Methods for Feature Detection in Point Clouds

Christopher Weber[1], **Stefanie Hahmann**[2], **and Hans Hagen**[1]

1 **Technische Universität Kaiserslautern, Germany**
2 **Laboratoire Jean Kuntzmann, Université de Grenoble, France**

Abstract

This paper gives an overview over several techniques for detection of features, and in particular sharp features, on point-sampled geometry. In addition, a new technique using the Gauss map is shown. Given an unstructured point cloud, this method computes a Gauss map clustering on local neighborhoods in order to discard all points that are unlikely to belong to a sharp feature. A single parameter is used in this stage to control the sensitivity of the feature detection.

1998 ACM Subject Classification Edge and feature detection

Keywords and phrases point cloud, sharp features, reconstruction, Gaussmap, clustering

Digital Object Identifier 10.4230/OASIcs.VLUDS.2010.90

1 Introduction

The use of 3D scanning technologies in art, design, manufacturing and research has become more and more common over the last years. Possible applications range from research and medical applications to prototyping and design. For example during the development of a new product, 3D scanning devices can be used to digitize and validate a manually optimized prototype, thus speeding up the development process. It is of huge importance that no details of a scanned object are lost during the scanning process. But not only the scanning process itself needs to be optimized. Also the reconstruction of the raw point set data delivered by the scanning device needs to be performed properly. During the reconstruction of the surface one does not want to loose specific features of the original object, e.g. sharp edges or corners. Sharp corners and edges are often used as design element, for example in the car manufacturing industry to visually break up huge planes and underline the dynamic and unique design of a car. In such cases it is useful to know the exact positions of sharp features in the raw point cloud data. Other examples for the reasonable use of feature detection might be quality measurement, monitoring of a manufacturing process or medical applications. In non-photorealistic rendering feature lines are used to enhance the visual perception. Additional, in the case of mesh generation, mesh simplification and segmentation knowledge about the position of features can be of great help. Depending on discipline and application, "feature" can have different meanings. In computer graphics and CAD for 3D shapes this term is usually used for *free form features* including all kinds of visually prominent characteristics of a shape, from salient edges, ridge and valley lines to sharp features as line-type or corner features.

This paper aims first to give an overview over prominent existing feature detection methods. These methods are classified into different groups, polygonal based methods in Section 2 and point based methods in Section 3. We then present in Section 4 a new method especially dedicated to the extraction of sharp features on point-sampled surfaces. In our case, a point-sampled surface is a simple unstructured point cloud, where the points belong to a 2-manifold, without any further information about (mesh) connectivity, topology, parameterization or differential properties. Our algorithm is based on the observation that *sharp features* have the property to separate clearly two or more local surface parts with a tangent discontinuity. We thus introduce Gauss map clustering as a sharp feature detection operator. The presented method does not need any prior surface reconstruction. It can be applied directly on the point cloud and only a local neighbor graph need to be computed. The method is therefore fast and works for a wide range of sampling resolutions. It provides a set of sharp feature points as result, even in the presence of noise.

2 Polygonal Methods

There exist multiple techniques for feature extraction relying on polygonal meshes [6, 12, 9, 7, 13]. The following methods represent a group of different approaches. The list of methods is not intended to be exhaustive.

Hubeli and Gross [6] use a normal based multi-resolution framework and generate a set of edges with a normal-based classification operator. In a classification phase they assign a weight to every edge in the input mesh, proportional to the probability of belonging to a feature. The authors provide different types of operators for different mesh types like a "second order difference" operator for very coarse data sets, an "extended second order difference" operator for finer meshes, or a computational more expensive "best for polynomial" operator which performs well on noisy points sets. After this, in a detection phase they reconstruct the features from the information gained in the classification phase. According to the weights, they produce piecewise linear curves from the collections of edges that are assumed to belong to a feature. For thresholding they use a hysteresis thresholding. An edge is added as feature if it's weight is larger than an upper bound. If the weight is smaller but a neighboring edge is already selected as feature, the hysteresis thresholding provides another lower bound and accepts the edge if its weight is above this second threshold. All other edges are discarded as features. A thinning process then refines the edges to generate clear feature lines. For the thinning all patch-boundary edges are first inserted to a linked list. A first condition removes edges that are perpendicular to the mesh feature being extracted. A second condition makes sure that an edge is only removed if the patch will not become disconnected. If an edge is removed, new edges are inserted into the list and have to be analyzed since they became boundary edges. The process continues until the list is empty. They also present a multi resolution approach for their feature extraction to improve the quality. The process is not fully automatic since a user has to choose the classification operator and some parameters for the detection phase.

