Effective parameters for feature line detection from surface point clouds

Martin Christ\textsuperscript{1)}, Thomas Kalbacher\textsuperscript{2)}, Takeo Taniguchi\textsuperscript{3)}, Olaf Kolditz\textsuperscript{4)} and Keiji Mashimo\textsuperscript{5)}

1) M.Sc., Center of Applied Geology, Eberhardt-Karls University (Sigwartstr. 10, 72076 Tübingen, Germany)
2) Dipl.-Geol., Center of Applied Geology, Eberhardt-Karls University (Sigwartstr. 10, 72076 Tübingen, Germany)
3) Prof., Faculty of Environmental Science & Technology, Okayama University (3-1-1, Tsushima-naka, Okayama 700-8530)
4) Prof., Center of Applied Geology, Eberhardt-Karls University (Sigwartstr. 10, 72076 Tübingen, Germany)
5) Senior Engineer, Automotive Technology Center, Furukawa Electric Co., Ltd. (2-4-3, Okano, Nishi-ku, Yokohama 220-0073)

Accurate and consistent methods are required to reconstruct the surface of an object from a set of points, which are located on the surface of the object. The detection and generation of feature lines appearing on the surface is considered to be an attractive preprocessing step before the surface reconstruction process. In this paper, the authors propose a method to detect feature lines from point clouds. The detection and generation process depends on a number of effective parameters. The authors have identified four effective parameters for feature line generation: 1) the number k nearest neighbors, 2) curvature value, 3) cycle length and 4) minimum length of short branches. The efficiency of the detection by finding the parameter values that closely match the existing feature lines are examined through test problems of point clouds. Finally, the effectiveness of the method is evaluated by application to a remote sensed point data set representing topographic features of the Jordan valley.

\textbf{Key Words:} Mesh generation, point clouds, feature line, surface generation of 3D domain

1 Introduction

Accurate and consistent representations of three-dimensional objects are becoming increasingly important for many applications in engineering. Three-dimensional objects can be generated from a given set of geometric sample values. In many applications, these sample values are points that can be easily sampled from the surface of an object by remote sensing, laser scanners or other technologies [1, 2].

Surface reconstruction of an object from such measured point data typically begins with a determination of the topology of the objects surface by evaluating the neighborhood relations of points on different surfaces [3]. This process requires consideration of individual features such as edges or feature lines [4]. The assumption is that any three-dimensional object is composed of smooth surfaces that are separated by feature lines. The idea of surface reconstruction from feature lines is to fit multiple individual surface patches to the point cloud by triangulation; the individual patches are then connected together along the feature lines to form the complete surface [5]. The individual surfaces are generated by finding a set of triangles to cover the surface. Such triangles need to be created and possibly joined using the nearest neighbor graph. Finally, possible post-processing involves model refinement and surface quality assurance.

The problem of generating feature lines discussed in this study is closely related to the feature extraction problem described by Gumhold et al. [4]. Their approach is to extract different types of existing feature elements from point clouds such as crease and borderlines, or singleton ends. In contrast to their research, it is proposed to extract all potential feature lines of a data set, which depends on a number of parameters describing the geometry of the point cloud and the feature lines. Finding these parameter values will allow to closely match existing feature lines.

This study presents a method to detect and generate feature lines based on curvature approximation from surface point clouds, test results and the limitations of the method. It attempts to identify effective parameters for feature line generation and discusses the effects of parameter change on the quality of feature lines generated. The application of the method to a real world data set is another important aspect of this study.

2 From Points to Feature Lines

The input domain is a set of points in three-dimensional space located on the surface of an object. We want to detect and generate feature lines from these points by approximating the surface curvature. Surface curvature is proposed as one geometric property, which is valid for finding the difference between points near feature lines and on smooth surfaces. The first step involves the determination of the neighborhood for every point, which is employed for the classification of points according to surface curvature into a) points on smooth surfaces and b) points near feature lines. In the second step, potential feature lines are extracted. Then cycles in the feature line pattern are identified and gaps closed if necessary. Finally, short branches are removed. The procedure of generating feature lines is illustrated in Figure 1; in this case a cube.
2.1 Nearest Neighbor Query

The nearest neighbour query retrieves the \( k \) nearest neighbours of every point \( p_i \) and defines the neighborhood, which serves as domain for feature line detection. The neighborhood should only consist of points that are “near” to the given point [6]. The neighborhood should only consist of points that are “near” to the given point [6]. The simplest way to find the \( k \) nearest neighbours is by computing the Euclidian distance between all pairs of points. The \( k \) smallest distances for every data point are recorded and sorted [7].

