measurements are presented in Table 1 for the models presented throughout this paper. Ground truth feature edges are constructed from mesh models used to generate each of the point clouds in this paper. Our analysis computes the average and max distance by projecting the extracted feature splines onto the nearest ground truth feature curve, and reports their distances in terms of the length of each model's bounding box diagonal. On average, the distances between the extracted features and actual features is very small ($\leq 0.5\%$ of the bounding box diagonal d_B). Additionally, Table 1 reports the percent of the extracted feature curves; showing that the feature extraction algorithm annotates a majority of the sharp edges on the point-based model. This value is computed by measuring the percentage of ground truth feature curve segments with an extracted feature within distance $0.01d_B$ (1% of the bounding box diagonal).

The total time required to construct the polylines is determined by the number of *potential* feature points in data set, since only the feature point identification stage operates on the complete point set. Table 1 shows run times for the different stages in our algorithm as run against several models, carried out on a Pentium4 2 GHz platform. Our implementation is able to process models of half a million points within a few minutes, where the major part of computational costs is required for the RMLS projection (stages 2).

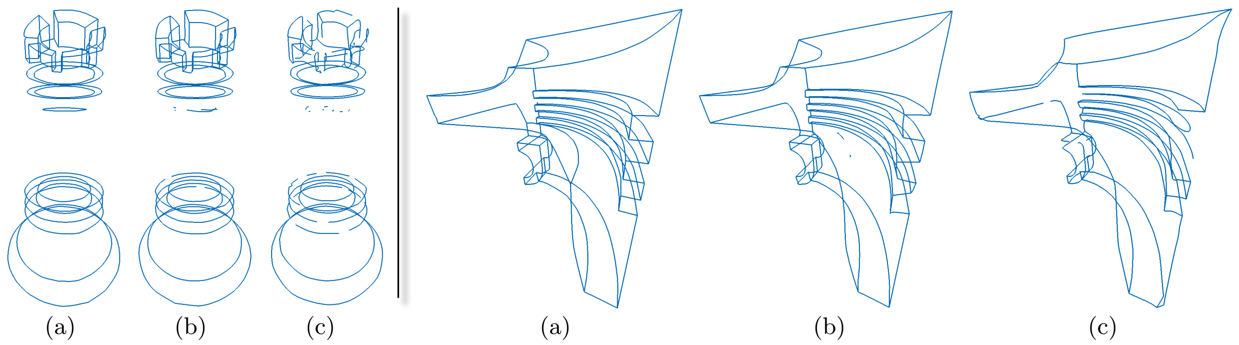

Fig. 12 Feature extraction from the chess piece and piston models; features extracted from the original model (a); features extracted from the original model after applying uniform noise $\|\delta(p)\| \leq 0.02d_B$ (b), and $\|\delta(p)\| \leq 0.05d_B$ (c), where d_B is the length of the model bounding box diagonal.

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 spline-based curves fit to the RMLS feature points smooths the perturbations and noise of the projections and extracts an accurate representation of the sharp features. Additionally, note that the extracted feature splines are coupled with normal vectors. We couple the projected points with the RMLS evaluated normals during the feature projection phase. The feature growing method groups normals of neighboring polyline vertices to establish consistent vector orientation, which is used to reorient the normals in the feature projected cloud. While fitting splines to the feature edge points, we also fit vector splines to the point normals, visualized as offset curves in Figure 13.

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 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 neighborhood selection methods to speed up MLS methods capable of detecting and reconstructing sharp features, 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 was partially supported by a Vietnam Education Foundation fellowship. Special thanks to John Schreiner for his cooperation and assistance with the mesh reconstructions, Shachar Fleishman for comments and suggestions on our work, and Jason Shepherd for providing several of the models used throughout this paper.