Hildebrand et al. [7] use anisotropic filtering on third order derivatives of the surface mesh. The derivatives are approximated by discrete differential geometric approximations. This way, the authors compute discrete extremalities, which are then smoothed and used to trace feature lines in regular triangles. Singular triangles need a special treatment. In this case the adjacent triangles are used to determine the feature line intersections with the singular triangle. After the first feature line extraction, a threshold filter is used to improve the stability of the feature extraction and to remove small ridges. The last step is an optional smoothing of the feature lines. Both methods ([6], [7]) use extrema triangles to build a set of sharp feature edges.

Watanabe and Belyaev [12] use the so called focal surfaces to detect curvature extrema on dense triangle meshes. If k_{max} and k_{min} are the largest and smallest principle curvature then the principle centers of curvature are points situated at the surface normal with a distance of $1/k_{max}$ and $1/k_{min}$ from the surface. These principle centers form the focal surface. It consists of two sheets, one for the minimal, and one for the maximal principal curvature. Watanabe now uses the property, that the singularities of the focal surfaces, called focal ribs, correspond to lines on the original surface where the principal curvature has extrema. They present a method for the estimation of the principle curvature on a dense triangle mesh and then show how the associated focal ribs can be used to identify the features. Also here a thinning process of the first results is necessary to construct a final feature line. The resulting lines show the regions of maximal curvature. The method is not specialized for the detection of sharp features.

All mesh-based techniques use the connectivity information and normals associated with the underlying mesh. But often surface scanning devices do not deliver a mesh as raw data, but an unsorted set of point data representing the original surface. In this case, a mesh-based method has to rely on the proper reconstruction of the features during the mesh generation.

3 Point Based Methods

Very few feature detection methods are dedicated to point-sampled geometry only. The major problem of these point based methods is the lack of knowledge concerning normal and connectivity information. This makes feature detection a more challenging task than in mesh based methods.

Gumhold et al. [5] present a method that uses the Riemannian graph to construct the connectivity information for the point cloud. The Riemannian is the graph, that contains the edges to the k nearest neighbors for every data point. The algorithm first analyzes the neighborhood of each point via a principal component analysis (PCA). The eigenvalues of the correlation matrix are then used to determine a probability of a point belonging to a feature. The analysis of the ellipsoid formed by the three eigenvectors and their eigenvalues allows further conclusions about the underlying feature type. This way the algorithm can differentiate between line-type features, border and corner points. The result is a quite dense set of points covering all kinds of features independent if the feature is sharp or not. This set of points is then reduced by computing a minimal spanning tree followed by a branch cutting. This is an elegant way to obtain a sparse set of points representing the feature line.

Pauly et al. [11] extended the PCA approach with a multi scale analysis of the neighborhoods. Based on the eigenvalue analysis of the covariance matrix, they compute a value for the surface variation in the local area around a sample point. To obtain more information, they use a multi scaling approach that varies the size of these neighborhoods. That means, they apply their feature detection operator to multiple neighborhood sizes, which allows to measure the persistence of the feature. A jump in the graph of the surface variation during the multi scaling shows the existence of new surface parts. Especially in noisy datasets, the multi scaling approach enhances the result of the usual PCA analysis. Since the method analyzes up to 200 neighborhood sizes for each point in the dataset, it is computationally more expensive. To handle relative huge neighborhood sizes of over 200 neighbors, they also show a way to solve the problem of neighbors not belonging to the same connected region. To estimate when a neighborhood becomes too large, they use a heuristic that looks for strong deviations in the normal direction. The algorithm recognizes all kinds of visual eminent features, but for the identification of only the sharp features inside the dataset, this method has to be modified. One way to adapt this approach to sharp features may be achieved with an adjustment of the thresholds for the feature recognition. With well chosen lower and upper thresholds the method may be able to identify only sharp features.

