EUROGRAPHICS 2003 / P.Brunet and D. Fellner
(Guest Editors)

Volume 22 (2003), Number 3

Multi-scale Feature Extraction on Point-sampled Surfaces

Mark Pauly Richard Keiser Markus Gross

ETH Zürich

Abstract

We present a new technique for extracting line-type features on point-sampled geometry. Given an unstructured point cloud as input, our method first applies principal component analysis on local neighborhoods to classify points according to the likelihood that they belong to a feature. Using hysteresis thresholding, we then compute a minimum spanning graph as an initial approximation of the feature lines. To smooth out the features while maintaining a close connection to the underlying surface, we use an adaptation of active contour models. Central to our method is a multi-scale classification operator that allows feature analysis at multiple scales, using the size of the local neighborhoods as a discrete scale parameter. This significantly improves the reliability of the detection phase and makes our method more robust in the presence of noise. To illustrate the usefulness of our method, we have implemented a non-photorealistic point renderer to visualize point-sampled surfaces as line drawings of their extracted feature curves.

1. Introduction

Point-sampled surfaces have emerged in recent years as a versatile representation for geometric models in computer graphics. The surface of a 3D object is described by a set of sample points without further topological information such as triangle mesh connectivity or a parameterization. Reducing the representation to the essentials, i.e. the geometric position of the sample points, is particularly useful when dealing with large data sets generated by modern acquisition devices [15]. To display such models, numerous point-based rendering systems have been developed, e.g. [20, 21, 25, 1]. Apart from acquisition and rendering, a variety of geometry processing applications have been introduced recently [18, 19, 26] that demonstrate the versatility of points as a geometric modeling primitive.

In this paper, we present a new method for detecting and extracting line-type features on point-sampled surfaces. This type of information can serve as input for many processing applications such as meshing, model segmentation, or anisotropic fairing. Feature lines can also be used for visualization to enhance the semantics of renditions of 3D objects. In Section 4 we will show how artistic line drawings of point-sampled surfaces can be created using the extracted feature curves.

Features are usually defined as entities of an object that are considered important by a human for an accurate description of the object. This definition is highly subjective, however, and very difficult to express in algorithmic form. Our goal was to design a feature extraction algorithm that requires no additional semantic information about the object. Also, our method should be semi-automatic, i.e. only require the user to specify a few thresholding parameters. Additional interaction with the object, such as setting seed points or guiding feature movement, is not necessary. We therefore base our feature definition on low-level information using a statistical operator that measures local surface variation. This operator classifies points according to the likelihood that they belong to a feature. To improve the robustness and reliability of the classification stage, we apply this operator at multiple scales, which allows us to measure the persistence of a feature [4]. Additionally, multi-scale classification provides further structural information per classified point, e.g. the characteristic scale at which a feature is most prominent.

We concentrate on line-type features. These are probably the most important features for surfaces, which are often composed of patches that are framed by feature lines. A feature line approximately passes along a ridge of maximum inflection, which is adequately captured in our surface variation estimate.

We believe that low-level feature extraction methods such as ours always require some user feedback, in particular for our example application of an artistic line-drawing renderer. To obtain visually pleasing renditions, the user has to adjust the various parameters of our feature extraction method until she is satisfied with the result. We therefore

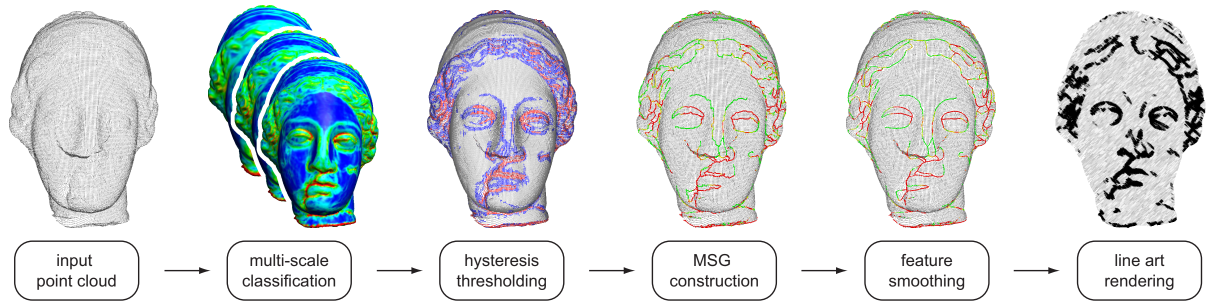

Figure 1: *Feature extraction pipeline.*

put particular emphasis on efficiency, allowing interactive control in a low-latency feedback loop.

We should note that additional knowledge about the object, e.g. knowing that the model is a laser-range scan of a human head, can of course be utilized to design more accurate feature extraction algorithms. Such information is very application-dependent, however, and limits the scope of suitable data sets considerably.

1.1. Previous Work

Feature extraction is a well-studied research area in many scientific fields, including computer vision, medical imaging and computational fluid dynamics. Most of the past research efforts concentrated on data defined in a Euclidean domain, e.g. images, volume data or flow fields. Feature extraction on surfaces, i.e. two-manifolds embedded in 3-space, has gained less attention, but is important in many fields such as range data analysis or reverse engineering. In these applications, the input data is typically a large point set acquired by some 3D scanning device. Thus feature extraction directly on the point cloud is very attractive, as it can be used to support early processing steps such as surface reconstruction or adaptive decimation.

Our method combines and extends existing techniques from different research fields. In particular, we integrate recent results from image processing, discrete geometric modeling and scale-space theory.

Canny [2] introduced an optimal filter for step-edge detection in images. He found that there exists a natural uncertainty principle between detection and localization performance and derives operators that are optimal at any scale. We use his method of hysteresis thresholding during feature classification.