2.2 Point Penalty Function

The information of the nearest neighbourhood is employed to derive a point penalty function that approximates surface curvature to estimate how likely it is that a point is close to a feature line. The calculation of the penalty function for curvature follows the approach suggested by Gumhold et.al. [4] and defines a penalty weight to every point \( p_i \) in the data set. In the first step, the average distance of every point \( p_i \) is calculated to its set of neighbours \( N_i \) given by

\[
a = \frac{1}{|N|} \sum_{q \in N_i} (q - p_i)
\]  

(1)

The centre location \( c_i \) and the correlation matrix \( C_i \) calculated for the set of \( k \) nearest neighbours \( N_i \) are given by

\[
c_i = \frac{1}{|N|} \sum_{q \in N_i} q
\]  

(2)

\[
C_i = \frac{1}{|N|} \sum_{q \in N_i} (q - c_i)(q - c_i)^T
\]  

(3)

From the correlation matrix the eigenvalues \( \lambda_1, \lambda_2 \) and \( \lambda_3 \) and corresponding eigenvectors \( e_0, e_1 \) and \( e_2 \) are computed by employing Jacobi iteration. Distance \( d \), which is the distance of point \( p_i \) from its tangent plane, is calculated with the eigenvector \( e_0 \) given by

\[
d = e_0^T (p_i - c)
\]  

(4)

The curvature \( K \) is calculated for every data point \( p_i \) and its corresponding neighbors \( q \) with distance \( d \) from the plane as shown in Figure 2.

\[
K = \frac{2d}{a^2}
\]  

(5)

The curvature penalty function is defined as

\[
\omega_{ki} = 1 - \frac{K}{K_{max}}
\]  

(6)

From the penalty function it becomes evident that the curvature value ranges from 0 to 1. A small curvature value indicates that a point is likely to be close to a feature line, whereas a large value indicates that a point is likely to be located on a smooth surface.

2.3 Minimum-spanning tree

From the curvature penalty weights, a minimum-spanning tree is computed that represents the final feature line pattern. In the first step, the penalty weights for every point \( p_i \) are transferred to the edges of the neighbor graph given by

\[
\omega_{edge} = 0.5 (\omega (p) + \omega (q))
\]  

(7)

The edges are sorted according to their weights, while edges that have a weight exceeding a defined threshold value are not considered. Additionally to the edge weight, an edge length term that favours short edges is employed. The algorithm to compute the minimum-spanning tree starts with the lowest weighted point and finds the lowest weighted edge that connects the point to one of its neighbors. Then, from either two points, it will find the next lowest weighted edge, without creating a cycle. The procedure continues until there are no edges connecting to a point in the minimum-spanning tree anymore. The next lowest weighted point that is not yet added to the minimum-spanning tree is taken and the procedure repeated. A depth first search algorithm allows easy computation of the shortest path between two points and is utilized to search recursively for cycles in the minimum-spanning tree. The process begins with any point in the minimum-spanning tree and searches as deep into the tree as possible from this point.
As the whole graph is searched, it does not matter which point in the minimum-spanning tree is selected as starting point. The path is then backtracked to the current point until an unvisited path is found and goes as deep as possible from there. This process is continued until the starting point is found. If the entire graph has been searched, the procedure is stopped. Otherwise, an unexplored point is selected and the search starts from there again.

2.4 Pruning

Pruning is the process of removing short branches. The feature line pattern looks like the one in Figure 1 c). Some of the branches are actually not part of the feature line pattern and need to be removed. The approach for pruning short branches considers every point \( p_i \) in the minimum-spanning tree with at least one incident edge. This condition satisfies the possibility of having only single edges in the minimum-spanning tree.

3 Effective parameters

In the process of detecting and generating feature lines, four parameters are identified to influence the quality of the final feature line pattern. For any given problem, an optimal value for the \( k \) nearest neighbors, the curvature value to detect points close to a feature line, the minimum length of a cycle in the minimum-spanning tree and the minimum length of short branches should be identified.

3.1 Number of \( k \) nearest neighbors

The number of \( k \) nearest neighbors defines how many points are included in the nearest neighbor query and directly influence the estimation of curvature value for every point \( p_i \). The \( k \) nearest neighbors can be thought of as a smoothing parameter. For any given data set, the parameter \( k \) should be set to a value small enough to select only the points that are close enough to the query point and large enough to reduce the probability of variance in predictions.

3.2 Curvature value

The definition of a curvature threshold value is essential for classifying points into two groups a) points on smooth surfaces and b) points near feature lines. A small value for curvature indicates that a point is likely to be close to a feature line, whereas a large value for curvature indicates that a point is likely to be located on a smooth surface. This implies that a high threshold value for curvature will potentially classify more points to be part of a feature than a low threshold value. The curvature value can therefore be considered as a measure of sensitivity of feature line detection.

