Vision, Modeling, and Visualization (2010)

Feature Preserving Sketching of Volume Data

J. Kerber[1,2] M. Bokeloh[1] M.Wand[1,2] J. Krüger[3,4] H.-P. Seidel[1,2]

[1] MPI Informatik Saarbrücken, [2] Saarland University,
[3] IVDA Group & IVCI, Germany, [4] SCI Utah, USA

Abstract
In this paper, we present a novel method for extracting feature lines from volume data sets. This leads to a reduction of visual complexity and provides an abstraction of the original data to important structural features. We employ a new iteratively reweighted least-squares approach that allows us to detect sharp creases and to preserve important features such as corners or intersection of feature lines accurately. Traditional least-squares methods This is important for both visual quality as well as reliable further processing in feature detection algorithms. Our algorithm is efficient and easy to implement, and nevertheless effective and robust to noise. We show results for a number of different data sets.

Keywords: line feature extraction, least square approximation, sketching, volume data

1. Introduction

Volume datasets provide a huge amount of information that is not easily graspable due to occlusion and visual clutter. This is a challenge for both a human observer as well as for automated data analysis techniques, and this motivates *feature extraction* techniques: The goal is to reduce the data to essential features that summarize important structural properties of the data set in a more compact form. This allows a human observer to focus on certain structural aspects (as extracted by the feature detector). For a computational data analysis, it transforms the problem from the difficult problem of "understanding" a continuous data set into a simpler problem of analyzing a collection of discrete feature elements. Obviously, it is of particular importance to preserve geometrical and topological key properties that provide insight into the data characteristics.

There is a long history of feature detection techniques that extract interest points [Low03], lines [MI97, OBS04, BBW*09], or entire regions [HFG*06]. In many applications, graphs of line features have been particularly effective as a coarse sketch of the geometry of a data set: For images and 3D surfaces, it is even often possible to reconstruct most of the salient geometry from just a few high curvature edges [OBS04,MW09], which makes them a good candidate for automated shape analysis techniques [BBW*09]. From the point of view of volume visualization, finding graphs of feature lines is also promising: On the one hand, line graphs provide connectivity so that the topological structure is easier to grasp than in clouds of feature points. On the other hand, occlusion problems of extended isosurfaces are avoided.

This paper describes a new scheme for summarizing volume data by a simple line sketch. Our approach is motivated by recent work that uses robust statistical estimation techniques for reconstructing surfaces while preserving sharp features [FCOS05, DTB06, OGG09]. Similarly, we employ robust statical fitting techniques to extract line skeletons that preserve features such as corners and intersections. Our algorithm consists of two stages: We first extract candidate regions by analyzing gradient magnitude and curvature [HKG01, HG02]. Simple thresholding of such interest regions typically leads to extended and smeared-out results. Therefore, in second step, we apply a mean-shift-like projection algorithm to contract the line candidate points to sharply located curves [CM02, WL08, HR08, BBW*09, CTO*10]. In both steps of our algorithm, we employ iteratively reweighted least-squares computations with robust bilateral weights, which significantly improves the results at sharp corners and in regions where several feature lines branch or intersect. This is the main contribution of this work. We argue that preserving such topological features is particularly important for human shape understanding as well as machine analysis because the branching structure and the sharp corners of the line sketch give important cues

© The Eurographics Association 2010.

DOI: 10.2312/PE/VMV/VMV10/195-202

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

about the data structure. Traditional least squares techniques usually fail in such regions as they violate the underlying assumption of fitting a single, unimodal model to the local data.

Our new technique is conceptually simple and easy to implement but nevertheless yields a substantial improvement in quality of the extracted feature lines over traditional least-squares techniques, which we demonstrate for a number of example data sets. The performance penalty is moderate, requiring only a small number of refitting operations to gain the additional robustness. Besides the purpose of visualization, we belief that our technique is also useful in the context of processing and analyzing volume data (such as measuring, registration, matching, or abnormality detection), but this paper focuses on the extraction algorithm and its use in visualization.

2. Related Work

The detection and extraction of contour lines and skeletons has been in the focus of research since the early days of computer graphics, vision and visualization. In his pioneering work [Can86] Canny extracts edges from 2D images by detecting lines along local gradient extrema. Elder and Zucker [EZ98] exploit the information covered in edges to achieve storage compression for images. Their characterization of an edge, as zero crossings of the image Laplacian is more general, and their method is capable of detecting blurred edges over multiple scales. Zhan et al. [MM94] describe an extension for 3D edge detection and a generalization of the 2D sobel operator to the volumetric case as presented by Prabir et al. [BW96].