Demarsin et al. [2] also searched for sharp features in point cloud data. Their goal is to produce closed sharp feature lines. They choose a region growing method that segments the point cloud into clusters and identify the regions of sharp features. Based on the analysis of the normals of the points, they segment the point cloud in clusters with equal normal behavior. From these clusters they build up a graph that connects the neighboring clusters. The edges in this graph are then used as indication for the existence of a sharp feature in the related area. Similar to Gumhold [5] and Pauly [11], they use a graph approach and construct a minimum spanning tree of these candidates. This gives them an initial reconstruction for the feature lines. A fixed parameter for the maximum branch length is then used to cut of the short branches of the tree. In the next step, they close the feature lines. For each open endpoint in their graph they compute the n nearest neighbors among the other endpoints. The distance and the length of the paths of the neighboring endpoints is used to determine a good connection. After this they cut of the branches of the possibly remaining endpoints and smooth the graph to get their final closed feature lines.

The mentioned techniques for the detection of features in point clouds are mostly used as a preprocessing step for another processing step, e.g. a surface reconstruction with sharp features, but there exist also several reconstruction methods that preserve sharp features during a surface reconstruction of a point cloud without preprocessing. For example the methods shown by Amenta et al. [1], Guy and Medioni [4], Fleischmann et al. [3] and Öztireli et al. [10]

4 Feature extraction via Gaussian Map Clustering

This section presents our feature extraction method. It is a point-based method like the methods mentioned in Section 3. The method can be divided into three steps: In the first step, the data structure used for the analysis of the point set is built. Subsequently this data structure is used to generate local neighborhoods inside the point set. In the last step these neighborhoods are analyzed, and points belonging to sharp features are identified.
We define a *point cloud* as a simple set of 3D point coordinates $P = \{p_1, p_2, ..., p_N\}$, $p_i \in \mathbb{R}^3$ without any normal or connectivity information. The data points are unstructured, but supposed to belong to a 2-manifold surface. Let $N = |P|$ be the number of points.
The type of sharp feature we want to detect in the point cloud can vary from edges or lines between two surfaces to corners where three or more surfaces meet.

4.1 Analysis of neighborhoods

To detect sharp features in the point cloud, we have to analyze the neighborhood of every point in the dataset, similar to [11],[5], and decide if the point is a sharp feature point or not. As neighborhood, we use the *k-nearest*, i.e. the k points with the shortest distance to the sample point. The k-nearest neighbor search is a well studied problem and many algorithms exist, since it is just a variant of the nearest neighbor search problem. For performance reasons, we use a kd-tree implementation as underlying data structure. Building the kd-tree is performed in a pre-processing step for our algorithm. After the construction of the local neighborhood N_p for a sample point $p \in P$, the next step is to analyze it and decide if p belongs to a sharp feature or not. During tests a neighborhood size of 16 turned out to deliver good results. For the following analysis we apply a Gauss map clustering.

4.2 Discrete Gauss map

Let N_p be the neighborhood of p containing the k nearest neighbors and $I_p = \{1, ..., k\}$.

We now take a set T of all possible $k \cdot (k-1)$ triangles with p and two neighborhood points as vertices

$$T = \{ \quad \Delta_{ij} = \Delta(p, p_i, p_j) \quad | \quad i \neq j, \quad i, j \in I_p \}.$$

An example for three possible triangles is shown in Figure 2. The normal vector of one of these triangles Δ_{ij} is given by

$$n_{ij} = \overline{pp_i} \times \overline{pp_j}. \tag{1}$$

Note that $n_{ij} = -n_{ji}$. The discrete *Gauss map* of the neighborhood of p can now be defined as the mapping of T onto the unit sphere S^2 centered at p as follows

$$\begin{array}{ll} G_p & : T \to S^2 \\ \Delta_{ij} & \mapsto x_{ij} := p + \frac{n_{ij}}{\|n_{ij}\|}. \end{array} \tag{2}$$

Figure 1 shows the projection onto the gaussian sphere.

