Volume xx (200y), Number z, pp. 1–15

Surface Reconstruction from Unorganized Point Data with Quadrics

Marek Vančo[1], Bernd Hamann[1,2] and Guido Brunnett[3]

[1] Institute for Data Analysis and Visualization, University of California, Davis, USA
[2] Department of Computer Science, University of California, Davis, USA
[3] Computer Graphics and Visualization, Chemnitz University of Technology, Germany

Abstract
We present a reverse engineering method for constructing a surface approximation scheme whose input is a set of unorganized noisy points in space and whose output is a set of quadric patches. The local surface properties, necessary for the subsequent segmentation, are estimated directly from the data using a simple and efficient data structure - the neighborhood graph. Our segmentation scheme, based on principal curvatures, constructs initial point sub-sets, which may be enlarged or further subdivided based on associated approximation error estimates obtained through approximation of the initial segments by quadric surfaces. Our method is highly efficient and produces a high-quality piecewise quadric surface approximation of engineering objects, which we demonstrate for several simple and complex example data sets.

1. Introduction

The problem of reconstructing a surface model from an unorganized set of points measured from the surface of a given object arises in many areas including computer graphics, CAD/CAM, computer vision and reverse engineering [MM97, VMC97]. Building such a model is a problem of growing importance since efficient digital scanning technologies, like laser range scanning devices, become more easily available. Through the rapid development of accurate optical scanners point models are "cheap", moving the focus from object reconstruction based on traditional scattered data interpolation to more efficient approximation schemes for very large point sets. Before the points are approximated by surfaces, it is necessary to group the points into appropriate subsets, a process which is often referred to as segmentation [BMV01, BV02, MLM01, VMC97, VM02]. Segmentation is usually based on surface properties, which can be estimated from the data points. However, to implement an efficient and reliable segmentation of an arbitrary data set is a real challenge: due to noise in the data and non-uniform point sampling density the segmentation process can be ambiguous. This paper describes a new segmentation method for constructing quadric surfaces from unorganized point sets and provides detailed information regarding various problems that must be solved during constructing a meaningful segmentation.

In recent years, several papers dealt with the problem of segmentation of an object with subsequent surface extraction and approximation from different sources [AFS06, BMV01, BV02, BV04, GFA04, LZHM06], [LDB05, VFT07, VS05, WAFR99, WFRA98, WK05, YLL*05, YLW06]. An in-depth review of these papers is beyond the scope of this paper, but we refer the reader to a few selected references.

Werghi et al. [WAFR99, WFRA98] described a method for recovering and creating a complete model consisting of quadric surfaces. Instead of fitting separate quadric surfaces they took into account the relationship between adjacent surface patches incorporating geometric constraints into the fitting process. The algorithm expects as input depth images coming from range scanning. Other algorithms base the segmentation and fitting on meshes; typically triangular meshes are used. A good survey of methods (until 2002) for recovering quadric surfaces from meshes was provided by Petitjean [Pet02]. Guillaume et al. [GFA04] presented a curvature tensor-based decomposition of an object into surface patches with near to constant curvature. The boundaries of the surface patches are rectified in order to obtain nicely shaped segments. Wu and Kobbelt [WK05] used a variational framework for a subdivision of a mesh into so-called "proxies", which are approximated by planes, spheres, cylinders and rolling-ball blends. Yan et al. [YLW06] based their variational shape approximation, similar to Wu's method, on an $L^{2,1}$ metric and extended it for all quadric surface types. Lai et al. [LZHM06] presented a top-down hierarchical feature-based segmentation suitable for large models at multiple scales. First, the mesh points are mapped into $I\!R^6$ and subsequently a hierarchical feature-sensitive re-meshing step is performed. Finally, the algorithm segments the object using k-means clustering. Another feature-sensitive mesh segmentation was presented by Yamauchi et al. [YLL*05] and is based on clustering of the normal vectors, using a modification of a general purpose technique called "mean shift". Lavoué et al. [LDB05] proposed a segmentation of a mesh into regions with nearly constant curvatures using a region growing algorithm taking seed clusters provided by the K-means algorithm. During the postprocessing the boundaries of the resulting segments are rectified based on a *Boundary Score* error metric. Attene et al. [AFS06] used a hierarchical mesh segmentation based on a cost function for extraction of algebraic surfaces from the mesh (they extract planes, spheres and cylinders). Their general purpose greedy algorithm has different applications, e.g. mesh denoising and fairing or segmentation for character animation. Vieira and Shimada [VS05] proposed another complex framework for mesh segmentation based on region growing: they carefully select seed regions according to the surface properties and noise estimation from the data, which are in the next step parameterized and approximated with bicubic Bézier patches. The segments are then extended by adding compatible points to the seed regions.

The work of Benkö et al. [BMV01, BV02, BV04] seems to be the mostly related to our method, as they do not expect meshes and base the segmentation directly on points. However, there are some important differences:

- Benkö et al. based their segmentation and surface extraction on normal vectors. Our approach is primarily based on principal curvatures. Normal vectors are used only as additional information.
- The classes of surfaces to be reconstructed are different: we consider all quadric surface types and the torus. Benkö et al. focused on circular quadric surfaces and on translational, rotational and sweep surfaces.

Our surface reconstruction is illustrated in Fig. 1. We assume that objects to be reconstructed consist mostly of quadratic algebraic surfaces. The target input for our method are non-free-form engineering objects and parts. The most frequent types of surface patches of technical CAD objects are parts of quadric surfaces like spheres, cylinders and cones. In order to be able to process and segment a wide class of CAD objects, we do not restrict the reconstruction to circular surfaces (surfaces of revolution), but we are able to recognize all quadric and circular toroidal surface patches on objects.

The given points are organized using a simple data structure: *the neighborhood graph*. This graph provides us with connectivity and adjacency information of the data points. Moreover, the points locally approximate the surface shape, so they are used for the estimation of surface properties. "Sharp edges" negatively bias the approximation of surface properties, and therefore we improve the surface properties estimation with a procedure consisting of detection of sharp edges (based on the information provided by the first-order segmentation) and of re-estimation of the surface properties on a modified neighborhood excluding neighbors behind sharp edges. If necessary, the combination of the first-order segmentation and normal vector re-estimation can be repeated several times in order to achieve better results. The following high-order segmentation step subdivides the point set into initial point clusters and uses a surface fitting step for validation of the created clusters. If the segment satisfies specified approximation criteria, it is accepted; otherwise, the segment is further subdivided.

In Fig. 2, the reconstruction process is demonstrated for a data set. Note that the triangulation in this figure and in the whole paper as well is created after the reconstruction and is for visualization purposes only. Based on the estimated normal vectors **(b)** *top*, the first-order segmentation is performed **(c)**, which makes possible detection of sharp edges. The normal vectors close to these sharp edges are re-estimated **(b)** *bottom*. Based on the estimated principal curvatures **(d)** the final, second order segmentation, controlled by the quadric surface fitting, is started resulting in the final reconstructed object **(e)**.

In Section 2, we describe a stable estimation scheme for the normal vectors and principal curvatures. As principal curvatures are more sensitive to noise than normal vectors, we implemented and compared the estimation quality of several methods for points originating from algebraic surfaces.