For point clouds, Gumhold et al. [GWM01] present a technique to detect crease lines and border parts. They describe points and the constitution of their local neighborhoods by a correlation ellipsoid, among others, their results can be used to mark areas of interest in a non photo realistic manner. Pauly et al. [PKG03] extend the idea to multiple scales, which makes their method more robust to noise. They demonstrate how the acquired contours can be used to stylize the underlying geometry. Hildebrandt et al. [HPW05] propose a technique to extract feature lines from surface meshes to emphasize its visually most prominent characteristics. De Carlo et al. [DR07] present a method to detect and highlight principle as well as suggestive contours on a mesh model to increase their relative importance for a viewer.

Wang et al. [WL08] show how to extract a skeleton from a volume data set. They first shrink the entire model in a least square sense, followed by a thinning step. Hesselink et al. [HR08] introduce a slightly modified definition of a skeleton and demonstrate how to compute it efficiently. Recently, Cao et al. [CTO*10] demonstrate how to extract a point based skeleton using a Laplacian based contraction scheme.

Our approach is based on two strains of ideas: The first ingredient is classical detection algorithms for feature lines: We employ a differential surface analysis to detect salient lines. The formulation is similar to Hladuvka et al. [HKG01, HG02]. Our shrinking strategy is similar to Bokeloh et al. [BBW*09] and related to previous skeletonization techniques, as listed in the previous paragraph. The second ingredient is robust statistical fitting that assumes non-Gaussian distributions of differential properties [FCOS05, DTB06]. Our actual implementation of robust moving least squares fitting is most closely related to the work by Oztireli et al. [OGG09], who propose a similar technique for surface reconstruction with sharp creases.

3. Iteratively Reweighted Least-Squares

Before we go further into the details of the algorithm, we would like to briefly introduce the main computational tool our framework is based on, iteratively-reweighted least-squares fitting (see for example the work by Oztireli et al. [OGG09] for more details).

Assume that we would like to represent a function $f : \mathbb{R}^d \supseteq \Omega \to \mathbb{R}$ by a linear combination of basis functions $B = b_1, \ldots, b_k, b_i : \mathbb{R}^n \to \mathbb{R}$ using the (unknown) coefficients $\lambda_1, .., \lambda_k$. This problem can be solved by minimizing the quadratic error function

$$E(f) = \int_\Omega \omega(\mathbf{x}) \left(f - \sum_{i=1}^{k} \lambda_i \mathbf{b}_i(\mathbf{x}) \right)^2 d\mathbf{x},$$

which yields a simple linear system of equations. The weight function ω can be used to control the influence of data points. Frequently, ω is set to a smooth windowing function, such as a Gaussian, and its center is moved through the data. This transforms the original function f to a filtered function $\tilde{f}$ which is the *moving least squares* (MLS) reconstruction of f. We employ this strategy with a quadratic monomial basis to estimate differential properties of the volume. However, at (higher order) discontinuities, such as creases or corners, the least squares approach yields blurred and smeared out results, which is due to the quadratic penalty that disproportionally discourages larger fitting errors. This problem can be avoided by using a different error metric than the L_2-norm that implies a Gaussian error distribution (a Gaussian distribution is a distribution of maximum entropy at fixed variance, implying an unstructured error distribution). For most data sources, this is not realistic: Typically, we have much smaller errors but a few far out outliers due to sharp creases. In order to account for this, a heavier-tail distribution such as an L_p norm with $0 < p \leq 1$ should be used [DTB06]. This can be implemented easily using reweighting, i.e., by choosing a weight function ω that is reciprocal to a power of the residual of the fit: We start with an L_2 fit and reweight iteratively until convergence. This strategy is globally optimal for $p \geq 1$ due to the convexity of the functional. In our case,

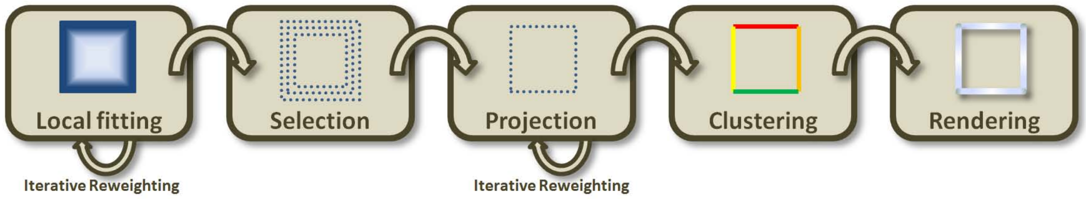


Figure 1: *Diagram describing the interplay of the different algorithm stages.*

we use a Gaussian kernel for reweighting that leads to a non-convex optimization problem but tends to preserve sharp features better (for a constant monomial basis, this reduces to the bilateral filter). In our application of fitting small local patches, non-convexity is not an issue as the L_2 initialization gives a good initial estimate.

A further extension is a more general, attribute-driven bilateral reweighting [FCOS05]: If we have multiple properties at each point in space such as normals or curvatures (not just the function value), we can also perform a Gaussian reweighting based on the difference of these properties. This is useful for preserving discontinuities in these properties as the reweighting implicitly partitions the problem into points with similar properties. We use this idea in our projection algorithm to robustly handle corners, junctions, and intersections of line segments.

