Extracting and Visualizing Structural Features in Environmental Point Cloud LiDaR Data Sets

Patric Keller[1], Oliver Kreylos[2], Marek Vanco[2], Martin Hering-Bertram[3], Eric S. Cowgill[4], Louise H. Kellogg[4], Bernd Hamann[2], and Hans Hagen[1]

[1] University of Kaiserslautern, Department of CS., 67663 Kaiserslautern, Germany
[2] Institute for Data Analysis and Visualization (IDAV), Department of Computer Science, UC Davis, CA 95616
[3] Fraunhofer-Institute ITWM, 67663 Kaiserslautern, Germany
[4] University of California, Department of Geology, UC Davis, CA 95616

Abstract. We present a user-assisted approach to extracting and visualizing structural features from point clouds obtained by terrestrial and airborne laser scanning devices. We apply a multi-scale approach to express the membership of local point environments to corresponding geometric shape classes in terms of probability. This information is filtered and combined to establish feature graphs which can be visualized in combination with the color-encoded feature and structural probability estimates of the measured raw point data. Our method can be used, for example, for exploring geological point data scanned from multiple viewpoints.

1 Introduction

Exploration of environmental point data sets is a challenging problem of recent interest. The LiDaR (Light Detection and Ranging) technology makes it possible to capture fast and accurate point data over large regions. As a consequence, LiDaR data sets are typically very large containing diverse and highly complicated objects. Occlusion, semi-transparent and reflecting objects, structures with small fractured and under-sampled surface components like trees, as well as scanning error due to motion of cars, humans, animals, etc. make the data difficult to interpret. While the raw data contains scientifically relevant structural information, this information, unless proper filtering and pre-processing methods are applied to the raw data, remain "hidden" in most direct visual representation due to data over-load and noise.

A possible way to explore point data is based on surface reconstruction [4, 9, 10]. Building consistent triangular meshes from such data is a difficult and often ambiguous task. Point-based approaches [1, 24] operate directly on the sample points without requiring the computation of mesh-based connectivity information. These approaches often use surface elements (*surfels*), assuming that point sets define smooth surfaces with well-defined normal vectors. For LiDaR data sets, this assumption may not always hold and the surfel-based approach may also hide topological ambiguities and, as a consequence of smooth interpolation schemes, eliminate structural artifacts, due to discontinuities in the original scanned geometry, that might be important. To avoid these drawbacks, we use a probability-based classification of the point cloud into subsets having different structural characteristics. Depending on this classification and a user's choices of parameter values we perform the extraction of feature graphs describing the structural composition of the point cloud. We obtain an explicit structural description of the whole data which can be used for further processing (e.g., identification and classification of objects in environmental data) or for exploration and visualization.

For geologists, our approach is of special interest since it makes possible the detection and visualization of features like creases in structures caused by earthquakes or rapid and automatic identification of key features in environmental data such as ridge lines, stream beds, and edges of terraces. When combining data from multiple scans, registration errors make feature detection and visualization even more challenging. Our goal is to provide a user with effective tools for the identification and interactive exploration of structural features in LiDaR data sets.

In section 2, we review related work on point set rendering and surface reconstruction. Section 3 discusses the specific problems of analyzing LiDaR data and proposes a classification of features used in our algorithm. The construction of feature graphs is discussed in section 4.

2 Related Work

The potential of point-based methods has been demonstrated in different applications [2],[19]. However, algorithms to process point sets are still in their infancy when compared with algorithms for common mesh-based approaches. In computer-aided design (CAD), point sets may be used as underlying representation for surface editing [24] (like the Pointshop 3D framework proposed by Zwicker et al. [28]). Several authors based their segmentation and surface recognition algorithms directly on point sets, [6],[7]. Concerning surface reconstruction from laser-scanned point sets, much research has been done following the approach by Eck and Hoppe [13].

In the context of complex geology-driven applications, besides the problem of correctly recovering surface topology, sharp feature lines like ridges/ravines or crest lines [17], [27], [25] need to be identified. Concerning point sets, there have been several efforts for robust feature detection [15][5][8]. Neal [23] provided an instructive survey of such techniques. However, these approaches do not differentiate between the structural types a feature might belong to, which makes them inappropriate for data sets of arbitrary topology especially point clouds resulting from environmental scans. Some of these methods perform feature extraction based on the estimation of curvature values obtained from surfaces that are approximated based on a local point neighborhood via methods like moving least-squares (MLS) [20]. The principal curvatures and curvature directions carry important information about local surface behavior, but their estimation is numerically sensitive to noise present in the data, and many estimation methods have been proposed [11], [16], [18]. Hamann [12] [16] approximated principal curvature values based on considering a local, triangulated point neighborhood and determining a least-squares quadratic bivariate polynomial. Feature lines also produce important landmarks for constructing meshes [14], [21] or for recognizing shapes like buildings [3]. Regarding the post-processing of laser range data, a variety of techniques improving the quality can be applied [22], [26].