References

1. Alexa, M., Adamson, A.: On normals and projection operators for surfaces defined by point sets. In: Proc. Symposium on Point-Based Graphics, pp. 149–155 (2004)
2. Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., Silva, C.T.: Computing and rendering point set surfaces. IEEE Transactions on Visualization and Computer Graphics **9**(1), 3–15 (2003)
3. Amenta, N., Kil, Y.J.: Defining point-set surfaces. ACM Transactions on Graphics **23**(3), 264–270 (2004)
4. Attene, M., Falcidieno, B., Rossignac, J., Spagnuolo, M.: 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 (2005)
5. Daniels, J., Ha, L., Ochotta, T., Silva, C.: Robust smooth feature extraction from point clouds. In: IEEE International Conference on Shape Modeling and Applications, pp. 123–136 (2007)
6. Demarsin, K., Vanderstraeten, D., Volodine, T., Roose, D.: Detection of closed sharp feature lines in point clouds for reverse engineering applications. Tech. Rep. TW 458, Department of Computer Science, K.U.Leuven, Belgium (2006)

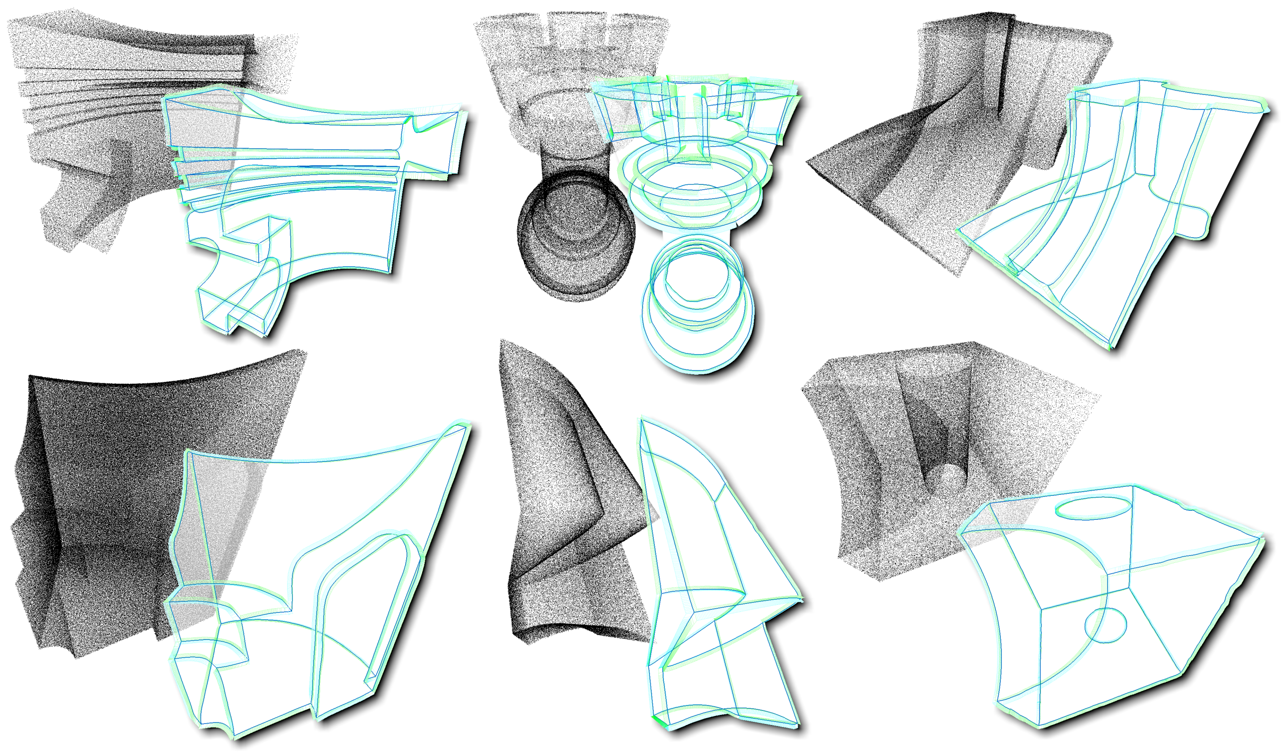

Fig. 13 Features extracted with our method; displaying the original point sets without normal information does not show surface characteristics; our method identifies sharp features and can define oriented surface patches without relying on input normal vectors.