4. Algorithm Description

The sequence of the different intermediate steps of our approach is depicted in Figure 1. Our algorithm first analyzes the local neighborhood of each voxel by fitting a quadratic function to it which is repeatedly refined. The fitted function allows for describing the geometric properties in the voxels neighborhood. From this, we select candidate regions containing topological features. In general, these regions exhibit a width of several voxels. In the projection step, we repeatedly shrink this extent which results in a dense skeleton-like representation. According to their context, we combine the yet independent feature points to crease lines. The final step consists of the appropriate visualization. In the remainder of this section, the above mentioned stages are explained in detail.

4.1. Local fitting

The first step of our algorithm is to estimate the second order differential properties of our input volume at each voxel. This is done using a MLS scheme with iterative reweighting. We employ a second order monomial basis **b** and estimate a coefficient vector **c** such that the resutling function f locally approximates the volume V well in the vicinity of a point $\mathbf{x}_0$.

$$\begin{aligned} \mathbf{b} &= (1,x,y,z,xy,xz,yz,z^2,x^2,y^2) \\ \mathbf{c} &= (c_1,c_x,c_y,c_z,c_{xy},c_{xz},c_{yz},c_{xx},c_{yy},c_{zz}) \\ f_{\mathbf{x}_0}(x,y,z) &= \mathbf{b}\cdot\mathbf{c}^T \end{aligned}$$

In order to estimate f, we will iteratively solve for multiple such solutions, indexed by a time parameter $t = 0,1,....$ In each iteration, we compute the $\mathbf{c}^{(t)}$ and thereby an $f^{(t)}$ that minimize

$$\sum_{\mathbf{x}\in N(\mathbf{x}_0)} (f^{(t)}(\mathbf{x}) - V(\mathbf{x}))^2 \cdot \omega^{(t)}_{\mathbf{x}_0}(\mathbf{x}),$$

where N denotes voxels of the volume that lie within the support of the weighting function $\omega_{\mathbf{x}_0}$. We initially use a purely spatial Gaussian window

$$\omega^{(0)}_{\mathbf{x}_0}(x) = \exp\left(-\frac{(\mathbf{x}-\mathbf{x}_0)^2}{2\sigma^2_{spat}}\right)$$

with a standard deviation σ_{spat} corresponding to a few voxels (we truncate the Gaussian when it becomes numerically close to zero; see Section 4.6 for a discussion of concrete parameter choices). Given the initial estimate, we perform a reweighting where we take ω to be the product of the spatial kernel used so far and a kernel function in the intensity domain:

$$\omega^{(t)}_{\mathbf{x}_0}(x) = \exp\left(-\frac{(\mathbf{x}-\mathbf{x}_0)^2}{2\sigma^2_{spat}} - \frac{(f^{(t-1)}(\mathbf{x}) - V(\mathbf{x}))^2}{2\sigma^2_{intens}}\right)$$

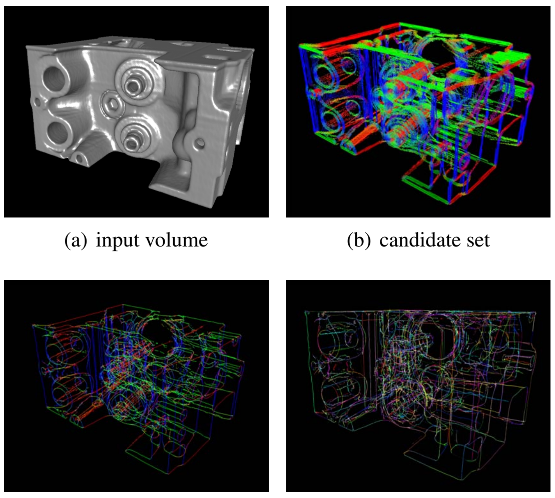
We iterate this reweighting until convergence. The coefficients obtained from this estimate yield a robust and denoised estimation of a second order Taylor series for the data in V at point $\mathbf{x}_0$: The linear coefficients represent the gradient and thus the normal direction of a local isosurface. We denote this by:

$$\mathbf{n} := (c_x,c_y,c_z),\ G := ||\mathbf{n}||,\ \mathbf{n_1} := \frac{\mathbf{n}}{G}$$

The quadratic coefficients yield estimates for the entries of the Hessian matrix:

$$\mathbf{H} = \begin{pmatrix} c_{xx} & c_{xy} & c_{xz} \\ c_{xy} & c_{yy} & c_{yz} \\ c_{xz} & c_{yz} & c_{zz} \end{pmatrix}$$

We multiply from the left and from the right with a projection matrix that cancels out contributions in normal direction



(a) input volume (b) candidate set

(c) projected candidate set (d) orientation clusters

Figure 2: *An illustration of the effects of the intermediate steps using the example of the engine block data set.*

and thus obtain the curvature tensor of a local isosurface:

$$\begin{aligned} P &= \mathbf{I} - \mathbf{n_1} \cdot \mathbf{n_1}^T \\ \overline{\mathbf{H}} &= P \cdot H \cdot P^T \end{aligned}$$