3 Feature Detection in LiDaR Data

Our work represents a first step toward the over-arching scientific goal of identifying and extracting regions in LiDaR data sets matching individual structural characteristics. One long-term goal was to be able to automatically differentiate from natural and man-made objects (e.g., trees and buildings). Detecting and joining objects corresponding to similar structural classes is another task we wish to accomplish. To reach these goals it is necessary to provide methods for detecting and extracting special structural features like corners, border-, crease- or ridge-lines in the form of a feature graph which can be used to perform further processing. In general, feature detection in environmental LiDaR data is a complicated task due to the nature of this kind of data. LiDaR data are generally collected either on the ground, using tripod-mounted scanners, from the air, using airplane mounted scanners or from satellites. These acquisition methods produce data sets typically exhibiting substantial noise levels since the measurement is taken from great distance. Under-sampling, occlusion and movements in the scanned environment are just a few more problems to mention.

Hence, we must assume that the resulting point cloud data in general does not provide a reliable base to directly perform feature extraction. We introduce a method that classifies points by considering whether they are part of a surface, a curve or a junction of either one. This approach allows us to adapt the feature graph extraction to the proper point class. By using stochastic and multi-scale-based means when processing the raw point cloud (especially for processing huge data sets) we introduce a rather stable approach capable of handling robustly relatively high noise levels in a data set. Our method combines the following steps:

1. Pre-processing of the 3D point cloud (sec. 3.1). This step includes an hierarchical decomposition of the point cloud into an octree-based voxel grid as well as the elimination of outlier points (sec 3.1).
2. Likelihood-based point classification depending on the characteristics of the local point neighborhood as well as computation of feature-specific values based on multiple scales (sec. 3.2).
3. Post-processing by smoothing the obtained feature values (sec. 3.4).
4. Feature graph construction (sec. 4).

3.1 Pre-processing

Hierarchical Point Cloud Decomposition: Since we are primarily dealing with environmental LiDaR data sets that are typically the result of several high-resolution scans (several million or even billion points) efficient data handling is important. Organizing the points into an octree-based voxel structure allows us to speed up frequently used operations like k-nearest neighbor search. To be able to visualize large-scale data, we follow an additional level-of-detail (LOD) approach. We modify point sets of each octree level to match a level-specific resolution and store them on disc. This approach allows us to dynamically reload point sets of a certain resolution if necessary.

Outlier Removal: Environmental LiDaR scans do not distinguish between the significant or irrelevant components; everything is scanned. In particular we under-sampling of small objects like leaves in environmental data sets including trees and other vegetation. Weyrich et al. [26] proposed a method assuming that potential outliers draw their local neighborhood from a larger vicinity than points within a well-sampled environment. Instead of using a graph-based approach we compute the mean distance of a point to its k-nearest neighbors and filter out all points whose mean distance exceeds a given threshold. This threshold has to be chosen carefully to avoid loosing important structural information.

3.2 Stochastic Point Classification

The goal of point classification is to decompose the point cloud P into subsets of points having different structural properties. We determine whether a point $p \in P$ is part of a curve- or a surface-like structure or is treated as a so-called critical point. We call those points critical points that are not part of a surface or a curve. This point type represents junctions of features lines (e.g., at corners) or borders/intersections between surfaces and/or curves. In general they are meant to represent discontinuities not associated with these structures. Hence, critical points are not suitable for directly performing methods of feature-line extraction but are used in the context of graph extraction (sec. 4) to represent underlying discontinuities.

In order to identify the type of a point p we determine the "shape" of the local point neighborhood $\mathcal{N}_r(p)$ of p. within a sphere of radius r centered at p. The radius is specified by the user and depends on the features being focused on. Determining the shape of $\mathcal{N}_r(p)$ is accomplished by comparing the distribution of $\mathcal{N}_r(p)$ in each dimension. The distribution is measured by using the eigenvalues of the covariance matrix $\mathbf{C}$ of $\mathcal{N}_r(p)$. Let $\lambda_0 \leq \lambda_1 \leq \lambda_2$ be the eigenvalues of $\mathbf{C}$. It is a well-known fact that $\mathcal{N}_r(p)$ has a spherical shape if $\lambda_0 \approx \lambda_1 \approx \lambda_2$, a disc-like shape if $\lambda_0 \ll \lambda_1 \wedge \lambda_1 \approx \lambda_2$ or a cylindrical shape if $\lambda_0 \approx \lambda_1 \wedge \lambda_1 \ll \lambda_2$. Using this information allows us to establish three point classes by defining the following sets:

$$\mathcal{P}_{cp} = \{p \in P | \lambda_0/\lambda_2 \geq \varepsilon\} \tag{1}$$
$$\mathcal{P}_{c} = \{p \in P | \lambda_1/\lambda_2 < \varepsilon\} \tag{2}$$
$$\mathcal{P}_{d} = \{p \in P | \lambda_0/\lambda_2 < \varepsilon\} \tag{3}$$