Hubeli and Gross [8] introduced a multiresolution framework for feature extraction on triangle meshes. Based on various classification operators they identify a set of feature edges and use thinning to extract line-type features from the set of selected edges.

Feature sensitive meshing of surfaces defined as the zero-set of a discrete 3D distance function has been presented in [13]. Here features are detected using the width of the normal cone spanned by adjacent vertices as a measure of surface curvature. The focus in this work is on avoiding the aliasing artefacts generated by the standard marching cubes algorithm and the authors report that their simple feature classification method yields good results for this purpose.

Geometric snakes have been used in [14] to extract feature lines in triangle meshes based on normal variation of adjacent triangles. This system requires the user to specify an initial feature curve, which is then evolved under internal and external forces and re-projected onto the surface using a local parameterization.

Gumhold et al. [5] presented a feature extraction method for point clouds that is similar to ours. They also use covariance analysis for classification and compute a minimum spanning graph of the resulting feature nodes. We extend this scheme using a multi-scale classification that allows robust feature extraction for noisy surfaces. By modeling the extracted feature lines using snakes, we also gain more control on the smoothness as compared to the spline fitting method used in [5].

1.2. Overview

Figure 1 gives an overview of our feature extraction pipeline. Given an unstructured point cloud $P = \{\mathbf{p}_i \in \mathbb{R}^3\}$ that approximates some two-manifold surface S, our algorithm starts by classifying points according to the likelihood that they belong to a feature (Section 2). This is done using a multi-scale approach that assigns weights ω_i to each $\mathbf{p}_i \in P$. After thresholding these weights, we compute a minimum spanning graph of the remaining sample points (Sections 3.1 and 3.2). Each separate component of the graph is modeled by a snake, an energy-minimizing spline that is attracted to the feature vertices. Using Euler integration, we can smooth the feature lines represented by the snakes, while maintaining a close connection to the underlying surface (Section 3.3). The extracted feature lines can then be visualized using non-photorealistic point-based rendering (Section 4).

2. Feature Classification

The first stage of our feature extraction pipeline is classification. For each point $\mathbf{p}_i \in P$ we compute a weight ω_i that measures the confidence that $\mathbf{p}_i$ belongs to a feature. Our feature classification is based on surface variation estimation using covariance analysis of local neighborhoods. We will show how this statistical approach can be incorporated into a scale-space framework that allows feature classification at multiple scales.

2.1. Scale Space

Since their introduction in 1983 [24], scale-space representations have been studied extensively in the context of feature detection for images. The fundamental idea is to model a signal $f : \mathbb{R}^d \to \mathbb{R}$ at different scales as $L : \mathbb{R}^d \times \mathbb{R}_+ \to \mathbb{R}$, where L is defined as a convolution of f with Gaussian kernels G of varying width t:

$$L(\mathbf{x}, t) = G(\mathbf{x}, t) \otimes f(\mathbf{x}) \qquad (1)$$

with $\mathbf{x} \in \mathbb{R}^d$ (see Figure 2). Equivalently, L can be defined as the solution of the diffusion equation

$$\frac{\partial L}{\partial t} = \frac{1}{2}\nabla^2 L\,. \qquad (2)$$

Given a scale-space representation $L(\mathbf{x}, t)$, we can then apply a classification operator to measure the desired function properties, e.g. curvature, at different scales.

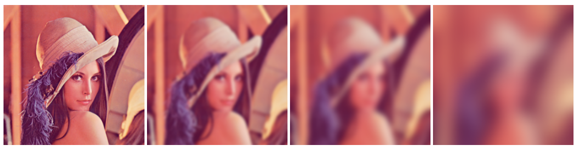

Figure 2: *Scale-space representation of an image with increasing scale factor from left to right.*

To transfer these concepts to point-sampled surfaces, we need to specify a suitable classification operator for manifold geometry, e.g. using curvature estimation from polynomial fits. Additionally, we have to define an appropriate Gaussian smoothing method. Choices include Taubin's iterative Laplacian smoothing [23], Desbrun et. al.'s curvature flow [3], or Kobbelt's variational fairing [12], which can all be generalized to point-sampled surfaces. This approach has some drawbacks, however. To compute a complete multi-scale classification, we have to apply the curvature estimation for each sample point at each scale. Since fitting a local polynomial is a fairly expensive operation, the computational overhead quickly becomes excessive for large models or high resolutions of the scale axis. Another problem is that the smoothing methods often produce surface deformation artefacts such as volume shrinkage, caused by inevitable distortions in the parameterization. Thus by applying these approximative Gaussian smoothers, surface curvature can even be increased as illustrated on the ears of the bunny in Figure 3.

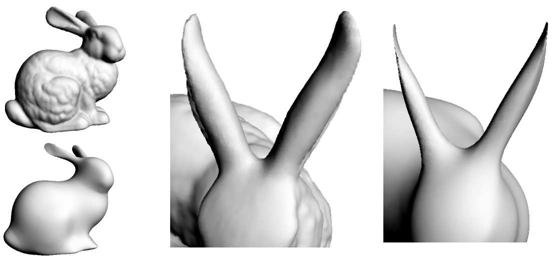

Figure 3: *Volume shrinkage leads to increased curvature at the bunny's ears for iterative Laplacian smoothing.*

2.2. Multi-Scale Surface Variation

As discussed above, the classical multi-scale method is difficult to transfer from the functional setting to discrete manifold geometry. Therefore, we use a different approach based on a statistical operator on local neighborhoods. We will show that the size of these neighborhoods can be used as a discrete scale parameter.