Obviously, $\overline{\mathbf{H}}$ has at most rank 2. Let therefore x_1,x_2 be the normalized eigenvectors and λ_1, λ_2 the corresponding eigenvalues (largest absolute value first) of $\overline{\mathbf{H}}$. After dividing by the gradient magnitude, the eigenvalues yield the principal curvatures $\kappa_{max} = \lambda_1/G, \kappa_{min} = \lambda_2/G$ of a (hypothetical) isosurface crossing $\mathbf{x}_0$. Correspondingly the eigenvectors yield the directions of principal curvature $\mathbf{k}_{max}, \mathbf{k}_{min}$.

4.2. Candidate selection

After computing differential quantities, we will extract regions that correspond to sharp feature lines in the volume. These regions are characterized by both a strong gradient magnitude (indicating a local surface) as well as a large maximum principal curvature. In this first step, we use a simple thresholding method to identify a candidate area of interest:

$$C = \{\mathbf{c}_i \in V | G(\mathbf{c}_i) > \tau_G, |K_{max}(\mathbf{c}_i)| > \tau_K\}$$

With τ_G and τ_K as thresholds for the gradient and the absolute value of K_{max}. By these parameters, the user can control how pronounced a line feature needs to be to be included in the candidate set C (see Section 4.6). As an example, Figure 2a) and b) shows input and the result of this stage for the well known engine block model. Please note that the color encodes the orientations of k_{min}, which is the tangential direction of the extracted lines

4.3. Projection

So far, we have a candidate set of voxels that are located near actual sharp features. However, their location is still smeared out. Our goal is now to shrink these smeared out feature regions to thin and (piecewise) smooth feature lines, a step that we call *projection* onto feature curves. Formally, for every voxel-quantized point $\mathbf{c}_i \in C$ from the candidate set, we compute a projected point $\mathbf{p}_i^{(}t)$ at a continuous coordinate. Our procedure is very similar to mean shift clustering [CM02]: Starting at the candidate position $\mathbf{c}_i$, We iteratively move the points $\mathbf{p}_i$ the the centroid of its local neighbors in the original candidate set C, weighted by a spatial Gaussian window. In the following, we will use $\mathbf{p}_i^t$ to denote the position of the moving point at step t of this iteration. Please note that we always compute weighted averages with respect to the original candidate set, not the moved points. Unalteredly, the described procedure would move the points to the centroids of regions where the highest density of candidate points is located (this is mean shift clustering). However, we want to extract lines, not centroid points. Therefore, we restrict the movement of the $\mathbf{p}_i$: For each point, the minimum principal curvature of the isosurface $\mathbf{k}_{min}$ is the tangential direction of line we are extracting. Correspondingly, we restrict the candidate points to not move into the direction of $\mathbf{k}_{min}$ (again, the constraint direction is fixed throughout the iteration). This procedure yields thin feature lines, but does not yet lead to satisfactory results in regions where lines of different direction are close to each other. Therefore, we again employ bilateral reweighting to separate the influence of lines of different tangential direction onto each other. A new position is now computed by the bilaterally weighted average of neighboring positions: in addition to spatial distance, we also take the deviation in tangential direction into account; points with very dissimilar tangents have only minimal influence on the projection. We summarize the procedure in the following algorithm:

$$\begin{aligned} \mathbf{p}_i^{(0)} &= \mathbf{c}_i \\ \mathbf{p}_i^{(t+1)} &= (\mathbf{I} - \mathbf{k}_{min}(\mathbf{c}_i) \cdot \mathbf{k}_{min}(\mathbf{c}_i)^T) \cdot \left(\frac{\sum\limits_{j \neq i} \omega_i^{(t)}(\mathbf{c}_j) \cdot \mathbf{c}_j}{\sum\limits_{j \neq i} \omega_i^{(t)}(\mathbf{c}_j)} \right) \\ \omega_i^{(t)}(\mathbf{x}) &= \exp\left(-\frac{(\mathbf{p}_i^{(t)} - \mathbf{x})^2}{\sigma_{spat}^2} - \frac{\angle(\mathbf{k}_{min}(\mathbf{c}_i), \mathbf{k}_{min}(\mathbf{x}))}{\sigma_{angle}^2} \right) \end{aligned}$$

In the case that the sum of weights of a point is very small (leading to a near-zero denominator in the weighting), we discard the point altogether as an outlier. As a result of this stage, we are given a narrow skeleton which consists of yet unconnected representatives. This is illustrated in Figure 2 c). Note the difference in thickness by preserving the orientation at the same time.