Here, $\mathcal{P}_{cp}, \mathcal{P}_c, \mathcal{P}_d$ define the points with spherical, cylindrical and disc-like neighborhood (with $\mathcal{P}_{cp} \cap \mathcal{P}_c = \emptyset$, $\mathcal{P}_{cp} \cap \mathcal{P}_d = \emptyset$ and $\mathcal{P}_c \subseteq \mathcal{P}_d$).The variable ε is a feature-specific value and determines when a point becomes a critical point. Usually ε is chosen to be $\frac{1}{2}$ but it can be adapted to meet a user's needs. For example, reducing ε increases the sensitivity of a point for being classified as a critical point, causing some points of $\mathcal{P}_c$ and $\mathcal{P}_d$ to migrate to $\mathcal{P}_{cp}$. This effect can be geometrically interpreted as extending the boundaries around a discontinuity (e.g., an intersection). This form of point differentiation implies $\mathcal{P}_c \subseteq \mathcal{P}_d$, since in some cases surfaces collapse to line-like structures. However, this characterization heavily depends on the size of the local neighborhood $\mathcal{N}_r(p)$ and would not lead to reliable shape evaluation results in presence of noise.Thus, we allow the neighborhood $\mathcal{N}_r(p)$ to vary in size by a given percentage of its original magnitude. We compute probability estimates by measuring the number of times $\mathcal{N}_r(p)$ can be assigned to one of the classes defined by (1), (2), (3) by evaluating

$$\mathcal{L}_{cp,c,d}(p) = \frac{1}{n+1} \sum_{j=0}^{n} \varphi_{cp,c,d}^{r_{min}+j\delta}(p), \tag{4}$$

$$\varphi_{cp}^s(p) = \begin{cases} 1 & \text{if } \lambda_0^s \geq \varepsilon \lambda_2^s \\ 0 & \text{else} \end{cases}, \varphi_c^s(p) = \begin{cases} 1 & \text{if } \lambda_1^s < \varepsilon \lambda_2^s \\ 0 & \text{else} \end{cases},$$
$$\varphi_d^s(p) = \begin{cases} 1 & \text{if } \lambda_0^s < \varepsilon \lambda_2^s \\ 0 & \text{else} \end{cases}$$

Here r_{min} (r_{max}) denotes the minimum (maximum) radius and n represents the number of considered neighborhoods. Hence, $\delta = (r_{max} - r_{min})/n$ and λ_i^s being the eigenvalues of $\mathbf{C}^s$ of $\mathcal{N}_s(p)$. $\mathcal{L}_{cp}, \mathcal{L}_c$ and $\mathcal{L}_d$ defining the likelihood of a point p corresponding to the shape classes defined by $\mathcal{P}_{cp}, \mathcal{P}_c$ and $\mathcal{P}_d$. We now divide the original point cloud based on the likelihood values into subsets of different character. For graph extractionwe use additional feature values capturing the strength of an underlying feature. The feature values I_{cp}, I_c and I_d for the points of the classes $\mathcal{P}_{cp}, \mathcal{P}_c$ and $\mathcal{P}_d$ are

$$I_{cp,c,d}(p) = \frac{1}{n+1} \sum_{j=0}^{n} f_{cp,c,d}^{r_{min}+j\delta}, \tag{5}$$
$$f_{cp}^s = \lambda_0^s \lambda_1^s / (\lambda_2^s)^2 \tag{6}$$
$$f_c^s = (\lambda_2^s - \lambda_1^s)\lambda_0^s / (\lambda_2^s \lambda_1^s) \tag{7}$$
$$f_d^s = \kappa^s \tag{8}$$

represent the feature values for the different point neighborhoods $\mathcal{N}_j(p)$. κ^s denotes the maximum absolute curvature of polynomial patches fitted to $\mathcal{N}_s(p)$. Since we are also interested in feature extraction from surface-related structures, using curvature is an obvious approach. The feature values I_{cp}, I_c and I_d express the strength of a feature associated with a point p by a value averaged over multiple scales. The values f^s_{cp}, f^s_c increase when the shape of $\mathcal{N}_s(p)$ approaches the ideal form of a sphere, a line and decrease otherwise. f^s_d increases as the curvature increases.

3.3 Estimating Curvature

There are several approaches concerning the approximation of curvature estimates from point clouds [11], [16], [18]. Our curvature estimation process is based on computing curvature values from polynomials fitted MLS to point neighborhoods $\mathcal{N}_r(p)$. We use the root mean squared (RMS) curvature to approximate the "feature strength". There are several other curvature measures like mean, Gaussian or maximal curvature we have considered and tested. For our purposes the RMS curvature was most appropriate and produced best results.

3.4 Post-processing

We are interested in strengthening feature values corresponding to selected features and smoothing out remaining feature values. A Gaussian like filter applied on the local point neighborhood $\mathcal{N}_r(p)$ ensures that noise caused by small artifacts is eliminated.