In Section 3, we summarize the functionality and the benefit of our first-order segmentation, in detail presented

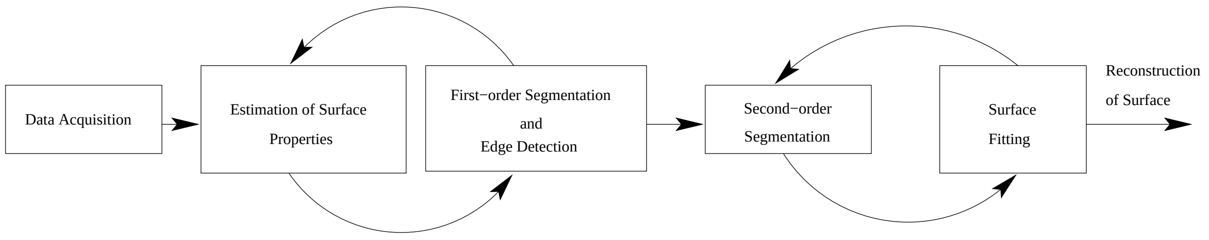
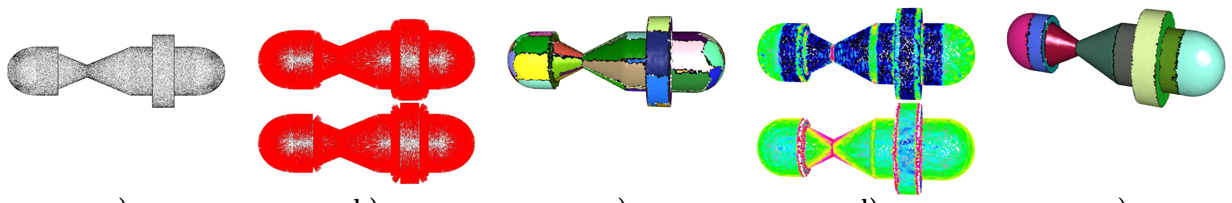

Data Acquisition
Estimation of Surface Properties
First−order Segmentation and Edge Detection
Second−order Segmentation
Surface Fitting
Reconstruction of Surface

Figure 1: *Reconstruction process.*



a) **b)** **c)** **d)** **e)**

Figure 2: *Reconstruction process:* ***a)*** *unorganized point cloud;* ***b)*** *normal vectors - top: initial estimation, bottom: normal vectors after re-estimation;* ***c)*** *first-order segmentation;* ***c)*** *curvature plot - top: minimum curvatures, bottom maximum curvatures;* ***d)*** *reconstructed object.*

in [VB04], especially for the detection of sharp edges and the re-estimation of the surface properties of points close to these edges.

Section 4 provides a detailed description of our *top-down* segmentation procedure controlled by the surface fitting. We present a method for the computation of the distance of a point to the quadric surface and compare recent linear methods for approximation of a given points set by quadrics and discuss their stability.

We present results and computation times of our surface reconstruction approach in Section 5, and demonstrate it for synthetically created point set data contaminated with random noise and one real data set obtained with a high-end optical scanner.

2. Estimation of surface properties

As input we assume n unorganized points $\mathcal{M} = \{\mathbf{P}_1, \ldots, \mathbf{P}_n\}$ which were acquired from the object's surface during a scanning process. Dependent on the accuracy of the scanning device, the digitized surface may be more or less noisy. Apart from the points we do not require any other information. Local surface properties, like normal vectors, principal curvatures or curvature directions, have to be estimated. We base the estimation process and the whole subsequent segmentation and surface fitting on a simple data structure: *the neighborhood graph.* Compared to a *global triangulation* of the point set [ACK01, BMR*99, BC00, DGH01, DS05, EM94, HDD*92, OBA*03], which is in general hard to compute and can be ambiguous depending on sampling density and noise level [ACK01, DS05], the k-nearest neighbors can be computed very efficiently [VB07, VBS99]. For objects with insufficient sampling on concave areas, Euclidean neighborhoods can contain incorrect neighbors (from the other side on the concave area). Here, k-nearest geodetic neighbors are preferred [VB07]. In the following, we denote the neighborhood of a point $\mathbf{P}$ as $\mathcal{N}(\mathbf{P})$.

The computation of the normal vector $\mathbf{N}(\mathbf{P})$ of a point $\mathbf{P}$ can be done locally using the computed neighborhood in a two-pass procedure: in the first pass, a best-fit plane $\mathcal{B}$ of the neighborhood $\mathcal{N}(\mathbf{P}) = \{\mathbf{Q}_1, \ldots, \mathbf{Q}_k\}$ is determined as $\sum_j \langle \mathbf{Q}_j - \mathbf{C}, \mathbf{N}(\mathbf{P})\rangle \to \min$ with the constraint $\|\mathbf{N}(\mathbf{P})\| = 1$, where $\mathbf{C}$ is the center of mass of the neighborhood $\mathcal{N}(\mathbf{P})$. Here, $\langle ., . \rangle$ denotes the dot product of two vectors. In the second pass, a new local coordinate frame is defined $(\mathbf{P}, \mathbf{X}, \mathbf{Y}, \mathbf{N}(\mathbf{P}))$, where $\mathbf{X}$ and $\mathbf{Y}$ are two arbitrary orthogonal vectors in the plane $\mathcal{B}$, and $\mathcal{N}(\mathbf{P})$ is approximated in this new coordinate frame by a quadratic or a cubic polynomial $z = f(x, y)$. With increasing noise level we prefer quadratic polynomials and a larger neighborhood, as a quadratic function has worse approximating but better smoothing behavior than a cubic one [VB07].

Stable estimation of principal curvatures and curvature directions is an interesting research topic [DB02, Ham93, GI04, KLM98, RB05, Rus04, TT05]. Hamann [Ham93], for example, approximated the natural neighbors of a point, where an explicit triangulation is known - the "platelet" - with a bivariate function $z = f(x, y)$. Krsek et al. [KLM98] provided an analysis and comparison of the estimation quality of three different methods: **a)** approximation by circles, **b)** Hamann's method and **c)** a Dupin cyclide method. Douros and Buxton [DB02] approximated the neighborhood with an implicit polynomial of second order (a quadric surface). Rusinkiewicz [Rus04] presented an efficient finite-differences approach for irregular triangle meshes that resembles the common method for estimating per-vertex normals on meshes. Goldfeather et al. [GI04] extended the approximation of the neighborhood with a bivariate function (generalization of Hamann's method for neighborhoods) with estimated normal vectors. Razdan and Bae [RB05] proposed a Bézier patch-fitting method, computing the curvatures from biquadratic Bézier patches that are locally fit to the data.

We compared several methods for synthetically created point sets with known exact principal curvatures. Every test was applied to a data set without noise, which was subsequently contaminated with *light* and *medium* noise levels. Note that the noise level is related to the average edge length of the six nearest neighbors to each point of the data set: $e_{\text{ave}} = \frac{1}{n}\sum_i^n \frac{1}{6}\sum_j^6 \|\mathbf{P}_i - \mathbf{Q}_j\|$, where $\mathbf{Q}_j \in \mathcal{N}(\mathbf{P}_i)$. Perturbing the points with a random noise with an amplitude up to 40% of e_{ave} is considered *light* noise, up to 80% of e_{ave} *medium*, up to 120% *strong* and above 120% *extreme*.