4.4. Clustering

Since many positions coincide after the projection step we will decimate the feature points first. Although we want to reduce their number for further processing, we need to keep those which belong to different lines especially at junction points. Therefore we start a context aware resampling at an arbitrary point by erasing all points closer than half a voxel diameter, whose orientation of $\mathbf{k}_{min}$ differs less than σ_{angle}. We repeat this for all remaining points.

Now that the feature points are pruned, we cluster them to crease lines by a region growing approach. We again start at an arbitrary point and project a line segment through this point, whose orientation is given by the minimal principal curvature direction. Then we collect all points in a local neighborhood with a distance less than σ_{spat} to the projected line segment and which do not differ more than σ_{angle} in their own orientation. This is repeated until no valid neighbors can be added. Then a new cluster starts at the next point which has not been assigned before, until all points have been considered.

The representatives gathered in these clusters form the *control points* of the crease lines. This clustering step comes along with an abstraction of structural information from points to curves, and so leads to a reduction in the number of scene elements. Circles and circular arcs are approximated by several clusters. Figure 2 d) shows a zoom on the engine block model. Different colors indicate different clusters.

In order to overcome problems with outliers we remove clusters with less than 4 representatives.

As a corner or junction, we define those points where two clusters intersect or approach closer than σ_{spat} and whose orientation differs more than $2 \cdot \sigma_{angle}$. These regions are specially marked for later highlighting.

4.5. Rendering

We have implemented two different options for rendering the result. The first is a straightforward point-based rendering, where every extracted point on a line is just displayed as a dot, or alternatively, as a small line segment in line direction (k_{min}) with length proportional to the sample spacing. This corresponds to the native output format of our feature extraction pipeline but it is obviously not a good representation for rendering with complex shading. In particular, it is hard to perceive occlusion and depth order for a simple, uniformly colored line drawing. Therefore, we convert the point sampled lines into cylindrical surfaces of fixed radius R in an (optional) additional step. This yields a visualization that looks like a network of pipes. In order to compute a valid manifold triangle mesh for line networks of general topology, we employ a moving least squares (MLS) based implicit surface fitting technique inspired by the work of Shen et al. [SOS04] (who consider the case of surfaces rather than lines): For each sample point, indexed by $i = 1..n$, we setup an infinite cylindrical distance function f_i that encodes the distance to the straight line of infinite length that passes through the point sample in k_{min}-direction. In order to evaluate the implicit function f at an arbitrary position in space, we use a partition of unity weighted combination of the implicit functions f_i:

$$f(\mathbf{x}) = \frac{\sum_{i=1}^{n} \omega(\mathbf{x})(f_i(\mathbf{x}) - R)}{\sum_{i=1}^{n} \omega(\mathbf{x})}$$

This gives us an implicit moving least squares (IMLS) scheme [OGG09] for which its zero level set encodes the surface to be extracted. We use a simple Gaussian kernel as weighting function ω with a standard deviation above sample spacing to create smooth results. Because this function has (at least numerically) compact support, we do not evaluate the implicit function for points in space that are farer away from than the support of the weighting function. This is implemented using a standard octree decomposition of space that only subdivides space in proximity to sample points. The maximum number of subdivisions is a user parameter that determines the resolution of the resulting mesh. For all octree leave nodes that are within the support of the Gaussian kernel centered around a line sample point, we run a standard marching cubes algorithm to create a triangle mesh. In order to avoid displaying small outlier segments, we delete connected components with a small number of triangles.

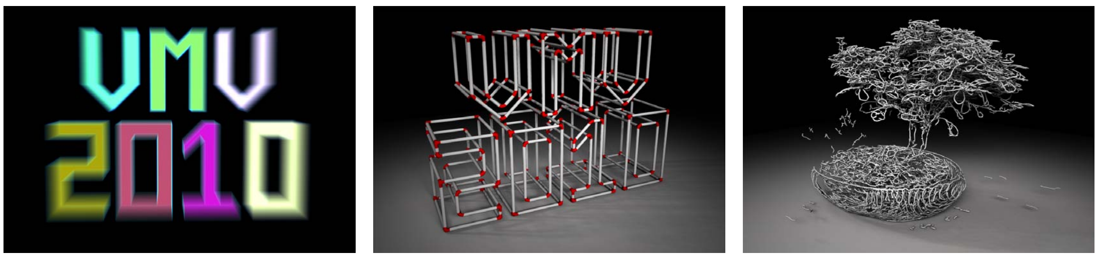
4.6. Parameters

In practice we use spatial Gaussian windows with a standard deviation σ_{spat} of 2-3 voxels and a truncation outside a support volume of 5^3 voxels. The standard deviation σ_{intens} is set to a fixed fraction of the observed intensity range, for our examples we use $\sigma_{intens} = 5\% \cdot (I_{max} - I_{min})$. The thresholds for the candidate selection τ_G and τ_K are also a percentage of the highest gradient and the highest absolute value of the maximal principal curvature, found in the entire volume. Their tuning is crucial and is discussed in the next section. σ_{angle} which affects the bilateral weight in the projection and is later used in the clustering, ranges between 20° and 30°.