4 Generating the Feature Graph

The structural features are detected and extracted following the idea of [8], [29] and [25] of identifying initial seed nodes and performing an edge-growing approach to connect related nodes. The seed nodes for a graph are represented by all points having a minimal type probability. It remains difficult to define connections between seed nodes such that the resulting graph reflects the underlying feature-lines. Most methods start by connecting seed nodes by proximity. However, this can cause a variety of branches inducing perturbations in further propagation. Therefore, controlling the propagation direction plays an important role.

Several extraction methods determine the derivatives of extracted feature values, like curvature, to identify and follow the direction of its minimal descend. Since these values vary considerably due to noise this differential approach would lead to poor results. Instead, we propose defining the propagation direction of a node by applying PCA to a set of surrounding nodes. This approach is more stable assuming that the local environment of the actual node is of sufficient size. In contrast, critical points are not subject to this graph extraction procedure since they represent discontinuities. These points are used to establish critical nodes to recover junctions of feature lines. The final step consists of graph simplification aiming at a simpler visual representation. We summarize the basic principles of our method.

4.1 Selecting Seed Nodes

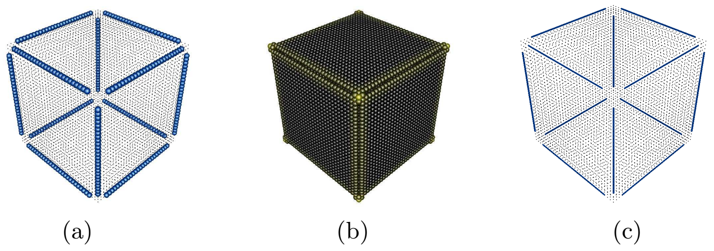


Fig. 1. Principle of graph extraction for a simple test cube data set (3000 points): (a) Initial seed nodes (blue), (b) corresponding critical nodes (black), (c) extracted edges after propagation.

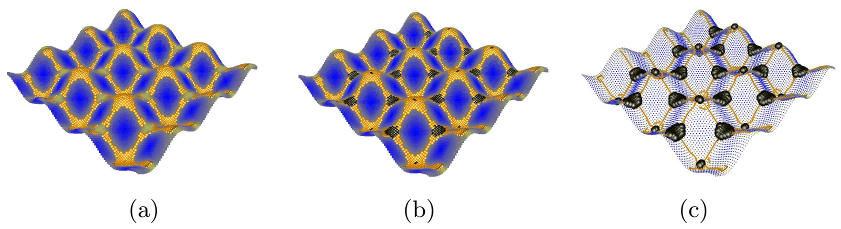


Fig. 2. Principle of graph extraction for surface-related features using a curved sheet data set (10000 points): (a) Initial seed nodes (orange) of the underlying surface (blue), (b) seed nodes with corresponding critical nodes (black), (c) critical nodes together with the extracted edges after propagation.

The seed nodes defining the basis of the individual graphs are identified by using the probability values computed according to the equations in sec. 3.2. A graph $\mathcal{G} = (V, E)$ formally consists of a set of nodes V and set of edges $E \subseteq V \times V$ connecting the nodes of V. The initial graph for representing curve-related features is defined by $\mathcal{G}_c^0 = (V_c, \emptyset)$ with

$$V_c = \{p \in P | \mathcal{L}_c(p) = max(\mathcal{L}_{cp}(p), \mathcal{L}_c(p), \mathcal{L}_d(p))\}. \tag{9}$$

Figure 1a) shows an example for identified seed nodes for a simple test data set. The graph for representing surface-related features is set up differently.

The user specifies curvature-related thresholds σ_{min} and σ_{max} restricting the strength of the features to be extracted, balancing critical and feature-specific nodes among the points of $\mathcal{P}_d$. For example, σ_{min} separate features like creases and non-creases appearing on a surface. The parameter σ_{max} controls the transition between creases and corners. The initial graph representing surface-related features is $\mathcal{G}_d^0 = (V_d, \emptyset)$ with

$$V_d = \{p \in P | \mathcal{L}_d(p) = max(\mathcal{L}_{cp}(p), \mathcal{L}_c(p), \mathcal{L}_d(p)) \wedge \sigma_{min} < I_d(p) \leq \sigma_{max}\}. \quad (10)$$

Fig. 2a) shows an example concerning surface-related features. The critical nodes corresponding to $\mathcal{G}_c^0$ and $\mathcal{G}_d^0$ are defined as

$$CV_c = \{p \in P | \mathcal{L}_{cp}(p) = max(\mathcal{L}_{cp}(p), \mathcal{L}_c(p), \mathcal{L}_d(p))\} \quad (11)$$
$$CV_d = CV_c \cup \{p \in V_d | \sigma_{max} < I_d(p)\} \quad (12)$$

with $\mathcal{L}_{cp}(p), \mathcal{L}_c(p), \mathcal{L}_d(p)$ determined according (4)Figs. 1b) and 2b) show an example of extracted critical nodes for the cube data set and the sheet test set. The nodes of the graph $\mathcal{G}_c^0$ and $\mathcal{G}_d^0$ are not yet connected. A subsequent edge propagation step, discussed in the next section, produces a first approximation of the underlying features.