A slight modification of Goldfeather's approach turned out to work well for all tested noisy data sets:

$$\sum_i w_i \begin{pmatrix} 1 & x_i & y_i & x_i^2 & x_i y_i & y_i^2 & x_i^3 & x_i^2 y_i & x_i y_i^2 & y_i^3 \\ 0 & 1 & 0 & 2x_i & y_i & 0 & 3x_i^2 & 2x_i y_i & y_i^2 & 0 \\ 0 & 0 & 1 & 0 & x_i & 2y_i & 0 & x_i^2 & 2x_i y_i & 3y_i^2 \end{pmatrix} \mathbf{K} = \sum_i w_i \begin{pmatrix} z_i \\ -\frac{n_i^x}{n_i^z} \\ -\frac{n_i^y}{n_i^z} \end{pmatrix}$$

where $(x_i, y_i, z_i)^{\mathrm{T}}$ are the coordinates of the ith neighbor of $\mathbf{P}$, $(n_i^x, n_i^y, n_i^z)^{\mathrm{T}}$ are the coordinates of its transformed normal vector, $\mathbf{K} = (a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9)^{\mathrm{T}}$ is the coefficient vector, and $w_i = \max(\langle \mathbf{N}(\mathbf{P}), \mathbf{N}(\mathbf{Q}_i)\rangle, 0)$ is our introduced weight. As the neighborhood is transformed to the local coordinate frame, $\mathbf{N}(\mathbf{P}) = (0, 0, 1)^{\mathrm{T}}$, the weight can be rewritten to $w_i = \max(n_i^z, 0)$.

The weight introduced into the estimation algorithm guarantees more stability, especially for objects containing many sharp edges or areas with high curvature, assigning low weights to these points. Due to more degrees of freedom, cubic polynomials tend to approximate the neighborhoods better than quadratic ones. Nevertheless, with increasing noise level the approximation using cubic polynomials provides worse results. Quadratic approximation smooths the estimates, suppressing the noise, when it becomes significant relative to the neighborhood size. Up to light noise levels a cubic polynomial is a better choice and provides good results; for higher noise levels we switch to the quadratic approximation as it prevails regarding approximation accuracy over the cubic one. At the beginning of the whole reconstruction process, the user has to specify an estimate of the noise level in the data E_n. This value is then used for the computation of the parameters (method, neighborhood size, etc.) for estimation of all surface properties and later for the segmentation procedure.

As stated by other researchers before, the neighborhood size is the crucial parameter for stable curvature estimation. While small neighborhoods are suitable for clean data, larger neighborhoods tend to be less sensitive in the presence of noise by smoothing the estimates. In our application, the neighborhood size is adjusted according to the user-specified parameter E_n.

3. First-order segmentation

The goal of a segmentation is to subdivide the point set into "meaningful" parts according to surface properties. During the first-order segmentation step the point set is split according to the identified discontinuities in the information related to the surface first-order derivatives, the normal vectors. We use a *region-growing* algorithm with two segmentation thresholds [VB04]. In Fig. 3, the differences between the first-order (left image of **(a)** and top image of **(b)**) and second-order (right image of **(a)** and bottom image of **(b)**) segmentation with fixed thresholds are shown. The point set **(a)** was contaminated with *light* noise, **(b)** with *medium* noise.

The first-order segmentation provides reliable information about sharp edges and areas with high curvature in the point set. These highly curved regions cannot be in general approximated well by a smooth quadratic or cubic polynomial and therefore the normal vectors of points close to these sharp features are often estimated inaccurate.

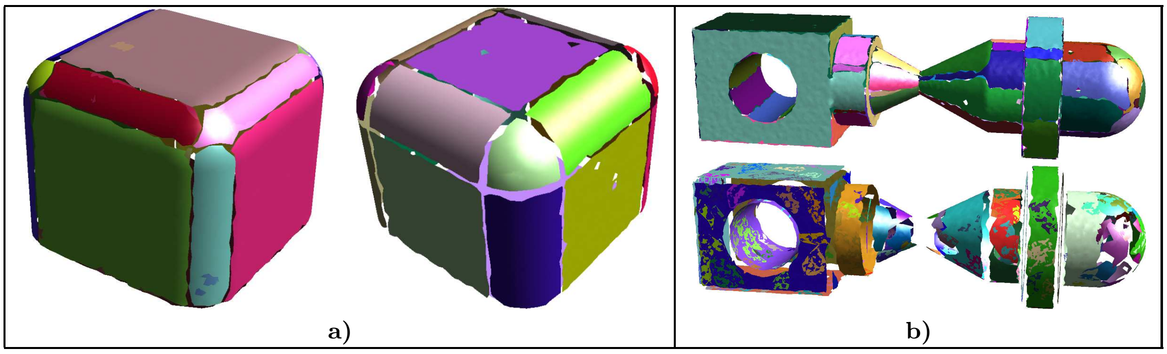


Figure 3: *Examples of first- and second-order segmentation with fixed thresholds.*

Using the first order segmentation we can detect locations with sharp edges in the data set and re-estimate the normal vectors of points close to these edges by temporarily excluding the points behind the edge from the neighborhood and re-constituting the neighborhood with neighboring points only on one side of the edge. The idea of the re-estimation is sketched in Fig. 4. The normal vector re-estimation leads to a significant improvement of the accuracy of the normal vectors for objects with many sharp edges [VB04].

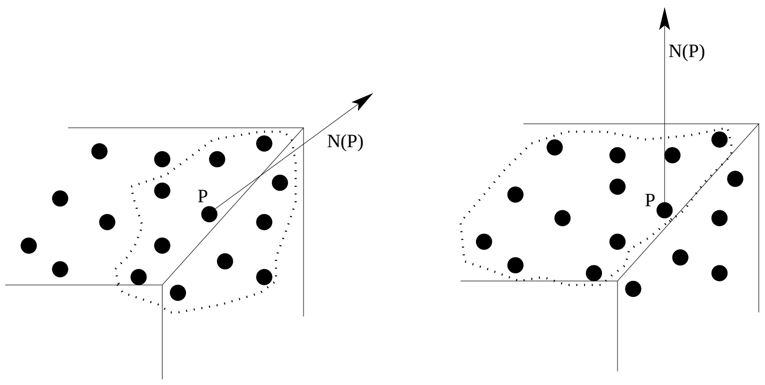


Figure 4: *Re-estimation of normal vectors of points close to sharp edges.*

4. Higher-order segmentation and surface fitting

The first-order segmentation is suitable for the detection of sharp edges, i.e. C^1 discontinuities, but it has a few drawbacks due to insufficient information provided by the normal vectors:

- It is unable to detect tangent-continuous, but curvature-discontinuous transitions between two algebraic surfaces - like a G^1-transition between a cylinder and a sphere, see Fig. 5 **a)**.
- Due to the segmentation parameters one algebraic surface can be subdivided into many segments, see Fig. 5 **b)**.
- It can happen, especially for noisy point sets, that G^1 discontinuous transitions are not detected, like the transition between a cylinder and a cone shown in Fig. 5 **c)**.

Due to these disadvantages of the first-order segmentation and the fact that the normal vectors do not provide enough information to segment and recognize algebraic surfaces reliably in a straightforward manner, higher-order surface properties - principal curvatures - are additionally used for the subsequent segmentation and surface fitting. We note that principal curvature-based segmentation may be unable to detect sharp edges if the principal curvatures of points close to sharp edges were exact, e.g. in the case of two perpendicular planes coming together. In order to handle correctly this possible but very improbable case, the angle between normal vectors of two adjacent points must fulfill the angle criterion from the first-order segmentation.