Model	$\|\delta(p)\|$	τ $[10^{-2}]$	$\|\mathcal{F}\|$	Run Time [seconds]					Analysis		
				(1)	(2)	(3)	(4)	(5)	Mean Dist.	Max Dist.	% Extracted
Fandisk	0	1.3	55.7k	191	192	11.5	0.8	0.0	$0.0013d_B$	$0.0089d_B$	99%
Hook	0	0.6	49.6k	207	196	0.7	0.6	2.3	$0.0005d_B$	$0.0049d_B$	100%
Piston	0	0.6	80.3k	146	207	0.3	0.7	9.9	$0.012d_B$	$0.19d_B$	99%
	$\leq 0.02d_B$	1.7	97.9k	196	698	1.3	0.7	68	$0.0044d_B$	$0.25d_B$	97%
	$\leq 0.04d_B$	3.0	82.2k	169	552	18.9	1.9	29	$0.0049d_B$	$0.18d_B$	90%
Ra	0	1.2	40.8k	159	116	1.0	0.0	1.6	$0.0013d_B$	$0.025d_B$	99%
Rook	0	1.1	99.6k	216	471	5.8	3.0	8.8	$0.0022d_B$	$0.046d_B$	99%
	$\leq 0.02d_B$	3.0	112k	250	840	21.7	12	19	$0.0025d_B$	$0.055d_B$	97%
	$\leq 0.04d_B$	3.4	124k	296	1337	19.0	18	23	$0.0042d_B$	$0.057d_B$	92%
S-Cube	0	1.75	33.6k	186	324	2.5	0.3	5.6	$0.0025d_B$	$0.025d_B$	100%
Knuckle	0	0.95	70.1k	176	348	10.3	1.3	7.5	$0.0010d_B$	$0.026d_B$	98%

Table 1 Run time performance of our feature extraction implementation for models, illustrated in Figure 13, that contain 500000 points each; the stages in our pipeline correspond to point identification (1), point projection (2), curve propagation (3), curve completion (4), and spline fitting (5). The analysis includes the mean and max distance from the extracted features to a ground truth, as well as the percent of actual features identified.

7. Desbrun, M., Meyer, M., Schröder, P., Barr, A.H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proc. ACM SIGGRAPH, pp. 317–324 (1999)
8. Fleishman, S., Cohen-Or, D., Silva, C.T.: Robust moving least-squares fitting with sharp features. ACM Transactions on Graphics **24**(3), 544–552 (2005)
9. Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: Proc. ACM SIGGRAPH, pp. 209–216 (1997)
10. Gumhold, S., Wang, X., McLeod, R.: Feature extraction from point clouds. In: Proc. 10th International Meshing Roundtable, Sandia National Laboratories (2001)
11. Hildebrandt, K., Polthier, K., Wardetzky, M.: Smooth feature lines on surface meshes. In: Proc. Symposium on Geometry Processing, pp. 85–90 (2005)
12. Hoppe, H.: New quadric metric for simplifying meshes with appearance attributes. In: Proc. IEEE Visualization, pp. 59–66 (1999)
13. Hubeli, A., Gross, M.: Multiresolution feature extraction for unstructured meshes. In: Proc. IEEE Visualization, pp. 287–294 (2001)
14. Jenke, P., Wand, M., Bokeloh, M., Schilling, A., Strasser, W.: Bayesian point cloud reconstruction. Computer Graphics Forum **25**(3), 379–388 (2006)
15. Jones, T.R., Durand, F., Desbrun, M.: Non-iterative, feature-preserving mesh smoothing. ACM Transactions