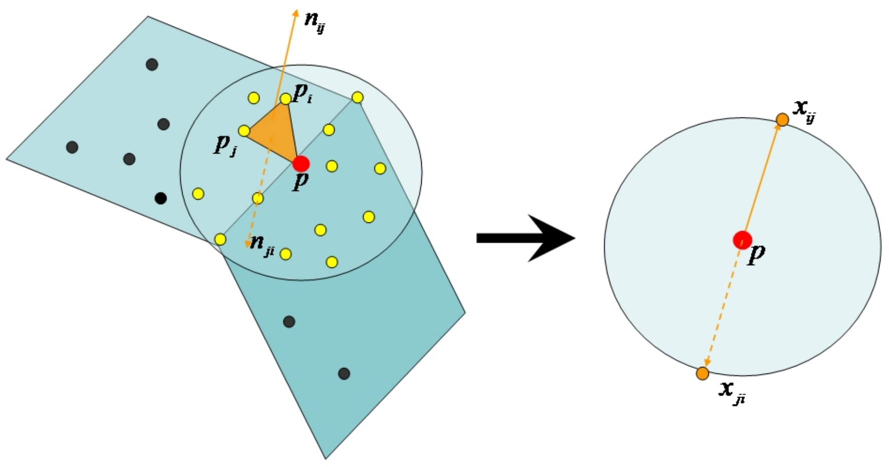

Figure 1 The projection of the normal of one of the possible triple (p, p_i, p_j) onto the gaussian sphere, with the two possible results x_{ij} and x_{ji}

4.2.1 Gauss map clustering

Feature detection is now performed by analyzing the clustering behavior of these normals on the Gauss map G_p of the set $T = \{\Delta(p, p_i, p_j)\}$.

The motivation for this idea is the fact that in the case of a smooth piecewise C^0 surface, the patterns of resulting cluster on the Gauss map is different whether the point is flat, curved (elliptic, hyperbolic or parabolic) or tangent plane discontinuous. In the nearly flat case, the Gauss map of neighbor points will present one cluster of points on the sphere, see Figure 3 (left). In the case of a curved point (parabolic, hyperbolic, or elliptic) the points will not form clusters, but spread on the sphere over a larger region, see Figure 3 (middle). And a tangent plane discontinuity will lead to a pattern, where the points of the sphere form two distinct clusters, see Figure 3 (right).

Working with a simple point cloud, the difficulty is that we have only very limited information about the underlying surface. No local triangulation or normal vectors are available. For this reason, we defined the Gauss map as projection of all triangles in T (see Sect. 4.2). As a consequence of using all triangles $\Delta(p, p_i, p_j)$, the resulting Gaussian map will contain some additional noisy points that correspond to the triangles with p and one data point from each of the planes. These noisy points are distributed over the whole gaussian sphere (see Figure 2 right), while the correct points (i.e. points belonging to triangles of p and two points of one of the planes) are positioned in clearly recognizable clusters. The dominance of the clusters makes it easy to ignore the noisy points during the computation of the clusters. Figure 2(right) illustrates these assumptions for the common case of p lying on the sharp edge of two intersecting planes, with half of the $k = 16$ neighborhood points lying on each plane. The result of the Gauss map computation shows two pairs of (opposite) clusters of $O(k^2/4)$ identical points. These pairs of clusters correspond to each of the intersecting planes. Note that the reason for the pairwise clusters is the lack of knowledge about the original surface, especially the normal directions. The points in the neighborhood are unsorted and so we can not control which triangle ($\Delta(p, p_i, p_j)$ or $\Delta(p, p_j, p_i)$) is used for the gaussian clustering. Based on this fact, both normals, n_{ij} and n_{ji} are normals of possible triangles, also see Figure 1. The result are clusters on opposing sides of the sphere, although they belong to the same plane, but where arbitrary projected into opposing directions. The red points on the right picture in Figure 2 are an example for these opposing clusters. All other points on the sphere (noisy points) correspond to situations with one point of each plane and the sample point forming a triangle. This noise is sparsely distributed over the sphere. The example in Figure 2(right) was implemented in Matlab.