5. Results

Figure 3 (a) shows a synthetic volume. The letters and numbers exhibit very sharp edges and corners. Each symbol is given a different intensity value, indicated by different colors in the original volume. The red coloring in the results marks the positions of corners and junctions (b). Note the sharpness and connectivity as well as the faithful depth order in the rendering. In part (c) we demonstrate the result for the bonsai tree model. It preserves the sharp outlines of the leaves and allows to trace branches. We omitted to colorize junctions here because they are too frequent, especially at the roots in the plant pot. In Figure 4 (a) we show the final rendering for

© The Eurographics Association 2010.



(a) (b) (c)

Figure 3: *An artificially generated volume (a) and its extracted line sketch (b). Our result for the bonsai tree model (c).*

the engine block data set. Part (b) shows a zoom on a corner of our result where 3 edges meet. In (c) we demonstrate what happens if the projection to the plane orthogonal to the line direction is not applied. In this case it behaves like regular mean-shift filtering, thus the edges move away form their junction point and tear apart the geometry.

The advantages of our method over a traditional least-squares baseline approaches are examined in figure 5, using the engine block example again. The upper left image shows the candidate set computed by the method described in this paper. The lower left image shows the corresponding image with reweighting omitted (setting σ_{intens} to infinity), but using the same parameters otherwise. The resulting candidate set is larger (227k to 122k) and does not as clearly separate feature lines in the volume. Moreover, many more planar areas are recognized as features. This shows the shortcomings of a standard MLS for detecting sharp edges and corners. The middle column shows the results of the projection step for the both images on the left, using the same parameter settings. One can easily see that the planar regions remain as dense artifacts on the left and in the background. In the right column we see what happens if the bilateral weighting of the orientation is omitted. Edges lose their sharpness, and mutual influence of adjacent lines of different direction starts to show up: Note in particular the circles on the left side of the engine block. Without reweighting, they show a distorted shape because they are attracted by the points which belong to an intersecting line. In order to evaluate the improvement of the techniques described in this paper, we have to compare the lower right image (no reweighting at all) to the middle image in the upper row. Obviously, the feature lines are extracted less accurately and in several places, even the topology of the line graph is wrong without reweighting (see the inner left bottom corner, for example).

5.1. Open problems

Since spheres, cylinder and slightly curved planes exhibit a high gradient as well as a high value of the maximal principal curvature, it is hard to distinguish those undesired candidate regions from linear structures. Introducing another threshold which takes into account K_{min} in order to not detect spherical parts as candidates does not work because corner regions have high values for K_{min} as well. We therefore restrict ourselves to thresholding K_{max} only. That is why the thresholding is so crucial for the quality of the results. Either the thresholds are low enough to allow all small details at the cost of introducing undesired regions or they are too high and important structures are lost but the candidate set is free of false positives. Our examples use rather low thresholds. Spheres, cylinders and planes are characterized by many features side by side which point into the same direction. Our projection algorithm terminates very early because the attraction of the points in the neighborhood is equally high. This leads to numerous relatively small clusters. This fact allows us to use the post-processing step described in Section 4.5 which removes small unconnected components from the final output. This removes most artifact in problematic areas but potentially sacrifices small linear features as well.

Figure 6 show our result for a micro-ct scan of a mechanical part. One can see how problematic regions look before and after the post-processing. A more general solution would probably be to integrate path along the surface in $\mathbf{k}_{max}$ direction and check whether the angle spanned by similar curvature spans more than $180°$; we leave the examination of such filtering techniques for future work.

5.2. Performance

The most time consuming part of our algorithm is the local fitting step. It is linear in the number of voxels, the kernel size and the number of reweighting steps. The duration of the projection stage depends on the size of the candidate set and the neighborhood radius. The time for convergence varies from model to model.

Our prototype implementation is written in C++ using OpenMP™ to allow parallel computation. The performance was measured on an Dual Socket Intel® I7™ system running at 2.66 GHz with 24 GB of main memory. Table 1 includes timings for the intermediate steps of several models, with parameter settings as they are explained in section 4.6.

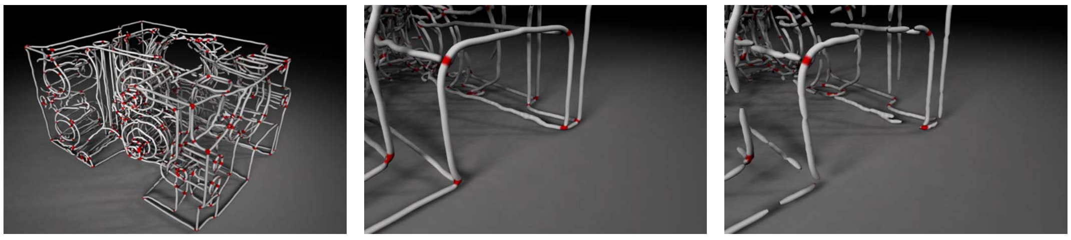


Figure 4: *The rendered outcome for the engine data set and a zoom on the lower part of the left side (a) and (b). The same view on a result achieved without taking care of the orientation in the projection step (c).*