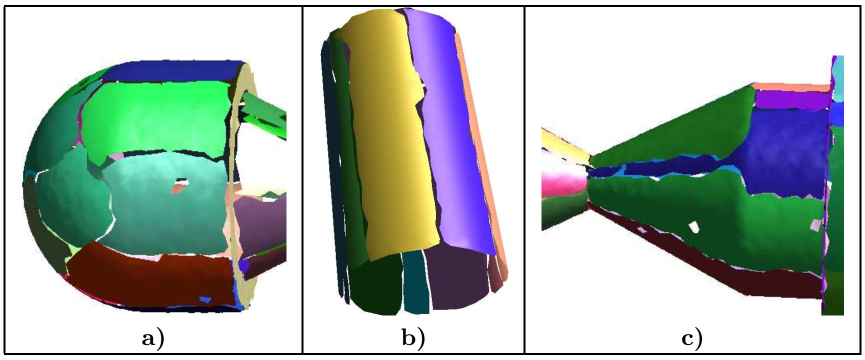


Figure 5: *Problems during the first-order segmentation.*

Compared to normal vectors, principal curvatures offer more information about the local behavior of the surface, but they have also disadvantages:

- They are more sensitive to noise in the data.
- They are not uniformly scale-invariant like normal vectors. While for the first-order segmentation the segmentation parameters work for almost all point sets, the curvature thresholds necessary for the second-order segmentation as presented in [VB04] have to be adjusted to object size; they can even vary within one object.

To deal with the first problem we compared several recently published curvature estimation methods described in Section 2. We chose the Goldfeather's approach [GI04] as it supports stable curvature estimation. If the noise level exceeds a threshold we apply an appropriate smoothing method to the points [VB07]. To solve the second problem we designed a new segmentation procedure, which automatically subdivides an object into general quadric surfaces with minimum user interaction.

The overall structure of the segmentation procedure is outlined in Algorithm 1. The algorithm supports multiple passes with each pass improving and enlarging existing recognized segments that already have been successfully approximated with a quadric surface, and creating new segments fulfilling the segmentation criteria. The main part of the whole segmentation process is described in Alg. 2. It is based on a top-down subdividing approach of the point set based on *region-growing*. Sorting the points with regard to their principal curvatures in the beginning guarantees that the segmentation procedure starts with points on simple surfaces, e.g. points on planes, cylinders or cones. Points on or close to sharp edges or highly curved areas where the curvature estimates can be quite inaccurate are processed at the end, with a high probability that only a few of these points remain unmarked, as they can be assigned to already recognized segments during segment enlargement.

One general problem of a region-growing algorithm is a proper specification of thresholds, e.g. a good termination criterion for the growth of a segment. Determining universal thresholds for the principal curvatures is complicated by the fact that principal curvatures depend on the object size. Furthermore, with increasing noise level estimated curvatures are smoothed by increasing the neighborhoods, which makes the estimation more robust against noise but unfortunately blurs transitions between two algebraic surfaces. Therefore, the segmentation thresholds have to be estimated directly from the curvature differences on such transitions between surfaces. Unfortunately, these transitions are not known a priori and their estimation directly from the data, e.g. from curvature derivatives [BPK98, YBS05], is a complex process typically requiring substantial user interaction [YBS05]. Furthermore, computation of curvature derivatives is extremely sensitive to noise.

As the goal of our segmentation procedure is to minimize user interaction and the number of parameters to be provided, and to produce a stable subdivision of the object into distinct quadratic algebraic patches, a top-down segmentation approach has been chosen. The top-down approach allows us to start with arbitrary thresholds and, after checking the segmentation using surface fitting, either to accept the segmentation or decrease the thresholds and refine the segmentation.

The initial segmentation thresholds are derived from the mean value of all principal curvatures of processed points, and a segment S_k is created using these thresholds. The points of S_k are approximated by a quadric surface, and the average distance D_a of the points to the fitted surface is computed. If $D_a < T_s$, S_k is accepted and enlarged. Note that we handle a torus using an extra procedure, following [VB04]. When a torus is detected, the torus parameters are estimated and a special enlarging procedure for the torus case is called. During general enlarging, additional neighboring points of S_k are added to it, as long as their distance to the fitted surface stays within T_s and they do not belong to any other segment. T_s is the only segmentation threshold the user has to provide, ranging from 0.05 for "clean" data set to one for extremely noisy data. If no more points can be added to S_k, all points of S_k are marked and the region-growing procedure proceeds with the next unmarked seed point from the sorted array. If $D_a > T_s$, S_k is released, the segmentation thresholds are decreased, and the region-growing procedure creates a new segment S_k.

Compared to region-growing with fixed thresholds [VB04], our procedure can process (segment and approximate by a quadric) all points several times, if created segments are rejected due to large initial thresholds, leading to longer computation times. The most time-consuming part is the computation of the average distance of the points in a segment to the corresponding approximated quadric surface, which is non-linear and iterative, see Section 4.1. The times for the segment creation and approximation procedures can be neglected compared to the average distance computation times. But not all distances from all segment points to the fitted surface have to be computed in order to determine that a segment should be rejected. Thus, segment rejection can be optimized by testing the average distance after a few distance computations.

Algorithm 1 Segmentation Algorithm

```
1. Unmark points in non-regular segments. (We call a segment regular when it contains more points than a prescribed number.)
2. Perform Algorithm 2 described below.
3. Sort all segments in descending order according to their numbers of points.
4. For all regular segments $S_k, k = 1, \dots$ do
   Let $\mathbf{Q}$ be a neighbor of $S_k$ and $\mathcal{Q}(S_k)$ be the quadric surface which approximates $S_k$ (Alg. 2). If $d(\mathbf{Q}, S_k) < T_s$ and for segment $S_m$ containing $\mathbf{Q}$ applies $m > k$ (i.e. $S_m$ is processed later than $S_k$), remove $\mathbf{Q}$ from $S_m$ and insert it into $S_k$.
5. If the number of points in all non-regular segments is higher than $\lambda n$, where $n$ is the number of all points and $\lambda$ is a parameter (typically we choose $\lambda = 1\% - 3\%$), then slightly increase $T_s$ and repeat the whole procedure. Note that Algorithm 2 takes as input only points of non-regular segments.
6. If the number is less than $\lambda n$, then for all points in all non-regular segments do:
   • Pick a point $\mathbf{P}_i$.
   • Find all neighboring segments $\mathcal{S} = \{S_1, \dots, S_j\}$ in the vicinity of $\mathbf{P}_i$.
   • Find in $\mathcal{S}$ the segment $S_k$ with the shortest orthogonal distance from a neighboring point $\mathbf{Q}$ to the fitted surface of $S_k$.
   • If the distance is less than $2T_s$, remove $\mathbf{Q}$ from the current segment and add it to $S_k$.
   • Repeat until non-regular segments are empty or the distances from the remaining points to the nearest segment is greater than $2T_s$.
   • If there are points remaining, remove them from the data set, as they are possible outliers.
```

Setting a proper value for the segmentation parameter T_s is very important, and it affects not only the number and the shape of resulting segments, but the processing time as well. Choosing a small value for T_s causes many segments refusals after surface fitting. In the worst case, after one pass of Alg. 2 all points belong to non-regular segments. Then, T_s is slightly increased and Alg. 2 is repeated. It is desirable that T_s is larger than the maximum noise amplitude. Otherwise, even correctly segmented and fitted algebraic surfaces may stay incomplete, as the distance of some noisy points to the surface is greater than T_s, as shown in Fig. 6 **(a)**.