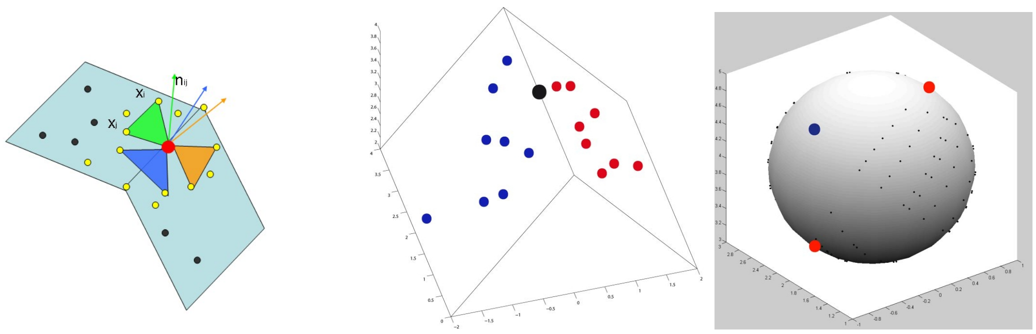

Figure 2 Left: Computation of the normal vectors used for feature identification. Middle: An example for a sharp feature situation. Right: The Gauss map G_p for the feature identification of the example in the middle, showing the clusters (big red and blue points) and the noisy points generated from wrong triangles.

Using this knowledge about the clustering behavior on the Gauss map, called Gauss map clustering, we can now determine whether the sample point p belongs to a sharp feature or not. Regarding cluster analysis, we have to consider the effect of the opposing clusters mentioned above. Since we don't have any information whether the normals points outside or inside of the surface and thus $n_{ij} = -n_{ji}$, each point x_{ij} in one of the clusters has its counterpart x_{ji} in the cluster on the other hemisphere of the Gaussian map.

For this reason and to avoid possible problems and wrong detections, we have to guarantee that opposing clusters are treated as one cluster. So instead of the original vectors, we use the lines defined by these vectors for the clustering. After clustering, the results are interpreted as follows. The case of one resulting cluster (i.e. two opposing clusters) corresponds to a flat

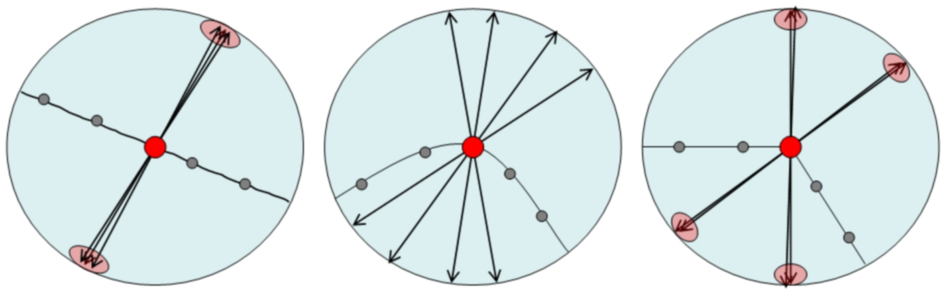

Figure 3 2D examples for clustering during feature detection

or nearly flat areas (Figure 3 left). Two or more clearly distinguishable clusters show the existence of a sharp feature nearby and the sample point of this neighborhood belongs to this feature (Figure 3 right). The case of no clustering and a sparsely distribution of points on the Gaussian map shows, that this region is curved or has at most a smooth feature, but not a sharp one (Figure 3 middle).
In many clustering algorithms one needs to specify the number of clusters to produce. However, we have to go the other way around. We want to distinguish our real clusters from sparsely distributed points in the Gauss map, and afterwards count the number of clusters. Therefore we use a hierarchical agglomerative ("bottom-up") clustering method [8]. Starting with one separate cluster for each point, we merge them step by step into larger clusters. Here the distance, which is used as the criterion for the merging process, and its definition is of importance for the success of the clustering.
We use the mean distance D_c between the elements as criterion, and define the distance as the angle between the lines which are defined by the opposing clusters as follows:

$$D_c(S_1, S_2) = \frac{1}{|S_1| \cdot |S_2|} \sum_{x \in S_1} \sum_{y \in S_2} d(x, y), \tag{3}$$

where S_1, S_2 are two clusters to be compared, $|S|$ is the number of elements in a cluster and d is the distance measure on our Gauss map. Each agglomeration increases the distance between the clusters. As a consequence, we can stop the clustering algorithm when the distance between the existing clusters exceeds a certain threshold $\sigma \in [0, \frac{\pi}{2}]$.
After the clustering we have to discard all clusters with only a few points. We do so to eliminate the effects of the noisy points mentioned earlier. If only one cluster remains, we know that the sample point does not belong to a sharp feature since the underlying area is a flat plane. If two to four clusters remain, we say that the point belongs to a feature. If the result consists of more than four clusters or no cluster at all, we decide that the point is not a feature since the result of many or no clusters is a strong signal for a curvy area without a sharp feature. For datasets with sharp features where more than four edges join, the parameter can be adjusted to match the occurring situation.