4.2 Edge Propagation

Edge propagation is performed sequentially starting from the nodes having the highest feature values. Propagation is the process of connecting two nodes of the graph $\mathcal{G} = (V, E)$ by an edge. Principally, each seed node $v \in V$ is allowed to propagate into two directions $\boldsymbol{d}$ and $-\boldsymbol{d}$. We determine $\boldsymbol{d}$ by applying PCA to the local node neighborhood $\mathcal{M}_r(v) = \{w \in V \cup CV | \|\boldsymbol{v} - \boldsymbol{w}\| \leq r\}$ with r being the user-specified neighborhood size. The vector $\boldsymbol{d}$ is the eigenvector with the largest eigenvalue of the covariance matrix of $\mathcal{M}_r(v)$. Provided that no other node u is connected to v with $\boldsymbol{d} \cdot (\frac{\boldsymbol{u}-\boldsymbol{v}}{|\boldsymbol{u}-\boldsymbol{v}|}) > 0$, v is connected to $w \in \mathcal{M}_r(v)$ whereas $w = min_{w \in \mathcal{M}_{r,\boldsymbol{d}}}(v) \, (\|proj_{\boldsymbol{d}}(\boldsymbol{w} - \boldsymbol{v}) \cdot \boldsymbol{d} + \boldsymbol{v} - \boldsymbol{w}\| / \|\boldsymbol{w} - \boldsymbol{v}\|)$.

The nodes $w \in \mathcal{M}_{r,\boldsymbol{d}}(v) \subseteq \mathcal{M}_r(v)$ satisfy the inequality $\left(\frac{\boldsymbol{w}-\boldsymbol{v}}{\|\boldsymbol{w}-\boldsymbol{v}\|}\right) \cdot \boldsymbol{d} > 0$. To prevent unwanted edges all disconnected nodes $w \in \mathcal{M}_r(v)$ lying closer to v than the most distant connected neighbor are forbidden to propagate. Moreover, connections establishing acute angles with $\boldsymbol{d}$ are forbidden. Examples of resulting graphs are shown on Figs. 1(c) and 2c). Rather than using a minimal spanning graph as proposed by Pauly et al. [29], this approach provides greater consistency between edges and feature directions.

4.3 Connecting Critical Nodes

We have to connect the critical nodes to the already established edges of $\mathcal{G}$. Assume CV to be the set of critical nodes defined by (11) and (12). To recover junctions the nodes of CV have to be connected or merged with nodes of $\mathcal{G}$.We start clustering the nodes of CV by grouping neighbored critical nodes. The resulting clusters are defined by $\mathcal{D} = \{\dots, D_i, \cdots\}$ with cluster $D_i = \{w \in CV| \;\; v \in D_i \wedge \|\boldsymbol{v} - \boldsymbol{w}\| < r\}$ and $D_i \cap D_j = \emptyset$ for $i \neq j$. The initial cluster set is given by $\mathcal{D}^0 = \{\cdots, \{v\}, \{w\}, \cdots\}$ with $v, w \in V$ and $v \neq w$. We proceed by determining the nodes $d \in D_i$ of greatest feature values. These nodes represent the centroids of $D_i \in \mathcal{D}$. Depending on the user-specified radius r and the shape of D_i a cluster can have several centroidal nodes $d \in D_i$.

Next, we connect all nodes $v \in V$ to surrounding critical nodes $w \in CV$ with $\|\boldsymbol{v} - \boldsymbol{w}\| < r$. This step is performed in a way forcing the graph not to develop unwanted branches. In a subsequent step all nodes $w \in D_i$ connected to a node $v \in V$ are merged with the centroidal node $d \in D_i$. Should there exist more than one centroidal node in a cluster D_i we force w to connect to the centroidal node closest to w. Finally, we delete all unconnected critical nodes producing the final graph representation. Fig. 2(c) shows an example with the critical node cluster shown in black. The remaining connected critical/centroidal nodes are shown in Fig. 3(c).

4.4 Graph Simplification

We note that $\mathcal{G}_c$ and $\mathcal{G}_d$ possess certain unwanted structures like short branches, loops etc. To get rid of these artifacts we have developed the following framework.

Graph Simplification: In our context, simplification refers to the process of collapsing nodes that are closer to each other in order to reduce complexity of an extracted graph. We collapse nodes v and w of a graph $\mathcal{G}$ if the following criteria are fulfilled:

a) $\|\boldsymbol{v} - \boldsymbol{w}\| \leq r$
b) $(v, w) \in E$ and $\nu(v) > 1$ and $\nu(w) > 1$

Here, $\nu(v)$ defines the number of edges incident to v. Since $\mathcal{G}_c$ and $\mathcal{G}_d$ are intended to reflect structural features we force the position of the resulting graph node to agree with the position of the node possessing the highest feature value. In case of participation of critical nodes the rules are adapted to preserve critical nodes. The entire collapse procedure is performed sequentially in accordance with ascending distance between the node pairs.