Fig. 6 illustrates the behavior of the segmentation procedure using various values for T_s. Both point sets were contaminated with *medium* noise with a maximum amplitude of 50% of e_{ave}. For case **(a)**, the segmentation parameter was set $T_s = 0.1$ and for **(b)** it was $T_s = 0.3$. The segmentation procedure required 19.24 seconds and created 20 regular segments in case **(a)**, and 2.28 seconds and 11 regular segments in case **(b)**.

The data set in Fig. 6 **(c)**, **(d)** and **(e)** is more difficult to segment properly, as the point density of the data set is low. The segmentation parameter was $T_s = 0.1$, $T_s = 0.5$ and $T_s = 0.9$ for the cases **(c)**, **(d)** and **(e)** respectively. For case **(c)** 20 regular segments were produced in 9.56 seconds, for case **(d)** 24 regular segments in 2.08 seconds, and for case **(e)** 19 regular segments in 0.56 seconds.

Algorithm 2 Segmentation and Fitting Algorithm

1. Sort all unmarked points lexicographically according to their principal curvatures, i.e. Index($\mathbf{P}_i$) < Index($\mathbf{P}_j$) if $\kappa_i^{\min} < \kappa_j^{\min}$ or $\kappa_i^{\min} = \kappa_j^{\min} \wedge \kappa_i^{\max} < \kappa_j^{\max}$. Set the initial segmentation thresholds according to the mean value of the principal curvatures of all points: $T_{\text{init}}^{\min} = \gamma \frac{1}{n} \sum_{i=1}^{n} \kappa_i^{\min}$ and $T_{\text{init}}^{\max} = \gamma \frac{1}{n} \sum_{i=1}^{n} \kappa_i^{\max}$, where γ is a constant.
2. Pick the first unmarked point from the sorted array and create a regular segment S_k using a simple and efficient region-growing algorithm.
3. Approximate S_k by a general quadric surface. We denote the surface by $\mathcal{Q}(S_k)$.
4. Compute an average distance D_a of the points of S_k to the fitted surface, i.e. $D_a = \frac{1}{n_k} \sum_i d\left(\mathbf{P}_i, \mathcal{Q}(S_k)\right)$.
5. If $D_a > T_s$ (T_s being a user-defined threshold), reduce the segmentation thresholds and go to step 2.
6. Otherwise, enlarge S_k by inspecting its neighboring points which do not belong to any other segment (unmarked points): for a neighbor $\mathbf{Q}$, if $d\left(\mathbf{Q}, \mathcal{Q}(S_k)\right) < T_s$, add $\mathbf{Q}$ to S_k. At the end of the enlarging process, all points of S_k are marked and the procedure proceeds with the next unmarked point using the initial segmentation thresholds.
7. If points have been added to S_k repeat surface fitting. i.e. go to step 3.

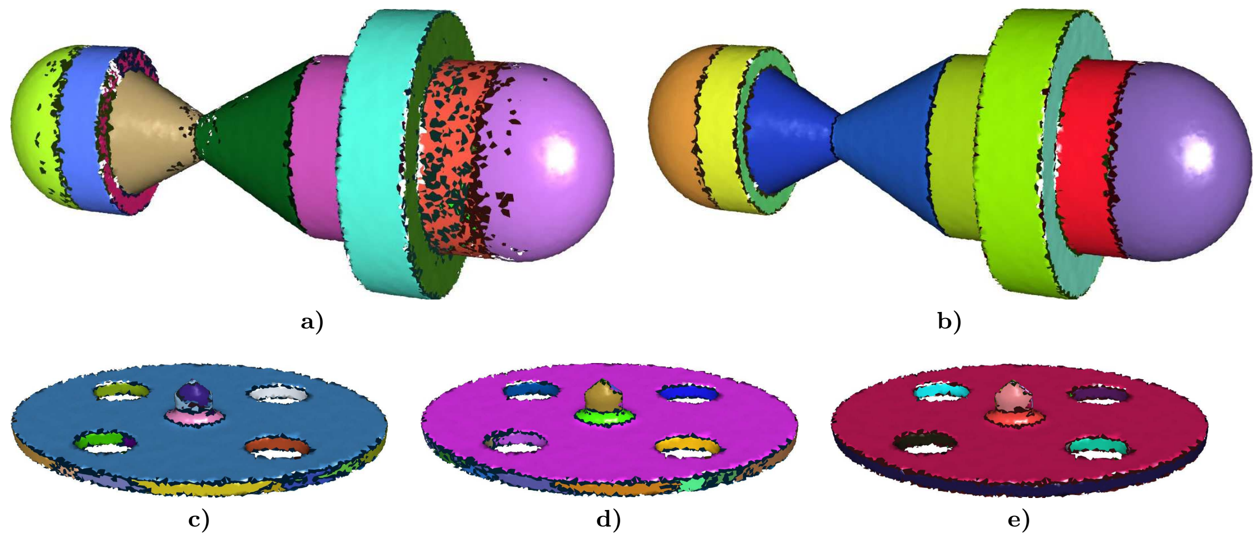


Figure 6: *Segmentations obtained by different values of the segmentation parameter T_s.*

4.1. The orthogonal distance

While the computation of the orthogonal distance from an arbitrary point $\mathbf{P}$ to special cases of quadrics like planes, circular spheres, cylinders, cones or of the quartic surface like circular tori is a relatively simple computational task, the orthogonal distance to a general quadric is much more complex and it cannot be found analytically. We consider the quadric surface equation in the form

$$\begin{aligned} \mathrm{F}(\mathbf{x}) = ax^2 + by^2 + cz^2 + 2fxy + 2gyx + 2hzx + 2px + 2qy + 2rz + d &= \\ \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} + 2\mathbf{B}^{\mathrm{T}}\mathbf{x} + d &= 0, \end{aligned} \tag{1}$$

where $\mathbf{A}$ is the 3x3 symmetric matrix $\mathbf{A} = \begin{pmatrix} a & f & h \\ f & b & g \\ h & g & c \end{pmatrix}$ and $\mathbf{B}$ is the vector $(p, q, r)^{\mathrm{T}}$.

The normal vector $\mathbf{N}$ at a point $\mathbf{x}$ is given by $\mathbf{N}(\mathbf{x}) = \frac{\nabla \mathrm{F}(\mathbf{x})}{\|\nabla \mathrm{F}(\mathbf{x})\|} = \frac{\mathbf{Ax}+\mathbf{B}}{\|\mathbf{Ax}+\mathbf{B}\|}$.

To obtain the nearest point on $\mathrm{F}(\mathbf{x})$ to $\mathbf{P}$ we intersect a line $g : \mathbf{x} = \mathbf{P} + t\nabla\mathrm{F}(\mathbf{x}) = \mathbf{P} + t(\mathbf{Ax}+\mathbf{B})$ in the orthogonal direction to $\mathrm{F}(\mathbf{x})$ with the quadric surface. The point $\mathbf{x}$ on g can be expressed as:

$$\mathbf{x} = (\mathbf{I} + t\mathbf{A})^{-1}(\mathbf{P} + t\mathbf{B}). \tag{2}$$

Substituting (2) (after singular value decomposition) into $\mathrm{F}(\mathbf{x})$ yields an equation of order 6 at most.