Surface Variation. Various researchers have used principal component analysis of local point neighborhoods to estimate local surface properties, such as curvature, on point-sampled surfaces [5, 22, 19]. We define a local neighborhood as the index set N_p of the k-nearest neighbors of a sample point $\mathbf{p} \in P$. Let $\bar{\mathbf{p}}$ be the centroid and $\mathbf{C}$ the 3×3 covariance matrix of N_p defined as

$$\mathbf{C} = \frac{1}{k}\begin{bmatrix} \mathbf{p}_{i_1} - \bar{\mathbf{p}} \\ \ldots \\ \mathbf{p}_{i_k} - \bar{\mathbf{p}} \end{bmatrix}^{\mathrm{T}} \cdot \begin{bmatrix} \mathbf{p}_{i_1} - \bar{\mathbf{p}} \\ \ldots \\ \mathbf{p}_{i_k} - \bar{\mathbf{p}} \end{bmatrix}, i_j \in N_p\,. \qquad (3)$$

In [19], Pauly et. al. introduced surface variation $\sigma_n(\mathbf{p})$ as

$$\sigma_n(\mathbf{p}) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}, \qquad (4)$$

where the λ_i are the eigenvalues of $\mathbf{C}$ with $\lambda_0 \le \lambda_1 \le \lambda_2$. Note that the surface variation is invariant under rescaling and that the total variation is given as

$$\sum_{i \in N_p} |\mathbf{p}_i - \bar{\mathbf{p}}|^2 = \lambda_0 + \lambda_1 + \lambda_2\,. \qquad (5)$$

Figure 4 illustrates surface variation for different neighborhood sizes on the Max Planck model.

Multi-scale Variation Estimation. To apply the concepts of scale-space for our feature classification using surface variation $\sigma_n(\mathbf{p})$, we observe that the size n of the neighborhood of a sample $\mathbf{p}$ can be used as a discrete scale parameter. In fact, increasing the size of the local neighborhood is similar to applying a smoothing filter. This becomes intuitively clear if we look at the way the covariance matrix is defined as sums of squared distances from the neighbor-

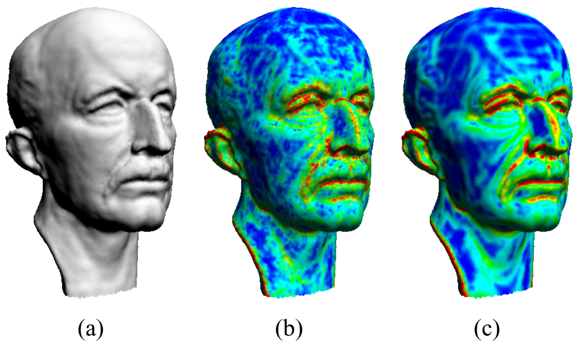

Figure 4: *Surface variation on the Max Planck bust. (a) original, (b) color-coded variation* σ_{10} *(blue corresponds to low values and red to high values), (c) variation* σ_{50}.

hood's centroid. If we increase the neighborhood size, each individual point contributes less to the surface variation estimate. Hence high-frequency oscillations are attenuated, analogous to standard low-pass filter behavior.

Comparison to Gaussian smoothing. To evaluate the multi-scale variation estimation, we compare our method with the traditional multi-scale approach using Gaussian filter kernels (see Section 2.1). We use a terrain model defined as a regularly sampled height-field. Since this surface can be parameterized without distortion, we can compute coarse scale representations using standard Gaussian filtering for grids. As illustrated in Figure 5, the classification on the smoothed surfaces using surface variation of smaller neighborhood sizes corresponds very well to the output of the variation estimate on the rougher surfaces with bigger neighborhood sizes. Even though some quantitative deviations are observable, in terms of feature classification both methods are almost equivalent and thus interchangeable. Note also that multi-scale surface variation can be computed very efficiently as described in detail in the Appendix.

2.3. Determining Feature Weights

Given a multi-scale variation estimate, we can let the user specify the appropriate scale of interest and simply use the variation estimate of that scale as our feature weights ω_i. Thus be selecting a single parameter, the scale, the user can decide whether fine-scale or coarse-scale features should be extracted.

Automatic Scale Selection. However, finding the right scale parameter is often difficult and this is why methods for automatic scale selection have been of interest in many fields. Lindeberg pioneered these techniques for functional scale-space representations [16]. His principle for scale selection states that the scale level at which some normalized derivative operator assumes a local maximum, reflects a characteristic length of the corresponding structure in the data. As illustrated in Figures 6 and 7 this principle can easily be transferred to the scale-space representation introduced above. To determine the feature weights, we look for the strongest local maximum in the surface variation at all points across the scale axis. The points on the ear, nose and leg in Figure 7, for example, have been classified as feature points because they exhibit a distinct local maximum in surface variation, while the point on the back shows no such characteristic.

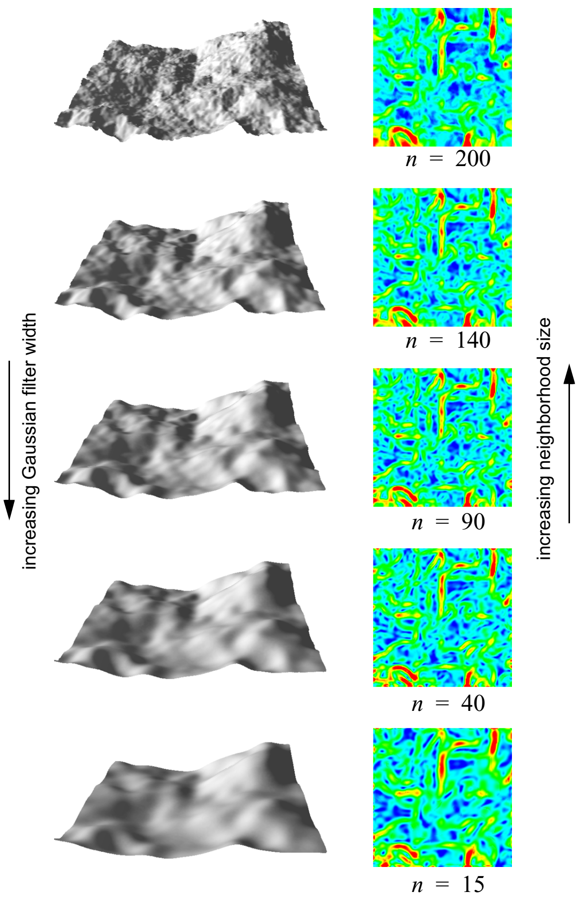
increasing Gaussian filter width
increasing neighborhood size
$n = 200$
$n = 140$
$n = 90$
$n = 40$
$n = 15$