Additional Pruning: The remaining graph still exhibits features irrelevant or unwanted for visual exploration. To reduce visual clutter and direct the user to the regions of interest we prune small branches by cutting off all edges $e = (v, w)$ of $v, w \in \mathcal{G}$ with $\|\boldsymbol{v} - \boldsymbol{w}\| < r$ and valence $\nu(v) = 1 \wedge \nu(w) \neq 2$ which are not of immediate interest.

Smoothing: Finally, we enhance the visual appearance of the final graph set $\mathcal{G}$ by applying common graph smoothing.

5 Results

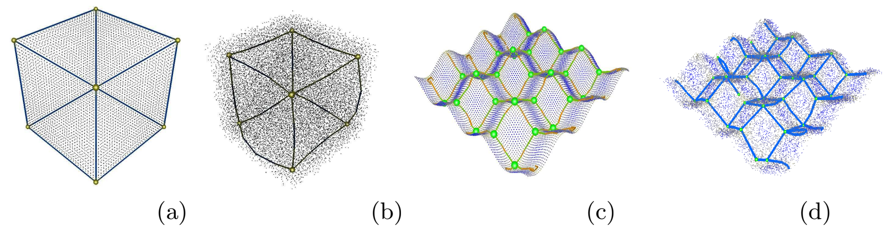


Fig. 3. Results of graph extraction for the cube data set (a) without and (b) with 5% noise. Images (c) and (d) show the resulting graphs for the curved sheet data set without and with 5% noise.

Our method provides a highly effective means for exploring the underlying structural characteristics in environmental LiDaR point data sets. The graph-based approach represents user-specified features of two different types in an efficient and reliable way, driven by a few pre-defined input values. Most important are the radius r of the feature size and the values σ_{min} and σ_{max} confining the type of the considered surface-related features. The radius r has to be chosen in a way that noise does not interfere with the classification process but also that the extraction of structural information is maintained. Generally, σ_{min} and σ_{max} are less crucial to determine. Once the radius is found the remaining feature extraction can be performed rather fast in an interactive way. Our method distinguishes features of three different types and allows classification of points in terms of probability. Moreover, we are able to identify junctions of feature lines not only by means of graph processing but using implicit information inherited from the initial point cloud. Structural discontinuities are represented in the final graph by critical nodes.

Our multi-scale approach also reduces the influence of noise. It produces reliable results up to a noise level of 5-8%. Fig. 3 shows the resulting graph related to curve-like structures without and with 5% noise relative tothe diagonal of the smallest bounding box containing the given point cloud. The examples show that all resulting graphs of the test data set reflect the correct underlying topology. Since determining likelihood and feature values is computationally expensive we de-couple this step from the graph extraction process and perform it in a pre-processing step. This allows the user to extract feature graphs in an nearly interactive manner.

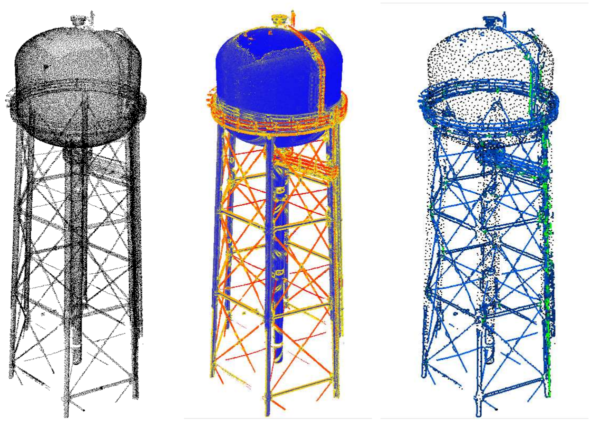
We have applied our method to several other environmental data sets. The data was collected using a LiDaR scanner mounted on a tripod, a method that enables fully three-dimensional scans of engineered structures. For example, Fig. 4(a) through Fig. 4(c) show the results of processing a more complex structure. In Fig. 4(a) we see the original point cloud of a water tower consisting of approximately 1.7 million points. Fig. 4(b) shows the computed feature values regarding curve-related features for each point. The extracted feature graph is presented in Fig. 4(c). More detailed parts of the graph are presented in Fig. 5. Fig. 5(a) shows a relative complex under-sampled object which is part of the tower. In this example the underlying feature lines and junctions were almost completely recovered. The second object shown in Fig. 5(b) is a crosspiece of the main pole of the lower part of the tower having mounted stiffeners. The extracted feature graph exhibits critical nodes colored in green representing discontinuities between surface- and curve-like structures.

The environmental point cloud shown in Fig. 6 is a part of the San Andreas Fault in California, USA. Since the scan was performed airborne the resulting point cloud contains a relatively higher level of noise. The edges of the extracted surface-related graph depicted in red represent the underlying ridges of the landscape. The critical nodes depicted as green points represent either underlying discontinuities or features having curvature values which exceed the user-specified limit σ_{max}. In this example the parameter values were chosen to capture ridges associated with relatively high feature/curvature values. Those parts of the landscape with high curvature values (tapered ridges or acute hills) are represented by critical nodes. For this example we chose $\sigma_{min} = 0.2, \sigma_{max} = 1.5$ and $r \approx 0.0046\%$ of the length of the data set diagonal of the smallest bounding box containing P.