4.3 Sensitivity to parameter choice

The threshold σ is the parameter we can use for the sensitivity of the feature detection. It corresponds to the angle of the sharp feature we want to detect. The choice of the value of σ depends on the dataset. A dataset with obtuse angles will need a lower value for σ than a dataset with acute angles. The effect of σ is shown in Figure 4. On the left side a too small σ was chosen, the features were not detected properly. In the middle σ was well chosen, the features are well detected. On the right side σ was chosen too big, resulting in the detection of too many feature candidates. The user adapts the σ value with a trial and error method.

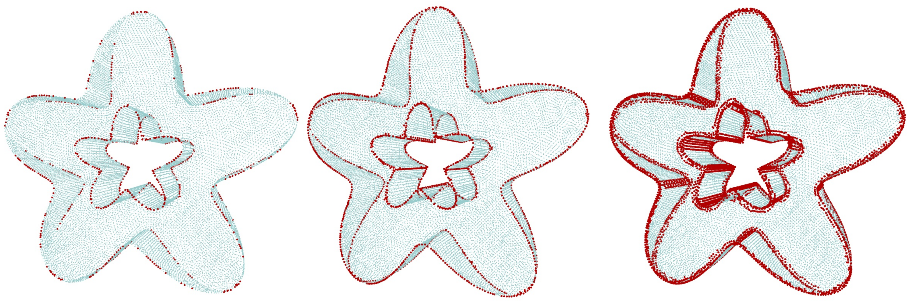

Figure 4 Different values of σ tested with the trim-star model, σ growing from left to right.

4.4 Sensitivity to noise

We also tested the methods behavior regarding noise. To do so, we generated noise by moving the points of the original dataset in a random direction with a maximum length of 5%, 10% and 15% of the dataset's bounding box. The results are shown in Figure 5. One can see that the method is relatively stable concerning noise and only strong noise results in the presence of outliers and false positive features (i.e. points which are detected as features but which are not features). To improve the result it would also be possible to use some smoothing techniques to reduce the outliers in the presence of strong noise. Indeed, just removing isolated candidates which are not surrounded by other features would reduce the resulting number of outliers significantly.

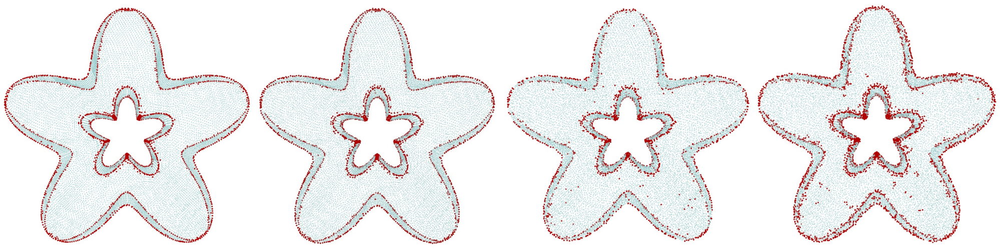

Figure 5 Noise test on the trim-star model. From left to right: no noise, 5%, 10%, 15% noise

5 Results and conclusion

The proposed method for sharp feature detection was tested on various constructed and real world examples and delivered good results, see Figure 6. It does not rely on local surface reconstructions, and no normal information is required. The resulting point cloud with the marked sharp features can be used for several applications like surface reconstruction, non-photorealistic rendering, mesh generation or surface modeling. In [5] a whole section is devoted to possible applications. Currently, the threshold for the sensitivity is user controlled and specific for a dataset. For most datasets, an automatic computation of the parameter