Figure 5: *Multi-scale surface variation on height field data. Left column: Scale-space representation of a terrain model with increasing smoothness from top to bottom. Right column: Corresponding surface variation with increasing neighborhood size from bottom to top.*

Persistence. Instead of using a single local maximum for classification, we can also look at the number of times that the surface variation exceeds a certain threshold $\sigma_{\max}$. For a point $\mathbf{p}_i \in P$ and a neighborhood size n we define

$$\Omega(\mathbf{p}_i, n) = \begin{cases} 1 & \sigma_n(\mathbf{p}_i) > \sigma_{\max} \\ 0 & \sigma_n(\mathbf{p}_i) \le \sigma_{\max} \end{cases}, \tag{6}$$

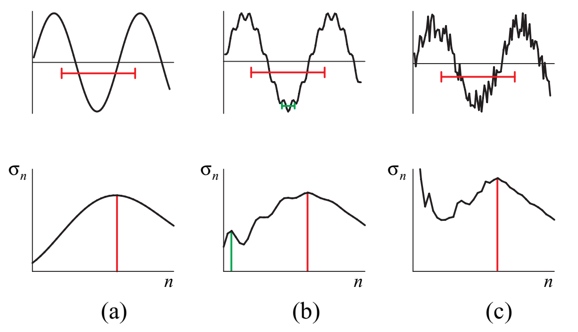

Figure 6: *Automatic scale selection for a 1D signal. The top row shows the signal, the bottom row the variation at the central point as a function of neighborhood size. The local maxima are indicated as vertical lines and the characteristic lengths as horizontal bars. In (a) a simple sine curve is analyzed, while in (b) another high-frequency sine wave has been added. Note how the different frequencies are reflected in the distinct local maxima of the variation estimate. In (c) random noise has been added to the signal. If we look at the coarser scales, i.e. larger n, we can still faithfully recover the feature.*

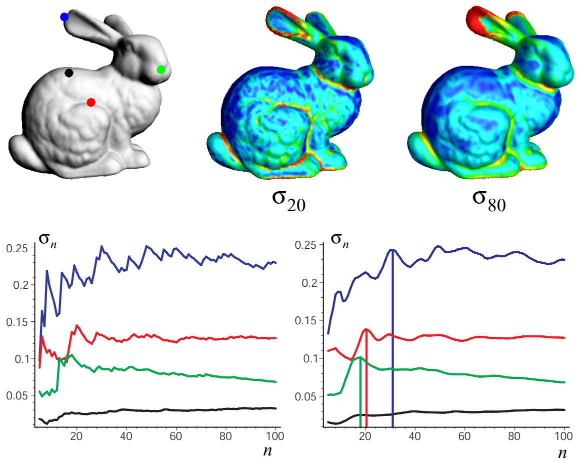

Figure 7: *Multi-scale surface variation. The left diagram shows values of σ_n for different points on the bunny as a function of neighborhood size n. To avoid instabilities in the detection of local maxima, these curves have been pre-smoothed as shown on the right. The vertical lines show the scale of the extracted maxima in surface variation.*

so that the corresponding feature weight ω_i is given as

$$\omega_i = \sum_n \Omega(\mathbf{p}_i, n) . \tag{7}$$

Thus this counting approach measures the *persistence* of a feature over all scales. (see also [4]).

Surface Boundaries. When dealing with non-closed surfaces, we extend the feature classification method to include points that lie on the surface boundary. To detect these points we use the method of Linsen et al. [17]. All points of a local neighborhood of size n_b of a point $\mathbf{p}_i \in P$ are projected into the tangent plane and ordered according to angle. Whenever the angular distance between two consecutive points exceeds some maximum angle α_b, $\mathbf{p}_i$ is classified as a boundary point and its feature weight ω_i is set to maximum. The parameters n_b and α_b effectively control the size of the holes that are detected as boundaries.

Neighborhood Size. Increasing the size of the local neighborhood when computing the variation estimate eventually violates the prerequisite that all points of the neighborhood belong to the same connected region of the underlying surface. We can deal with this problem in two ways: Either we use neighborhood relations that are more sophisticated than Euclidean distance, e.g. a Riemannian graph or mesh connectivity, if available. Or we can try to estimate when the neighborhood becomes too large and stop the calculations of the variation measure. A simple heuristic that works well in practice is to look for jumps in σ_n, as they indicate strong deviations in the normal direction. Figure 8 shows an example for this method for a point

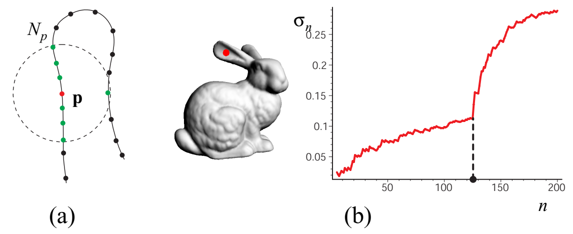

Figure 8: *(a) Illustration of an invalid neighborhood, (b) the critical neighborhood size for a point on the bunny's ear occurs at a jump in the variation-scale curve.*

on the bunny's ear. When increasing the size of the neighborhood, points from the opposite side of the ear will eventually be included in the set of k-nearest neighbors, as illustrated in Figure 8 (a). We can determine this critical neighborhood size by examining the variation-scale curve as shown in Figure 8 (b).