3.3 Cycle length

This parameter defines the minimum length of a cyclic feature line in the pattern. It is foremost used to close gaps in the feature line pattern, but also to avoid smaller cycles. Gumhold et al. [4] suggest that given the number of input points that cyclic feature line pattern should have at least the length of half the diameter of the described object. The diameter of an object refers here to the number of links (edges) between the two furthest points in a given data set.

3.4 Length of short branches

The parameter for the minimum length of branches defines the length of a feature line to be considered in the final pattern and is employed to refine the final feature line pattern. Selecting a small value for the minimum length many short branches are included, while choosing a larger value short branches are pruned away. Gumhold et.al. [4] suggested a value of cycle length/2 to remove short branches.

4 Application to test problems

The performance of the proposed algorithm is evaluated by using several models; cube, cosine, shell, cone, tetra and deco. In the first part, the effects of changing the parameter values on the extracted feature line pattern are illustrated for the cube model, while one parameter value is changed at the time and the remaining parameter values are kept constant.

4.1 Test results

At first experimental tests considered changes in the number of \( k \) nearest neighbors and the curvature value under the following assumptions: 1) given the number of input points a cyclic feature line pattern should have at least the length of half the diameter of the described object. For the cube model with 685 data points, this condition is satisfied by a length longer than 13 points and 2) branches can be safely removed if they are shorter than half the cycle length. For the cube model this condition is fulfilled by a length shorter than 7 points.

In the proposed algorithm, the average distance of the \( k \) nearest neighbors is employed to derive the point penalty function. Changing the number of neighbors in the nearest neighbor query suggests a change in the average distance and consequently, a change in the point penalty weight for every point \( p_i \). Figure 3 a) to c) shows a stepwise increases in the number \( k \) nearest neighbors for the cube model. It becomes evident that with a very small value for \( k \), no adequate set of feature lines is generated. Increasing the value to six or more neighbors, results in the desired feature line pattern for this model.

Figure 4 illustrates the effect of increasing the curvature value. From the definition of the curvature penalty function we can assume that with increasing curvature value the number of points classified to be close to the feature line increases. This trend is demonstrated in figure 4 a) to c). The curvature value is stepwise increased and an increase in number of points selected to be close the feature line can be
observed. However, an increase in curvature value from 0.2 to 0.8 does not cause any change in feature lines generated for the cube model.

The last parameter that is found to influence the quality of generated feature lines is the minimum length of short branches. By changing the minimum length a feature line should have to be included in the final feature line pattern allows pruning of undesired short branches. It is found that by setting the branch length to the suggested value of half the length of the minimum cycle all short branches are successfully removed illustrated in Figure 7.

In the process of generating features lines for the other models we identified a number of problems. One striking problem is related to borderlines. They represent a special type of feature lines indicating the border of the point cloud domain. The borderline extracted from the shell model in Figure 8 a) is simply the result of the initial surface curvature of the model. In contrast, the deco data set in Figure 8 b) and c) is composed of three absolutely plane surfaces with no initial curvature. As all neighbors of a point $p_i$ that is located close the border are situated on a plane with no initial curvature, the algorithm will compute a curvature value of 1 also for the border point $p_i$. Thus, it will classify the point to be located on a smooth surface instead of being close to the borderline.

Another difficulty arises from single corner points, which could be successfully identified based on the curvature value with this algorithm. Nevertheless this information could not be integrated into an adequate set of feature lines. This is shown in Figure 9.
The cosine model in Figure 10 shows the question of generating feature lines on smooth surfaces. The two outer feature lines are successfully extracted, but no feature lines at surfaces with gradual changes in surface curvature.

Finally, the enban model (Figure 11) shows the problem of extracting an appropriate feature line pattern for smooth surfaces being very close to each other. The algorithm failed to identify the points closed to the feature line.

4.2 Summary of test results

Table 1 and 2 attempt to give an overview of the tested models and summarise some of the parameter combinations that achieve acceptable results for the tested models, respectively.

Table 1. Summary of parameter values or ranges that generate the most appropriate set of feature lines for the tested models