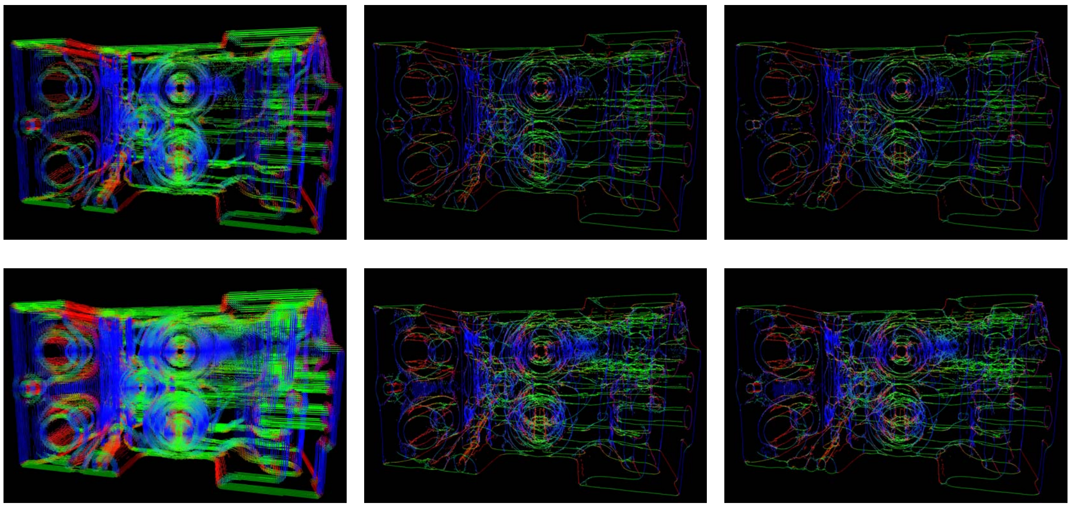


Figure 5: *A comparison of intermediate results achieved which illustrate the influence of correct reweighting. Upper row: candidate set computed with reweighting, lower row: without. Left column: initial candidate set. Middle column: After projection, with reweighting, Right column: after projection without reweighting. Our result is the upper/middle image, the base-line result is the lower right one.*

Other operations like thresholding, resampling, crease line clustering and corner detection only consume fragments of seconds in practice, thus they are not listed there. One can see that the bonsai tree has a very high number of feature points compared to the engine block model. Although the size of its candidate set is almost similar to the one of the mechanical part, the neighborhoods are much more complex and it takes longer until the projection reaches a steady state.

Model	Resolution	Fitting	$\|C\|$	Projection
Logo	[200,150,60]	11 sec	42k	1 sec
Engine	[256,256,256]	100 sec	123k	3 sec
Bonsai	[256,256,256]	100 sec	555k	25 sec
Part	[504,504,225]	360 sec	545k	13 sec

Table 1: *Listed timings for different models.*

6. Conclusion

We have presented a novel feature line extraction algorithm for volume data sets. In contrast to traditional methods, our algorithm employs robust statistical fitting techniques at several steps of the algorithm, which is motivated by recent work in feature preserving surface fitting [DTB06, FCOS05, OGG09]. As a result, we are able to reproduce sharp corners, junctions, and intersections of feature lines significantly more accurately than standard least-squares techniques, which frequently introduce geometric errors in these regions that are large enough to even alter the topology of the resulting feature graph. We believe that a robust extraction of line graphs of general topology is important in both visualization of data sets as well as automated data analysis. The shortcoming of traditional techniques, which yield artifacts at topologically sensitive junction areas, is particularly bad for both an effective visual summary

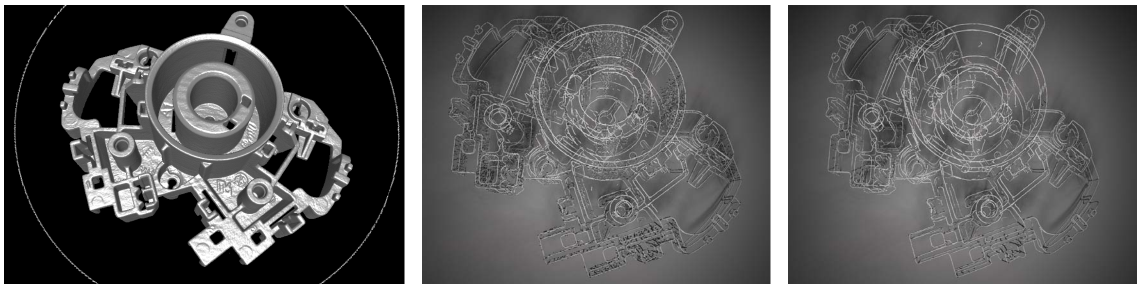


Figure 6: *Result renderings of a mechanical part (left). Before (middle) and after post-processing (right). Note that the disturbing artifacts around the middle cylinder and the surfaces at the outer boundary are removed.*

and even more for automated data analysis. Our new approach reduces these problems significantly. As a limitation and avenue for future work, our algorithm still has a few parameters for the user to be set. The automation of these choices could be improved. We also sometimes obtain artifacts from moderately curved cylindrical and spherical surfaces, which could be filtered out more effectively, as discussed above. In addition to this, we would also like to examine data processing algorithms that operate on line graphs such as feature-based symmetry detection [BBW*09].