3. Feature Reconstruction

Feature reconstruction consists of three stages: First we select a set $Q \subset P$ of *feature nodes*, i.e. points that with high probability belong to a feature. Then we compute a minimum spanning tree (MST) for these feature nodes. After pruning short branches and closing cycles, each component of the resulting graph is modeled as a snake, which allows user-controlled smoothing of the feature lines.

3.1. Selecting feature nodes

In the classification stage of Section 2 we assigned weights ω_i to each sample point $\mathbf{p}_i \in P$ that measure the confidence that $\mathbf{p}_i$ belongs to a feature. To select the relevant feature nodes $Q \subset P$, we could discard all points whose weights fall below a certain threshold. This hard thresholding can cause undesirable artefacts, however, such as interrupted or dangling feature lines. As suggested in [2], hysteresis thresholding can alleviate these effects by using two thresholds $\omega_{\min} < \omega_{\max}$. Points with corresponding weights smaller than $\omega_{\min}$ are discarded, while points with $\omega_i > \omega_{\max}$ are included into the set of feature nodes Q. All points with $\omega_{\min} \le \omega_i \le \omega_{\max}$ will be used to bridge the gaps between feature nodes during the construction of the minimum spanning tree (see below).

3.2. Minimum spanning graph

To create a set of feature patterns we first compute the minimum spanning tree (MST) of the set of feature nodes Q. We start by ordering all feature nodes $\mathbf{q}_i \subset Q$ according to their weights. Then we choose the feature node $\mathbf{q}$ with biggest weight as the seed point for the construction of the MST. We define an edge for each of the k-nearest neighbors $\mathbf{q}_i$ of $\mathbf{q}$ and compute corresponding edge cost as

$$c(\mathbf{q}, \mathbf{q}_i) = \frac{1}{\omega_{\mathbf{q}}\omega_{\mathbf{q}_i}} + \gamma \cdot \|\mathbf{q} - \mathbf{q}_i\|, \qquad (8)$$

where γ is an additional parameter that allows us to balance feature weights against Euclidean distance. All these edges are put on a heap ordered by increasing cost values. Then we take the edge with the smallest cost and add it to the MST, if both edge nodes are not already part of the tree. For the new feature node we compute a new set of edges and corresponding cost values and also put these on the heap. We repeat this process until the heap is empty. Figure 9 (b) shows the MST of the dinosaur head generated with this algorithm.

Pruning and closing cycles. As can be seen in this example, the MST of all feature nodes contains many short branches, which are usually treated as artefacts that do not describe salient features. To eliminate these short branches from our set of feature patterns, we use a bottom up graph pruning method. Our algorithm starts by sorting all leaves of the MST according to their depth. By traversing the tree upward from the deepest node, we can determine the longest path, which defines a new root of the MST. Now we recursively compute all branches of the tree and assign to each branch an importance value that is given as the length of the branch multiplied by the product of all edge weights. Thus we retain short branches that contain feature nodes with high confidence values and only prune those branches that with low probability are part of a feature line.

The MST construction above does not support cycles in the feature lines. It is often desirable, however, to allow closed loops as these more naturally describe certain feature lines. Therefore we transform the MST into a graph by closing cycles that are longer than a user-specified threshold. To close a cycle, we use those edges whose feature nodes are already in the graph. From both these nodes we traverse the tree upward until the two paths cross at a common node. The sum of the two path lengths then equals the cycle length. Note that our method for pruning and closing cycles does not require an expensive breadth first search as was done, for example, in [5]. Figure 9 (c) shows the MST of Figure 9 (b) after pruning and closing cycles.

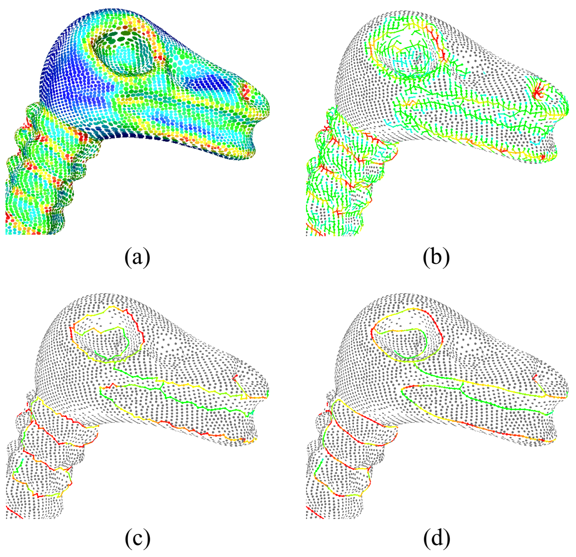

Figure 9: *Feature reconstruction on the dinosaur head: (a) feature weights, (b) minimum spanning tree of feature nodes, (c) MST after pruning and closing cycles, (d) smoothing with snakes.*

3.3. Active Contour Models

As can be seen in Figure 9 (c), the extracted feature lines connect samples of the original point cloud and are often jagged. This might be acceptable for partitioning algorithms, but for feature-based visualization methods (see Section 4) it leads to inferior rendering quality. We therefore need a mechanism for smoothing feature lines. Spline fitting has been used in previous approaches [5], but we found that it does not provide enough flexibility and control over the smoothness and accuracy of the extracted feature lines. In [11], Kass et al. introduced *snakes*, active contour models, for detecting features in images. A snake is an energy-minimizing spline that moves under internal and external forces. We use snakes to smooth the feature curves, while maintaining a close contact to the surface. The main benefit of snakes is their explicit control over the degree of smoothness that can be adapted to the specific application needs. Additionally, external constraints can easily be incorporated, for instance to enable user interaction for positioning feature lines.