<table>
<thead>
<tr>
<th>Model</th>
<th>k value (no. of neighbors)</th>
<th>curvature value</th>
<th>d value (minimum cycle length)</th>
<th>b value (minimum branch length)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cube</td>
<td>8 - 18</td>
<td>0.2 - 0.9</td>
<td>12+</td>
<td>4+</td>
</tr>
<tr>
<td></td>
<td>6 - 18</td>
<td>0.3 - 0.5</td>
<td>8+</td>
<td>2+</td>
</tr>
<tr>
<td></td>
<td>6 - 18</td>
<td>0.6+</td>
<td>12+</td>
<td>4+</td>
</tr>
<tr>
<td>cosine</td>
<td>16 - 18</td>
<td>0.4</td>
<td>12+</td>
<td>3+</td>
</tr>
<tr>
<td>sine</td>
<td>7 - 8</td>
<td>0.4</td>
<td>8+</td>
<td>8+</td>
</tr>
<tr>
<td>cone</td>
<td>6</td>
<td>0.4 - 0.6</td>
<td>16+</td>
<td>8+</td>
</tr>
<tr>
<td>tetra</td>
<td>20</td>
<td>0.6 - 0.8</td>
<td>12+</td>
<td>4+</td>
</tr>
<tr>
<td>shell</td>
<td>16</td>
<td>0.4</td>
<td>8+</td>
<td>2+</td>
</tr>
<tr>
<td></td>
<td>10 - 16</td>
<td>0.6</td>
<td>10+</td>
<td>4+</td>
</tr>
<tr>
<td>daikai</td>
<td>6-16</td>
<td>0.4 - 0.6</td>
<td>6+</td>
<td>2+</td>
</tr>
</tbody>
</table>

5 Application real world problems

An important objective of this study is the investigation of the effectiveness of the proposed algorithm for a model from the physical world. The method is therefore applied to a remote sensed point data set describing the topography of the Jordan Valley and its western escarpments. The data set consists of 8000 irregularly distributed points on terraces and cliffs as shown in Figure 12 a) and b). The algorithm detected effectively points located close to potential feature lines based on the curvature value. However, no adequate set of feature lines is generated from this point information. Two trends can be recognized: 1) points are located to be close to a feature line, but no feature line is generated and 2) the feature line generated shows a zigzag pattern and connects points located close to different feature lines with each other. It is assumed that the lack of point information or the irregular distribution of points causes both observed trends. It is assumed that increasing the number of points in particular on the vertical sections, i.e. cliff faces will considerably improve the feature line pattern generated. Further, a more equal distribution of points over the surface may also aid to improve the quality of generated feature lines.

Fig. 12 a) and b). Remote sensed data set: Jordan Valley. Points close the feature lines marked in red, Feature lines in black.
Table 2 Summary of tested models giving the number of points, their shape and a description on point distribution

<table>
<thead>
<tr>
<th>Model</th>
<th>Points</th>
<th>Description</th>
<th>Shape</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>cube</td>
<td>685</td>
<td>Cube with irregularly placed points on its surface, points</td>
<td>volume, single corner</td>
<td>points</td>
</tr>
<tr>
<td></td>
<td></td>
<td>directly located on feature lines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cosine</td>
<td>890</td>
<td>Cosine with regularly placed points on its surface, points</td>
<td>volume</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>directly located on feature lines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sine</td>
<td>705</td>
<td>Sine with regularly placed points on its surface, points directly</td>
<td>volume</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>located on feature lines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>shell</td>
<td>593</td>
<td>Shell with irregularly placed points on its surface, some points</td>
<td>3-D surface, borderline</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>directly located on feature lines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deco</td>
<td>3168</td>
<td>Deco data set representing an artificial shape with irregularly placed</td>
<td>3-D surface, borderline</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>points on its surface, points directly located on feature lines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cone</td>
<td>1749</td>
<td>Cone with regularly placed points on its surface, points directly</td>
<td>volume, single corner</td>
<td>point</td>
</tr>
<tr>
<td></td>
<td></td>
<td>located on feature lines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tetra</td>
<td>803</td>
<td>Tetrahedra with regularly placed points on its surface, points directly</td>
<td>volume, single corner</td>
<td>points</td>
</tr>
<tr>
<td></td>
<td></td>
<td>located on feature lines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>daikei</td>
<td>1514</td>
<td>Partial cone with regularly placed points on its surface</td>
<td>volume</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>enban</td>
<td>211</td>
<td>3D concave cone (very flat) with regularly placed points on its</td>
<td>volume</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>surface</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6 Conclusions

The work of this study leads to the proposal of an algorithm that approximates surface curvature to detect and generate feature lines from surface point clouds. The algorithm robustly detects and generates feature lines for most tested models depending on the selection of an appropriate combination of parameter values. Four parameters were identified to be important for the quality of feature lines generated; a) the number of neighbors, b) the curvature value, c) the minimum cycle length and d) the minimum length of a branch. The proposed method did not adequately address the problem of borderlines and single corner points. The information of single corner points could not be integrated adequately into the final feature line pattern. Further we identified difficulties to generate an adequate set of feature lines for models with gradual changes in surface curvature. Limitations due to the lack of point information became apparent in the application to the Jordan Valley data set. Future research needs to be concerned to identify other factors for detecting points close to feature lines such as the maximum angle in the neighbor graph of point $p_i$ and the summation of all angles in the neighbor graph. For the application of the proposed method to a real world data set a sufficient number of sample points and a suitable distribution is desired.

References

1) Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. and Stuetzle, W. 1992. Surface reconstruction from