In order to allow the user to perform the feature extraction interactively we have divided the whole approach into two steps. We provide a computational expensive pre-processing step concerning the point classification and the actual graph extraction step. The time complexity for the point classification is estimated to be $O(k^2 \cdot n \cdot log(n))$ (with n number of input points, k the average neighborhood size). The complexity of the graph extraction is assumed to be $O(m \cdot log(m) \cdot n)$ (with m being the number of identified seed nodes). This has been validated by our observations. For example, the point classification performed at the finest resolution for the water tower point cloud (see Fig. 4(a)) having about 1.7 million points has last 3h29m. The subsequent graph extraction was performed in approximately 116 secs. Computing the point classification for the geographic data set in Fig. 6 was completed after 72 minutes. The final graph was constructed after 29 secs. These results confirm that our approach holds great promise for feature detection and extraction for LiDaR data processing, analysis and understanding.

6 Conclusions

We have introduced a novel approach for extracting and visualizing features from LiDaR data sets. Our examples demonstrate the benefits of this method for extracting feature graphs from point clouds representing surface components like ridges and creases as well as curve-like components. As our method was designed as an interactive, user-controlled method offering also a high degree of automation, one specific strength is the ability for a user to influence the



(a) (b) (c)

Fig. 4. (a) original point cloud of a water tower consisting of 1.7 million points; (b) color coded feature values of each point regarding curve-related features(blue represents low, red high feature values); (c) corresponding feature graph (blue lines) for $\varepsilon = 0.5$ and $r \approx 0.038\%$ of the bounding box diagonal.

analysis process and obtain instant visual feedback. This work is fundamental for future research, for example to detect and measure deformation of environmental structures such as buildings and bridges exposed to terrestrial influences. Future work will be directed at feature-based segmentation of terrestrial data classifying objects of different type, such as trees, buildings, streets, etc. One next goal is to be able to extract topological information from landscapes by extracting and comparing ridge lines from multiple scans taken at different times to identify deformations by a comparative visualization method.

Acknowledgements

This work was supported by the DFG's International Research Training Group (IRTG) 1131 at the University of Kaiserslautern and the Center for Mathematical and Computational Modeling (CM)^2 It was also supported in part by the W.M. Keck Foundation through the UC Davis Center for Active Visualization in the Earth Sciences (KeckCAVES), Department of Geology. We thank the members of the Visualization and Computer Graphics Research Group at the Institute for Data Analysis and Visualization (IDAV) at UC Davis. We thank Dr. Gerald Bawden of the US Geological Survey for sharing tripod LiDaR data.

References

1. M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, 2003.
2. M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. In *Conference on Visualization*, pages 21–28. IEEE Computer Society, 2001.
3. A. Alharthy and J. Bethel. Detailed building reconstruction from airborne laser data using a moving surface method. In *20th Congress of International Society for Photogrammetry and Remote Sensing*, pages 213–218, 2004.

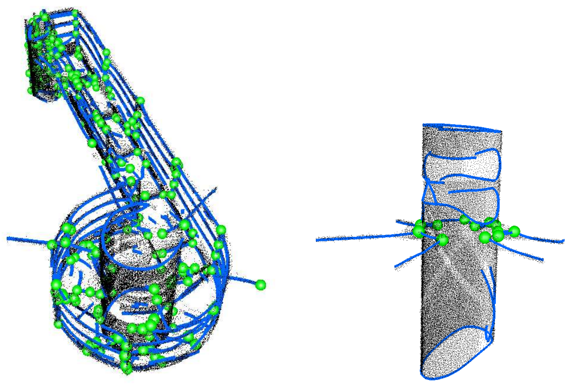


Fig. 5. Closeup of selected parts of the extracted graph of Fig. 5(c).