It is not acceptable to search numerically the six solutions in order to find the shortest distance. Therefore, numerical iterative approaches are usually used. One can iteratively minimize the quadratic distance $d(\mathbf{x}) = \|\mathbf{P} - \mathbf{x}\|^2$ with a constraint $\mathrm{F}(\mathbf{x}) = 0$ using Lagrange multipliers:

$$\begin{aligned} D(\lambda, \mathbf{x}) &= \|\mathbf{P} - \mathbf{x}\|^2 - \lambda \mathrm{F}(\mathbf{x}) \rightarrow \min \qquad \text{with} \\ \nabla D(\lambda, \mathbf{x}) &= 0. \end{aligned} \tag{3}$$

Equation (3) can be solved by using, for example, a globally convergent quasi-Newton method. We observed that a correct convergence of equation (3) highly depends on the starting point $\mathbf{x}_0$ and the value λ_0.

As every point of the underlying point set has an associated estimated normal vector, we intersect a line through $\mathbf{P}$ in the direction of its normal vector with the quadric surface $\mathrm{F}(\mathbf{x})$ to obtain a suitable starting point $\mathbf{x}_0$. Unfortunately, for outliers and some special cases the intersection can yield an empty set. In these cases we repeat the intersection using the gradient of the quadric in $\mathbf{P}$, i.e. $\nabla\mathrm{F}(\mathbf{P})$. $\nabla\mathrm{F}(\mathbf{P})$ provides a reasonable approximation of the normal vector only if $\mathbf{P}$ is close to $\mathrm{F}(\mathbf{x})$. If this second attempt returns no intersection, we set $\mathbf{x}_0 = \mathbf{P}$.

Tests have shown that the correct estimation of λ_0 is crucial for the convergence of the equation (3). If the starting point $\mathbf{x}_0$ lies on the quadric, i.e. we get an intersection point using the associated normal vector or gradient in $\mathbf{P}$, we set

$$\lambda_0 = \frac{2}{3}\left(\frac{(\mathbf{P} - \mathbf{x}_0)_x}{\nabla\mathrm{F}(\mathbf{x}_0)_x} + \frac{(\mathbf{P} - \mathbf{x}_0)_y}{\nabla\mathrm{F}(\mathbf{x}_0)_y} + \frac{(\mathbf{P} - \mathbf{x}_0)_z}{\nabla\mathrm{F}(\mathbf{x}_0)_z}\right). \tag{4}$$

This heuristics leads in almost all cases to a globally minimal distance. In some rare cases of divergence λ_0 has to be slightly decreased.

The choice of λ_0 is more critical when the starting point $\mathbf{x}_0 = \mathbf{P}$. Here, we observed more divergence cases, even when using different starting values for λ_0. Minimization of an extended Lagrangian function $D(\lambda, \alpha, \beta, \mathbf{x})$ containing an additional constraint $\nabla\mathrm{F}(\mathbf{x}) \times (\mathbf{P} - \mathbf{x}) = 0$ provided more stable results for these cases using the same strategy for estimation of the starting values. We prefer minimization of (3) since we need to minimize only four variables in that approach instead of six (the computation of the distance is the most time-consuming part of the whole segmentation process).

4.2. Surface fitting with quadrics

Approximation of points with implicit polynomials is a well-studied field in geometric modeling. The general approach minimizes the sum of squared algebraic distances:

$$g_a = \sum_i \mathrm{F}(\mathbf{x}_i)^2 \rightarrow \min\ , \tag{5}$$

with $\mathrm{F}(\mathbf{x}_i) = P^{\mathrm{T}}\mathbf{K}$, where P^{T} is the monomial basis of the implicit polynomial and $\mathbf{K}$ is the coefficient vector.

Unfortunately, an infinite number of coefficient vectors minimizes the expression (5), including the trivial solution $\mathbf{K} = 0$. Therefore, a minimization constraint for $\mathbf{K}$ should be specified. Different constraint schemes for (5) have been proposed, see for example [Alb74, Gna77, Pra87]. The most common of these schemes is based on normalization of the coefficient vector, i.e. $\mathbf{K}^{\mathrm{T}}\mathbf{K} = 1$. This leads to an eigenvalue problem, where the solution is the eigenvector associated with the smallest eigenvalue. However, the numerical stability of the solution is poor and the resulting implicit polynomial is extremely sensitive to changes of the coefficients; small changes of the coefficients may result in large fitting errors.

Our modification of (5) involves estimated normal vectors which are incorporated into the minimization process:

$$\begin{aligned} P^{\mathrm{T}}K &= 0, \\ \nabla\mathrm{F}(\mathbf{x}) \times \mathbf{N}(\mathbf{P}_i) &= 0. \end{aligned} \tag{6}$$

The fitting quality of this method is better than that of (5), especially for cylindrical and parabolic surfaces, but it still exhibits instability problems depending on noise and estimated normal vectors.

Taubin [Tau91] proposed to minimize the first-order Taylor approximation of the orthogonal distance instead of squared algebraic distances:

$$g_t = \sum_i \frac{\mathrm{F}(\mathbf{x}_i)^2}{\|\nabla \mathrm{F}(\mathbf{x}_i)\|^2}. \tag{7}$$

The minimization problem defined by (7) is non-linear and leads to an iterative evaluation with the need for a global search, which still does not guarantee convergence. Furthermore, (7) can become numerically instable when $\|\nabla \mathrm{F}(\mathbf{x}_i)\|^2$ is small. Taubin [Tau93] included into the orthogonal distance approximation higher-order partial derivatives at the points. However, this higher-order approximation remains still non-linear.

Ahn et al. [ARCW02] introduced a two-step approach for the approximation of point sets by implicit polynomials based on true orthogonal distance. In the first step, for every point $\mathbf{P}_i$ its orthogonal "contacting point" $\mathbf{P}'_i$ on the implicit surface is computed. In the second step, the sum of distances $\sum_i \|\mathbf{P}_i - \mathbf{P}'_i\|^2$ is minimized. Both steps are non-linear and computationally expensive.

Recently, two approaches for linear implicit polynomial approximation were published [BLCC00, HB04]. The authors' description of a *linear* approximation is that the solution can be found by $\mathbf{AK} = \mathbf{b}$, where $\mathbf{A}$ is a known matrix, $\mathbf{b}$ is a known vector, and $\mathbf{K}$ is the unknown coefficient vector.

The "3L algorithm" suggested by Blane et al. [BLCC00] is based on the construction of two additional level sets of $f(\mathbf{x}) = 0$ having distances $-\varepsilon$ and ε from the zero set. The goal of this approach is to find a polynomial with zero values at the original points and values $-\varepsilon$ and ε at the two additional constructed point sets below and above the original data:

$$g_b = \sum_i (P^{\mathrm{T}}(\mathbf{x}_i)\mathbf{K})^2 + \sum_i (P^{\mathrm{T}}(\mathbf{x}_i^U)\mathbf{K} - \varepsilon)^2 + \sum_i (P^{\mathrm{T}}(\mathbf{x}_i^L)\mathbf{K} + \varepsilon)^2 \quad , \tag{8}$$

where ε is a small positive constant and P^{T} is the monomial vector of the implicit polynomial.