References

[BBW*09] Bokeloh M., Berner A., Wand M., Seidel H.-P., Schilling A.: Symmetry detection using line features. *Computer Graphics Forum 28*, 2 (2009).

[BW96] Bhattacharya P., Wild D.: A new edge detector for gray volumetric data. *Computers in Biology and Medicine 26*, 4 (1996), 315–328.

[Can86] Canny F. J.: A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence 8*, 6 (1986), 679–698.

[CM02] Comaniciu D., Meer P.: Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24* (2002), 603–619.

[CTO*10] Cao J., Tagliasacchi A., Olson M., Zhang H., Su Z.: Point cloud skeletons via laplacian-based contraction. In *Proc. of IEEE Conf. on Shape Modeling and Applications* (2010), pp. 187–197.

[DR07] DeCarlo D., Rusinkiewicz S.: Highlight lines for conveying shape. In *International Symposium on Non-Photorealistic Animation and Rendering (NPAR)* (Aug. 2007).

[DTB06] Diebel J. R., Thrun S., Brünig M.: A bayesian method for probable surface reconstruction and decimation. *ACM TOG 25*, 1 (2006), 39–59.

[EZ98] Elder J. H., Zucker S. W.: Local scale control for edge detection and blur estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20* (1998), 699–716.

[FCOS05] Fleishman S., Cohen-Or D., Silva C. T.: Robust moving least-squares fitting with sharp features. *ACM TOG 24*, 3 (2005), 544–552.

[GWM01] Gumhold S., Wang X., Macleod R.: Feature extraction from point clouds. In *In Proceedings of the 10 th International Meshing Roundtable* (2001), pp. 293–305.

[HFG*06] Huang Q.-X., Flöry S., Gelfand N., Hofer M., Pottmann H.: Reassembling fractured objects by geometric matching. *ACM Trans. Graphics 25*, 3 (2006), 569–578.

[HG02] Hladuvka J., Gröller E.: Exploiting the Hessian matrix for content-based retrieval of volume-data features. *The Visual Computer 18*, 4 (2002), 207–217.

[HKG01] Hladuvka J., König A., Gröller E.: Salient representation of volume data. In *Data Visualization 2001, Proceedings of the Joint Eurographics – IEEE TVCG Symposium on Visualization* (2001), pp. 203–211,351.

[HPW05] Hildebrandt K., Polthier K., Wardetzky M.: Smooth feature lines on surface meshes. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing* (2005), pp. 85–90.

[HR08] Hesselink W. H., Roerdink J. B.: Euclidean skeletons of digital image and volume data in linear time by the integer medial axis transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30* (2008), 2204–2217.

[Low03] Lowe D.: Distinctive image features from scale-invariant keypoints. In *Int. J. Computer Vision* (2003), vol. 20, pp. 91–110.

[MI97] Ma K.-L., Interrante V.: Extracting feature lines from 3d unstructured grids. In *Proc. IEEE Visualization '97* (1997), IEEE Computer Society Press.

[MM94] MZhan S., Mehrotra R.: A zero-crossing-based optimal three-dimensional edge detector. *CVGIP: Image Understanding 59*, 2 (1994), 242–253.

[MW09] Mainberger M., Weickert J.: Edge-based image compression with homogeneous diffusion. *Computer Analysis of Images and Patterns (LNCS) 5702* (2009), 476–483.

[OBS04] Ohtake Y., Belyaev A., Seidel H.-P.: Ridge-valley lines on meshes via implicit surface fitting. In *Proc. Siggraph* (2004), pp. 609–612.

[OGG09] Oztireli C., Guennebaud G., Gross M.: Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum 28*, 2 (2009).

[PKG03] Pauly M., Keiser R., Gross M.: Multi-scale feature extraction on point-sampled models. *In Proc. of EUROGRAPHICS 2003 22*, 3 (2003).

[SOS04] Shen C., O'Brien J. F., Shewchuk J. R.: Interpolating and approximating implicit surfaces from polygon soup. *ACM Transactions on Graphics 23*, 3 (2004), 896–904.

[WL08] Wang Y.-S., Lee T.-Y.: Curve-skeleton extraction using iterative least squares optimization. *IEEE Transactions on Visualization and Computer Graphics 14*, 4 (2008), 926–936.