4. N. Amenta, S. Choi, and R. K. Kolluri. The power crust. In *6th ACM symposium on Solid Modeling and Applications*, pages 249–266. ACM Press, 2001.
5. A. Belyaev and E. Anoshkina. Detection of surface creases in range data. In *Mathematics of Surfaces*, pages 50–61, 2005.
6. P. Benkö and T. Várady. Direct segmentation of smooth, multiple point regions. In *Geometric Modeling and Processing - Theory and Applications*, page 169. IEEE Computer Society, 2002.
7. P. Benkö and T. Várady. Segmentation methods for smooth point regions of conventional engineering objects. *Computer-Aided Design*, 36(6):511–523, 2004.
8. J. I. Daniels, L. K. Ha, T. Ochotta, and C. T. Silva. Robust smooth feature extraction from point clouds. In *IEEE International Conference on Shape Modeling and Applications, 2007. SMI '07.*, pages 123–136, June 2007.
9. T. K. Dey and S. Goswami. Provable surface reconstruction from noisy samples. In *20th Annual Symposium on Computational Geometry*, pages 330–339. ACM Press, 2004.
10. T. K. Dey and J. Sun. An adaptive mls surface for reconstruction with guarantees. In *3rd Eurographics Symposium on Geometry Processing*, pages 43–52, 2005.
11. I. Douros and B. F. Buxton. Three-dimensional surface curvature estimation using quadric surface patches. In *Scanning*, 2002.
12. B. Hamann. Modeling contours of trivariate data, Mathematical Modeling and Numerical Analysis. In *Modelisation Mathematique et Analysis Numerique*, vol. 26(1), Gauthier-Villars, France, pages 51–75, 1992.
13. M. Eck and H. Hoppe. Automatic reconstruction of b-spline surfaces of arbitrary topological type. In *ACM Siggraph*, pages 325–334, 1996.
14. S. Fleishman, D. Cohen-Or, and C. Silva. Robust moving least-squares fitting with sharp features. In *ACM Siggraph*, pages 544–552, 2005.
15. S. Gumhold, X. Wang, and R. MacLeod. Feature extraction from point clouds. In *10th International Meshing Roundtable*, pages 293–305, 2001.
16. B. Hamann. Curvature approximation for triangulated surfaces. *Geometric Modelling, Computing Suppl. 8*, pages 139–153, 1993.
17. K. Hildebrandt, K. Polthier, and M. Wardetzky. Smooth feature lines on surface meshes. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing*, pages 85–90. Eurographics Association, 2005.
18. P. Krsek, G. Lukács, and R. Martin. Algorithms for computing curvatures from range data. In *The Mathematics of Surface VIII*, pages 1–16. Information Geometers. Winchester, UK, 1998.
19. C. Lange and K. Polthier. Anisotropic smoothing of point sets. *Computer Aided Geometric Design*, 22:680–692, 2005.
20. D. Levin. The approximation power of moving least-squares. *Mathematics of Computation*, 67(224):1517–1531, 1998.
21. B. Mederos, L. Velho, and L. H. de Figueiredo. Moving least squares multiresolution surface approximation. In *Sibgraphi*, 2003.

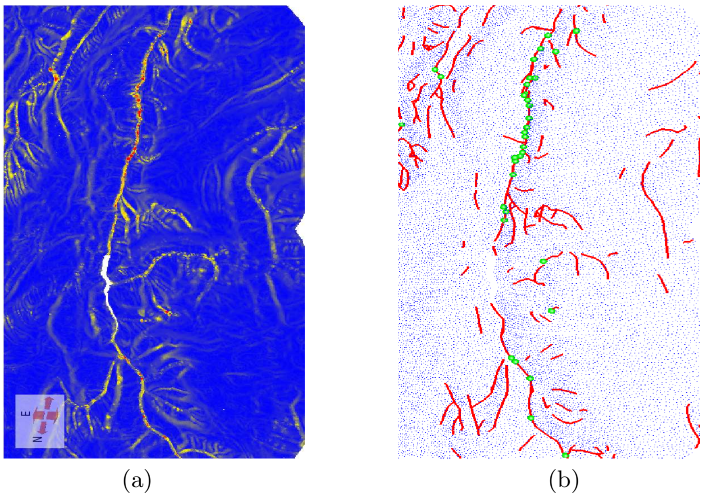


Fig. 6. Scan of San Andreas fault in California, USA, on a length of 6 kilometers. (a) presents the corresponding feature/curvature values (blue means low, red equals high curvature value). (b) shows the corresponding surface-related feature graph with feature lines depicted in red, and critical nodes as green points.

22. L. V. B. Mederos and L. H. de Figueiredo. Robust smoothing of noisy point clouds. In *SIAM Conference on Geometric Design and Computing*, 2003.
23. A. Nealen. An as-short-as-possible introduction to the least squares, weighted least squares and moving least squares methods for scattered data approximation and interpolation. In *Technical Report, TU Darmstadt*, 2004.
24. M. Pauly, R. Keiser, L. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. In *ACM Siggraph*, pages 641–650, 2003.
25. G. Stylianou and G. Farin. Crest lines for surface segmentation and flattening. *IEEE Transaction On Visualization and Computer Graphics*, 10(5):536–544, Sept. 2004.
26. T. Weyrich, M. Pauly, R. Keiser, S. Heinzle, S. Scandella, and M. Gross. Post-processing of scanned 3d surface data. In *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, pages 85–94, 2004.
27. S. Yoshizawa, A. Belyaev, and H.-P. Seidel. Fast and robust detection of crest lines on meshes. In *ACM Symposium on Solid and physical modeling*, pages 227–232. ACM Press, 2005.
28. M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop 3d: an interactive system for point-based surface editing. In *ACM Siggraph*, pages 322–329, 2002.
29. M. Pauly, R. Keiser, and M. Gross. Multi-scale Feature Extraction on Point-sampled Surfaces. In *Proceedings of Eurographics 2003*, pages 281–289, 2003.