Helzer et al. [HB04] presented another method - called "MinMax method" - for implicit polynomial fitting based on an in-depth analysis of the zero set sensitivity function. They examined the changes of the fitting error, assuming small changes of the polynomial coefficients, and established a zero set fitting error bound. On the contrary to the 3L method, the MinMax algorithm uses the points as well as the normal vectors.

Numerous tests have shown that the *MinMax* method performs best for most cases; it exhibits very good stability for all quadric surface types. Helzer et al. [HB04] demonstrated superior fitting quality of the *MinMax* method when compared to the 3L approach for curves of high degree in 2D. For surface fitting, the differences between *MinMax* and the 3L method are marginal; the 3L fitting approach suits for our purposes well. As both methods are linear in the number of points contained in a segment, computation time can be neglected when compared to the non-linear evaluation of the orthogonal distance function. Therefore, the points of every segment are approximated by a quadric surface twice using both presented methods, and we use the coefficient vector of the fit with the shortest average distance of the points to the quadric.

	Algebraic Distance	Orthogonal Distance
General LSF	0.00146943	0.02559303
Normal-constrained LSF	0.00148268	0.02342658
3L Method	0.00642055	0.00924565
MinMax	0.20496235	0.00924533

Table 1: *Average algebraic and orthogonal geometric distances of cylinder points to fitted surface.*

Fig. 7 illustrates the fitting quality of the discussed four linear methods for an elliptic cylinder, containing 4,613 points contaminated with medium noise. The corresponding algebraic and orthogonal distances are listed in Table 1. The points of the cylinder were projected onto the fitted quadric and triangulated for visualization purposes.

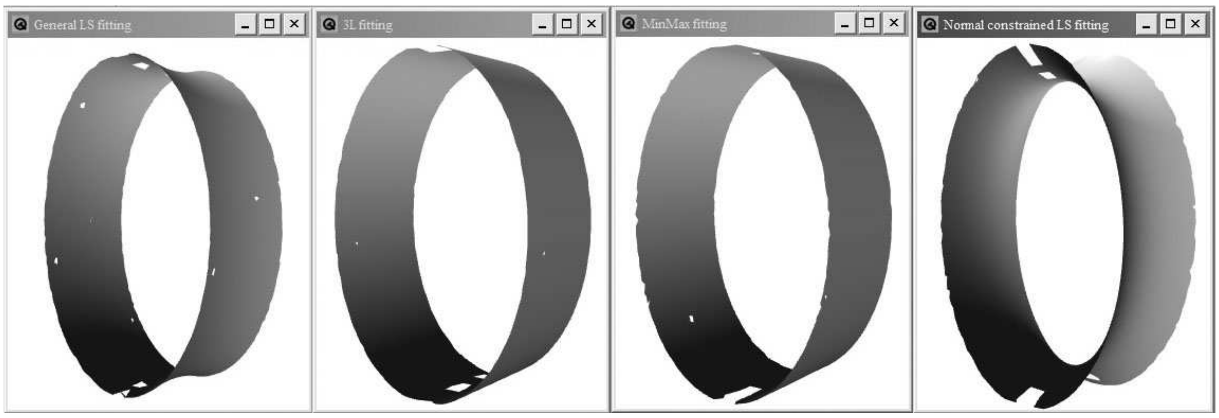


Figure 7: *Comparison of fitting quality of different methods. From left to right: minimization of squared algebraic distances; 3L method; MinMax method; normal vector-constrained minimization of squared algebraic distances.*

5. Results

We present segmentation results for synthetically created and scanned data sets. All synthetic data sets were contaminated with medium noise using an amplitude of at least 40% of e_{ave}. Furthermore, they were scaled using non-uniform scaling factors in order to obtain general quadric surfaces (elliptical surfaces).

In all cases, our reconstruction provided a complete decomposition of the point set into quadric surfaces and tori. For special cases (spheres and circular cylinders, cones and tori) we provide surface parameters like axes, radii, etc. The reconstruction pipeline is based on neighborhoods. Triangulation of segments was done for visualization purposes only. Holes in the segments are flaws in our triangulation procedure. In Fig. 9, left we have included a dialog showing the types of reconstructed surfaces and the number of points they contain.

Minor problems can arise during the reconstruction of the object shown in Fig. 8 **c**) - Revolution+Block. The transition between the green plane and the purple elliptic cylinder causes the problems. During processing of the green plane the enlarging procedure “steals” points from the cylinder, as they fulfill the distance criterion, i.e. the distance of the cylinder points to the plane equation is smaller than T_s. Even the built-in check requiring the normal vectors of two adjacent points to enclose a small angle, does not prevent this effect. For special cases of segments with constant, at least one, curvature we could perform additional tests of the curvature(s) of two adjacent points. But for general quadrics it is very difficult to detect or prevent assignment of points to wrong segments on such or similar transitions. Therefore, we assume that this disadvantage of our reconstruction can be dealt with manually by the user.

A general reconstruction problem involves a correct detection of C^1-continuous transitions between surfaces. Normal vectors do not provide enough information for it, and even detection using principal curvatures is not reliable due to smoothing of curvatures in such transition regions during curvature estimation. In our approach the shape of a C^1-continuous transition between surfaces is controlled by the noise. The higher the noise level, the greater the thresholds are set, leading to C^1-transitions that are less reliable, as demonstrated in Fig. 8 **d**). For moderate noise, see Fig. 8 **a**) and **c**), the C^1-transition was found correctly, without any outliers.

Processing times of the segmentation procedure together with surface fitting (excluding computation time for surface properties) are listed in Table 2. All tests were performed on a Windows XP based system with 2.16 GHz Intel Core 2 Duo T7400 CPU and 2x1GB SO-DIMM RAM, 533 MHz. Only the times for the first segmentation pass are listed. The processing times depends not only on the number of points but also on the number of constructed surfaces, the noise level and the user threshold T_s. When the user is not satisfied with the result, one can improve (enlarge) the segments by increasing the segmentation parameter T_s (Section 4). Usually, the second and further passes of the segmentation procedure introduce only small changes in the segmentation, as they operate on points which have not been assigned to any other regular segment before. The running times of the next passes are less than one second even for objects containing several hundred thousand points.