on Graphics **22**(3), 943–949 (2003)
16. Lee, I.K.: Curve reconstruction from unorganized points. Computer Aided Geometric Design **17**(2), 161–177 (2000)
17. Levin, D.: Geometric Modeling for Scientific Visualization, chap. Mesh-independent surface interpolation, pp. 37–49. Springer-Verlag (2003)
18. Lipman, Y., Cohen-Or, D., Levin, D.: Data-dependent MLS for faithful surface approximation. In: Symposium on Geometry Processing 2007 (2007)
19. Ochotta, T., Saupe, D.: Compression of point-based 3D models by shape-adaptive wavelet coding of multi-height fields. In: Proc. Symposium on Point-Based Graphics, pp. 103–112 (2004)
20. Pauly, M., Gross, M., Kobbelt, L.P.: Efficient simplification of point-sampled surfaces. In: Proc. IEEE Visualization, pp. 163–170 (2002)
21. Pauly, M., Keiser, R., Gross, M.: Multi-scale feature extraction on point-sampled surfaces. Computer Graphics Forum **22**(3), 281–290 (2003)
22. Scheidegger, C.E., Fleishman, S., Silva, C.T.: Triangulating point set surfaces with bounded error. In: Proc. Symposium on Geometry Processing, pp. 63–72 (2005)
23. Schreiner, J., Scheidegger, C., Fleishman, S., Silva, C.: Direct (re)meshing for efficient surface processing. Computer Graphics Forum **25**, 527–536 (2006)
24. Watanabe, K., Belyaev, A.G.: Detection of salient curvature features on polygonal surfaces. Computer Graphics Forum **20**(3), 385–392 (2001)
25. Woo, H., Kang, E., Wang, S., Lee, K.H.: A new segmentation method for point cloud data. International Journal of Machine Tools and Manufacture **42**, 167–178 (2002)
26. Yagou, H., Ohtake, Y., Belyaev, A.: Mesh smoothing via mean and median filtering applied to face normals. In: Proc. Geometric Modeling and Processing – Theory and Applications, pp. 124–135 (2002)
27. Yang, M., Lee, E.: Segmentation of measured point data using a parameteric quadric surface approximation. Computer-Aided Design **31**(7), 449–457 (1999)

Joel Daniels received a B.S. in Computer Science from the University of New Hampshire in 2003, graduating summa cum laude, and a M.S. in Computer Science from the University of Utah in 2005. He is currently working toward a PhD degree in the School of Computing at the University of Utah as a research assistant for the Geometric Design and Computation group headed by Elaine Cohen. His primary research interests include geometric modeling challenges, especially addressing the differences between discrete and continuous representations.

Tilo Ochotta received a Diploma degree in Computer Science from the University of Leipzig, Germany in 2002 and a Ph.D. degree in Computer Science from the University of Konstanz, Germany in 2007. He is now a Postdoctoral Research Associate at the Scientific Computing and Imaging (SCI) Institute at the University of Utah in Salt Lake City. His primary research interests include topics in computer graphics and geometric processing with emphasis on point-based modeling, shape representation, and compression. He is also interested in large data set processing in numerical weather prediction.

Linh K. Ha received B.E in Electronics and Telecommunication from University of Technology, Hanoi, Vietnam in 2002 and M.S in Electronics and Telecommunication from College of Technology, Vietnam National University in 2005. He is currently working toward PhD degree in Department of Computer Science, School of Computing, University of Utah. He has worked as a research assistant in Visualization and Geometric Computing group, Scientific Computing and Imaging Institute. His primary interests include point-based processing, volume rendering, out of core rendering especially High Parallel Computing. Currently, his research focuses on developing high parallel data flow architecture on GPU.

Claudio T. Silva received the BS degree in mathematics from the Federal University of Ceara, Brazil, in 1990, and the PhD degree in computer science from the State University of New York at Stony Brook in 1996. He is an associate professor of computer science and an associate director of the Scientific Computing and Imaging (SCI) Institute at the University of Utah. Before joining Utah in 2003, he worked in industry (IBM and AT&T), government (Sandia and LLNL), and academia (Stony Brook and OGI). He coauthored more than 100 technical papers and eight U.S. patents, primarily in visualization, geometric computing, and related areas. He is an active member of the visualization, graphics, and geometric computing research communities, having served on more than 50 program committees. He is co-editor of the Visualization Corner of the Computing in Science and Engineering magazine. Previously, he was on the editorial board of the IEEE Transactions on Visualization and Computer Graphics. He was papers co-chair for IEEE Visualization conference in 2005 and 2006. He received IBM Faculty Awards in 2005, 2006, and 2007, and a best paper award at IEEE Visualization 2007. He is a member of the ACM, Eurographics, and IEEE.