Energy Minimization. We model each component of the MSG as a parametric curve $\mathbf{v}(s)$ that tries to minimize the energy functional

$$\mathbf{E} = \int \mathbf{E}_{int}(\mathbf{v}(s)) + \mathbf{E}_{ext}(\mathbf{v}(s))ds , \qquad (9)$$

The internal spline energy consists of first- and second-order terms, modeling the behavior of a membrane and a thin plate, respectively:

$$\mathbf{E}_{int}(\mathbf{v}(s)) = \alpha(s) \cdot |\mathbf{v}(s)'|^2 + \beta(s) \cdot |\mathbf{v}(s)''|^2 / 2 , \qquad (10)$$

where $\alpha(s)$ and $\beta(s)$ control the relative weight of the two terms. The external energy is related to the surface variation:

$$\mathbf{E}_{ext}(\mathbf{v}(s)) = 1 / \tilde{\sigma}(\mathbf{v}(s)) , \qquad (11)$$

where $\tilde{\sigma}(\mathbf{v}(s))$ is computed by interpolating the maximum variation of each point $\mathbf{p} \in P$ at $\mathbf{v}(s)$. Discretization of the functional $\mathbf{E}$ finally leads to a system of Euler equations that we solve using Euler integration (see [11] for details). Figure 9 (d) shows the smoothing effect on the dinosaur head.

4. Non-photorealistic Rendering

The feature lines extracted by our method can be used as input for a variety of processing algorithms, including mesh generation, anisotropic fairing and model segmentation. In this section we introduce a point rendering system for creating line drawings of point-sampled surfaces. Based on the surface splatting technique of Zwicker et. al. [25], our renderer takes as input a surfel model of the original surface and the output of our feature extraction method. Each feature line is converted into a set of feature surfels by sampling the model surface along the feature line. The surfels of the original model are only rendered into the z-buffer to resolve visibility and are assigned the background color. The final image is then only composed of the visible feature surfels. Note that no shading computations are applied, which significantly improves rendering performance.

To enhance the semantics of our renditions, we can utilize the additional information of the classification stage. We scale the splat radii of the feature surfels according to scale and adjust the intensity (e.g. grey level) according to the maximum surface variation. Thus features on coarser scales are rendered as thicker lines, while features on fine scales are rendered as thinner lines. Also prominent features are rendered at high intensities, while less significant features are rendered at low intensity. With these simple extensions, we achieve a very intuitive effect, similar to what an painter would do when drawing an image. To obtain more artistic looking renditions, we apply an additional screen space filter, as illustrated in Figures 10 to 14.

5. Results

We have implemented the feature extraction pipeline described in the previous sections and tested our method on a variety of point-sampled surfaces. Tables 1 and 2 summarize performance data for the different stages of the pipeline. For multi-scale classification we evaluate the surface variation estimate σ_n for each point $\mathbf{p} \in P$ for all neighborhood sizes n between 15 and 200 using the method described in the Appendix. We use kd-trees for computing the set of k-nearest neighbors, similar to [19]. Note that the interactive feedback loop for adjusting the various thresholding parameters does not include the multi-scale classification stage, which hence needs to be executed only once.

Model	multi-scale-classification	MST, pruning, closing cycles	snakes (100 Euler steps)
Igea	25.156	1.468	0.203
Cat	1.829	0.062	0.031
Gnome	6.703	0.203	0.047
Dinosaur	10.187	2.484	0.234
Dragon	64.422	8.469	1.125

Table 1: *Timing of the feature extraction pipeline in seconds on an Intel Pentium IV, 2.8 GHz.*

Model	#input points	#feature nodes	#snake points
Igea	134,345	40,509	7,327
Cat	10,000	3,593	798
Gnome	54,659	9,655	1,714
Dinosaur	56,194	45,879	6,395
Dragon	435,545	203,713	31,154

Table 2: *Complexity of the different stages.*

Figure 10 shows feature lines extracted on the Igea model. The middle image shows a rendition without the artistic screen space filter. The cat model of Figure 11 is more difficult, since it exhibits strong local imbalances in the sampling pattern. Still our method faithfully recovers the salient surface features. Figure 12 shows an example of a noisy laser range scan, which demonstrates that our multi-scale method is superior to single-scale classification. Also note that this is a difficult example for a (semi-) automatic feature extraction algorithm, because humans have a very clear and distinct perception of important feature lines in faces. The dragon and dinosaur models (Figures 13 and 14) show that our feature reconstruction method in connection with the point-based rendering method is very suitable for generating artistic line drawings of complex geometric surfaces.