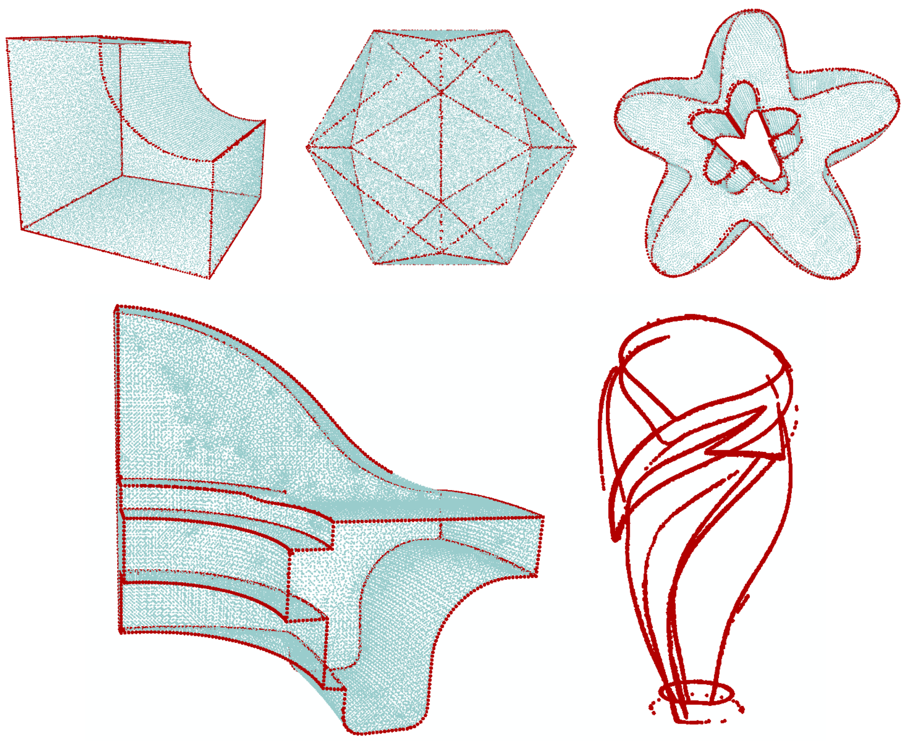

Figure 6 Sharp feature detection on a cube, an icosahedron, the trim-star model model, the fandisk and a vase model.

can reduce user dependency. The method detects line-type and corner-typed sharp features. Cone peak features are currently not recognized by it, but it should be possible to extract a particular clustering behavior for these cases and adapt the method to recognize these.

Acknowledgements

The vase model is provided courtesy of INRIA, the fandisk and trim-star are provided courtesy of MPII, all by the AIM@SHAPE Shape Repository. This work was supported by the IRTG 1131 of the DFG (German Research Foundation). It has been partially supported by the Deutsche Forschungsgemeinschaft INST 248/72-1.

References

1 Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 249–266, New York, NY, USA, 2001. ACM.

2 Kris Demarsin, Denis Vanderstraeten, Tim Volodine, and Dirk Roose. Detection of closed sharp edges in point clouds using normal estimation and graph theory. *Comput. Aided Des.*, 39(4):276–283, 2007.

3 S. Fleischmann, Daniel Cohen-Or, and C.T. Silva. Robust Moving Least-squares fitting with sharp features. *ACM Trans. Graph.*, pages 37–49, 2005.

4 G. Guy and G. Medioni. Inference of surfaces, 3D curves, and junctions from sparse, noisy, 3D data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1265–1277, 1997.

5 Stefan Gumhold, Xinlong Wang, and Rob McLeod. Feature extraction from point clouds. *Proceedings of 10th International Meshing Roundtable*, 2001.
6 Andreas Hubeli and Markus Gross. Multiresolution feature extraction for unstructured meshes. *Proccedings of IEEE Visualization*, pages 287–294, 2001.
7 K. Hildebrand, K. Polthier, and M. Wardetzky. Smooth feature lines on surface meshes. *Proceedings of Symposium on Geometric Processing*, 2005.
8 Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, Boston, MA, USA, 2th edition, 2009.
9 Leif Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel. Feature sensitive surface extraction from volume data. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 57–66, New York, NY, USA, 2001. ACM.
10 Cengiz Öztireli, Gael Guennebaud, and Markus Gross. Feature Preserving Point Set Surfaces based on Non-Linear Kernel Regression. *Computer Graphics Forum*, 28(2), 2009.
11 Marc Pauly, Richard Keiser, and Markus Gross. Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum*, 2003.
12 Kouki Watanabe and Alexander G. Belyaev. Detection of salient curvature features on polygonal surfaces. *Computer Graphics Forum*, pages 385–392, 2001.
13 Tino Weinkauf and D. Günther. Separatrix Persistence: Extraction of Salient Edges on Surfaces Using Topological Methods. *Computer Graphics Forum (Proc. SGP '09)*, 28(5):1519–1528, July 2009.