6. Conclusions & Future Work

We have presented a complete semi-automatic feature extraction pipeline for point-sampled surfaces. Our main

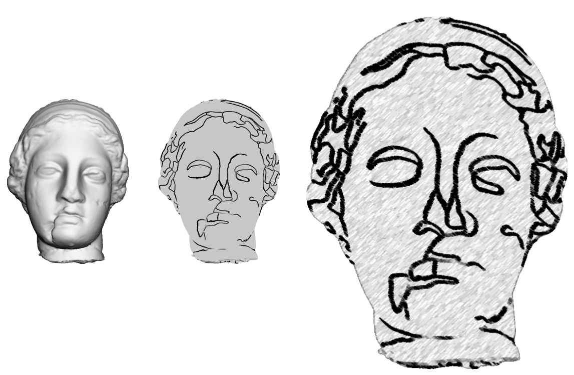

Figure 10: *Feature reconstruction on the igea model.*

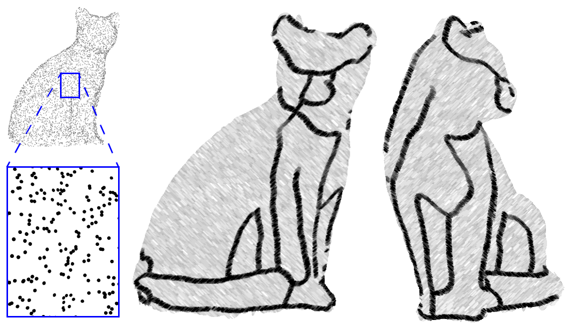

Figure 11: *The cat model is sampled very non-uniformly.*

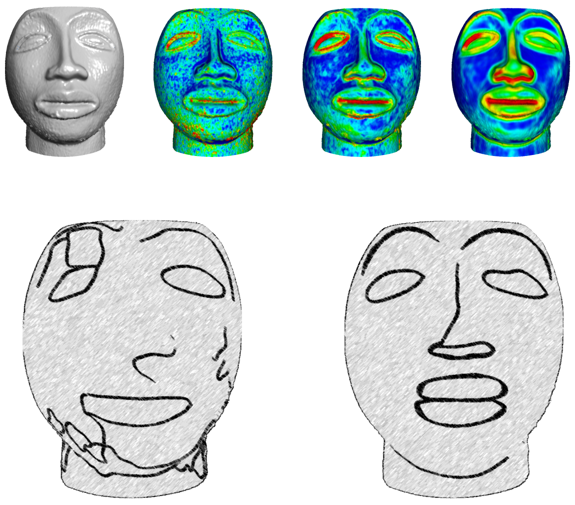

Figure 12: *Multi-scale feature extraction (bottom right) is superior to single scale extraction (bottom left) on a noisy range scan. The top row shows the original point cloud and variation estimates for different scales.*

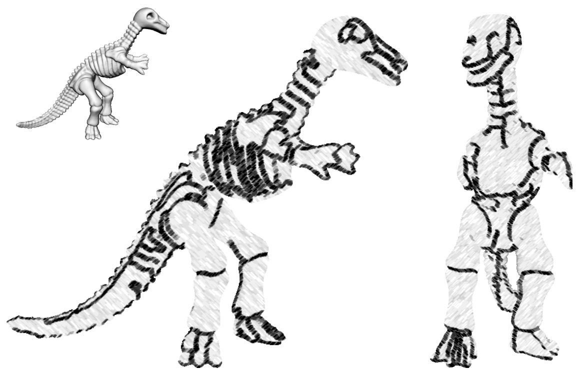

Figure 13: *The dinosaur model.*

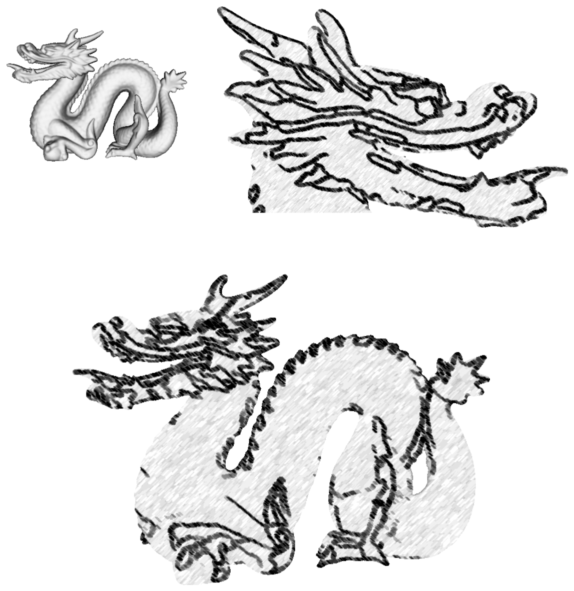

Figure 14: *The dragon model.*

contribution is a new classification framework that allows discrete surface analysis at multiple scales. Feature lines are extracted using a minimum spanning graph, which is modeled by a set of snakes for subsequent smoothing. We also present a new point-based non-photorealistic renderer that directly utilizes the additional information of the classification stage to enhance the semantics of the renditions.

Multi-scale feature analysis offers a number of advantages. First it makes the method more robust in the presence of noise. Second it allows coarse-scale features to be extracted even though the curvature might be low. Third it provides additional structural information, which can be used, for example, to adapt the width of feature lines according to the scale. We believe that our framework is general enough to be easily extended to incorporate more semantic information. It could, for instance, serve as a preprocess for model-based feature extraction that tries to match the extracted feature lines to a parameterized feature model. The surface variation estimate only considers the eigenvalues of the covariance matrix. More insights could be gained by also looking at the distribution of the eigenvectors over scale. In particular for early processing applications, such as surface reconstruction, this information could prove very useful.

Acknowledgements

This work is supported by the joint Berlin/Zurich graduate program Combinatorics, Geometry, and Computation, financed by ETH Zurich and the German Science Foundation (DFG).

References

1. Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., Silva, T. Point Set Surfaces, *IEEE Visualization 01,* 2001
2. Canny, J. A Computational Approach to Edge Detection. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol PAMI-8, No. 6, Nov. 1986
3. Desbrun, M., Meyer, M., Schröder, P., Barr, A. Implicit Fairing of Arbitrary Meshes using Diffusion and Curvature Flow. *SIGGRAPH 99*, 1999
4. Edelsbrunner, H., Letscher, D., Zamorodian, A. Topological Persistence and Simplificiation. *Discrete and Computational Geometry*, Springer, 2002
5. Gumhold, S., Wang, X., McLeod, R. Feature Extraction from Point Clouds. Proc. 10th Int. Meshing Roundtable, 2001.
6. Hall, P.M., Marshall, A.D., Martin, R.R. Incremental Eigenanalysis for Classification. Proc. of the British Machine Vision Conference 1998, Vol. 1., 1998
7. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W. Surface reconstruction from unorganized points. *SIGGRAPH 92*, 1992
8. Hubeli, A., Gross, M. Multiresolution Feature Extraction from Unstructured Meshes. *IEEE Visusualization 01*, 2001
9. Jolliffe, I. *Principle Component Analysis*. Springer-Verlag, 1986
10. Kalaiah, A., Varshney, A. Differential Point Rendering. *Rendering Techniques 01*, Springer Verlag, 2001.
11. Kass, M., Witkin, A., Terzopoulos, D. Snakes: Active Contour Models. *Int. Journal of Computer Vision*, 1988
12. Kobbelt, L. Discrete Fairing. *Proc. 7th IMA Conference on the Mathematics of Surface*s, pp. 101-131, 1997
13. Kobbelt, L. Botsch, M., Schwanecke, U., Seidel, H. Feature Sensitive Surface Extraction from Volume Data. *SIGGRAPH 01*, 2001
14. Lee, Y., Lee, S. Geometric Snakes for Triangle Meshes. *EUROGRAPHICS 02*, 2002
15. Levoy, M., Pulli, K., Curless, B., Rusinkiewicz, S., Koller, D., Pereira, L., Ginzton, M., Anderson, S., Davis, J., Ginsberg, J., Shade, J., Fulk, D. The Digital Michelangelo Project: 3D Scanning of Large Statues. *SIGGRAPH 00*, 2000
16. Lindeberg, T. Feature Detection with Automatic Scale Selection. *Int. Journal of Computer Vision*, vol. 30, no. 2, 1998
17. Linsen, L. Point Cloud Representation. *Technical Report*, Faculty of Computer Science, University of Karlsruhe, 2001
18. Pauly, M., Gross, M. Spectral Processing of Point-Sampled Geometry, *SIGGRAPH 01*, 2001
19. Pauly, M., Gross, M., Kobbelt, L. Efficient Simplification of Point-Sampled Geometry, *IEEE Visualization 02*, 2002
20. Pfister, H., Zwicker, M., van Baar, J., Gross, M. Surfels: Surface Elements as Rendering Primitives. *SIGGRAPH 00*, 2000
21. Rusinkiewicz, S., Levoy, M. QSplat: A Multiresolution Point Rendering System for Large Meshes. *SIGGRAPH 00*, 2000
22. Shaffer, E., Garland, M. Efficient Adaptive Simplification of Massive Meshes. *IEEE Visualization 01*, 2001
23. Taubin, G. A Signal Processing Approach to Fair Surface Design. *SIGGRAPH 95*, 1995
24. Witkin, A. Scale-Space Filtering. *Proc. 8th Int. Joint Conference on Artifical Intelligence,* 1983
25. Zwicker, M., Pfister, H., van Baar, J., Gross, M. Surface Splatting. *SIGGRAPH 01*, 2001
26. Zwicker, M., Pauly, M., Knoll, O., Gross, M. Pointshop 3D: An Interactive System for Point-based Surface Editing. *SIGGRAPH 02*, 2002

Appendix

We present an incremental method for computing the surface variation $\sigma_n(\mathbf{p})$ at a point $\mathbf{p}$ for increasing neighborhood size n (see Section 2.2). Assume that we have computed a neighborhood N_p with mean $\bar{\mathbf{p}}$, covariance matrix $\mathbf{C}$ and corresponding eigenvalues $\lambda_0 \le \lambda_1 \le \lambda_2$. Now we increase the neighborhood size by one, i.e. include the next closest point $\mathbf{q}$ to N_p. The new mean $\bar{\mathbf{p}}'$ can be obtained as $\bar{\mathbf{p}}' = (n\bar{\mathbf{p}} + \mathbf{q})/(n+1)$ and the new covari-

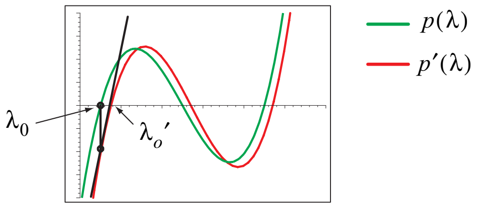

Figure 15: *Exploiting coherence when computing surface variation using Newton's method.*

ance matrix $\mathbf{C}'$ as

$$\mathbf{C}' = \frac{n}{n+1}\left(\mathbf{C} + \frac{\mathbf{q}'\mathbf{q}'^T}{n+1}\right), \qquad (12)$$

where $\mathbf{q}' = \mathbf{q} - \bar{\mathbf{p}}$ [6]. To compute the eigenvalues we find the roots of the characteristic polynomial, i.e. solve

$$p(\lambda) = |\mathbf{C} - \lambda\mathbf{I}| = \lambda^3 + p\lambda^2 + q\lambda + r = 0\,. \qquad (13)$$

The explicit formula for analytically computing the roots of a cubic polynomial uses trigonometric functions. We propose a more efficient method that exploits coherence of the local neighborhood. We are only interested in the smallest eigenvalue λ_0' of $\mathbf{C}'$ (note that $-p = \lambda_0 + \lambda_1 + \lambda_2$, cf. Equation 4) hence we use Newton iteration to find λ_0'. Since the characteristic polynomial $p(\lambda)$ changes only slightly when adding another sample point to the neighborhood of $\mathbf{p}$, λ_0 provides a very good initial guess for λ_0' (see Figure 15). Due to the quadratic convergence of the Newton scheme, we then typically require less than 3 iterations.