	Revolution	MechPart	Revolution+Block	Curved Box	ITW Object
No. of points	54,854	22,211	90,974	27,792	158,562
No. of quadric surfaces	11	28	21	19	34
Computation time	2.30	4.92	3.56	0.95	5.95

Table 2: *Computation times in seconds.*

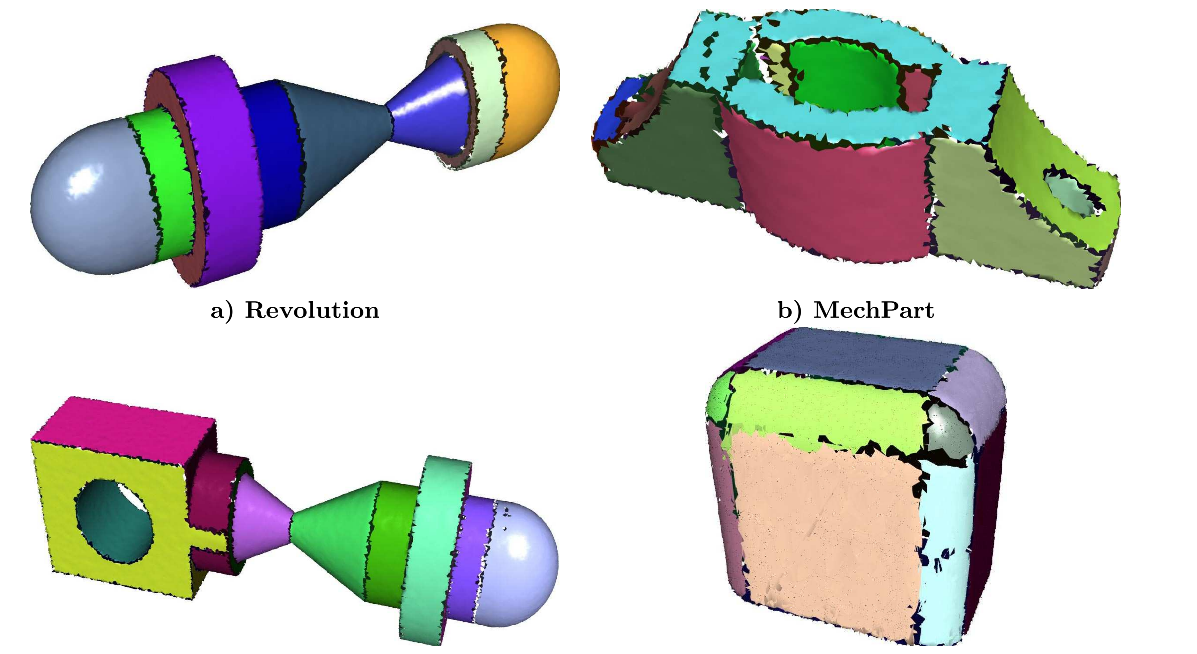
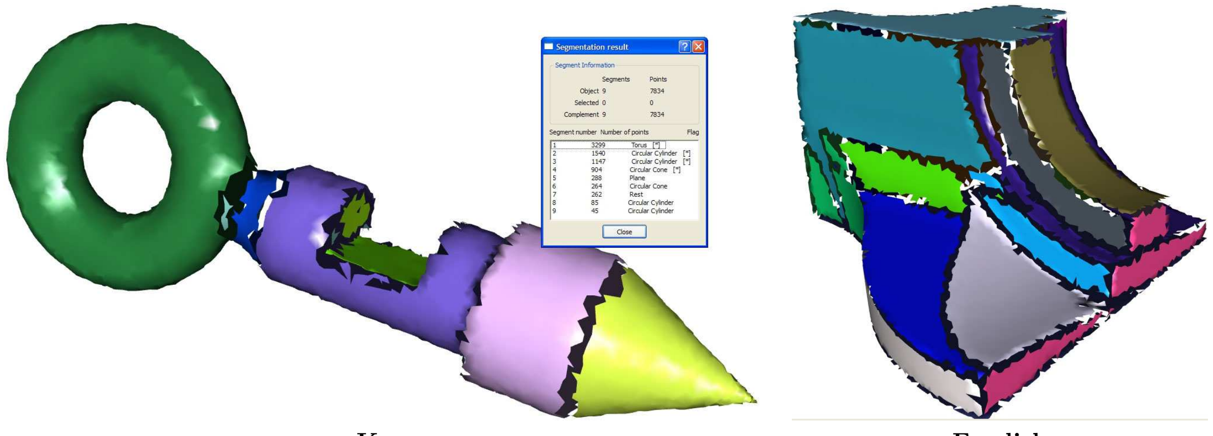


Figure 8: *Segmentation and reconstruction results of synthetic point sets contaminated with noise levels of 50% of* e_{ave} ***(a)**, **(b)**, **(c)** and 80% of* e_{ave} ***(d)**.*



	Segments	Points
Object	9	7834
Selected	0	0
Complement	9	7834

Segment number	Number of points		Flag
1	3299	Torus	[*]
2	1540	Circular Cylinder	[*]
3	1147	Circular Cylinder	[*]
4	904	Circular Cone	[*]
5	288	Plane	
6	264	Circular Cone	
7	262	Rest	
8	85	Circular Cylinder	
9	45	Circular Cylinder	

Figure 9: *Segmentation and reconstruction results of point sets contaminated with noise levels of 50% of* e_{ave} *(left), and 40% of* e_{ave} *(right).*

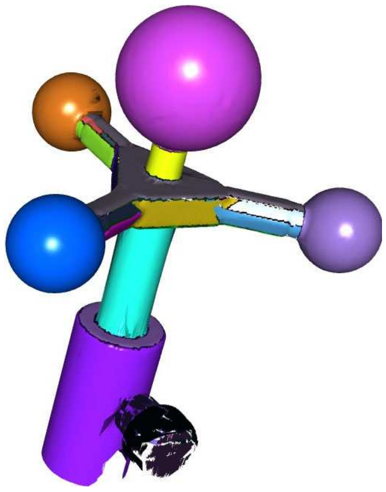


Figure 10: *Segmentation and reconstruction result of real data set (courtesy of ITW, Chemnitz, Germany).*

6. Conclusions and future work

We have presented a new method for constructing several quadric surface patches from an unorganized noisy point set. Geometrical surface properties, necessary for the segmentation, are estimated directly from the given points using methods that are stable even in the presence of high noise. For the segmentation procedure, a *top-down* approach combined with *region-growing* has been used in order to generate a meaningful and high-quality subdivision of the underlying surface. We also ensure that the number of parameters to be provided by user is small. The subsequent fitting of the segmented points by quadric surface patches provides a continuous approximation of the object's surface and it controls the segmentation procedure by returning needed feedback, that allows to decide, whether a segment was approximated well by a quadric or it should be further subdivided.

Although every point in the data set can be processed several times, our implementation of the segmentation procedure is efficient, processing on the order of 100,000 points within a few seconds, running on a computer with a 2GHz Intel Core Duo CPU. Through modification of the one user-specified segmentation parameter, the initial segmentation, when it was not satisfactory, can be improved easily and quickly, as the segmentation procedure is optimized for *multi-pass* processing.

The result of the segmentation strongly depends on the segmentation parameter T_s, which the user has to provide. We intend to analyze the noise in the data and provide a good estimation of T_s, in order to avoid unnecessarily long processing times.

7. Acknowledgments

We thank the members of the Visualization and Computer Graphics Research Group at the Institute for Data Analysis and Visualization (IDAV), at the University of California, Davis, for their support and valuable comments. The first author was supported by the German Research Foundation (DFG) by a post-doctoral fellowship.

References

[ACK01] Amenta N., Choi S., Kolluri R. K.: The power crust. In *6th ACM symposium on Solid Modeling and Applications* (2001), ACM Press, pp. 249–266.

[AFS06] Attene M., Falcidieno B., Spagnuolo M.: Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer: International Journal of Computer Graphics 22*, 3 (2006), 181–193.

[Alb74] Albano A.: Representation of digitized contours in terms of conic arcs and straight-line segments. *Computer Graphics and Image Processing 3* (1974), 23–33.

[ARCW02] Ahn S. J., Rauh W., Cho H. S., Warnecke H.-J.: Orthogonal distance fitting of implicit curves and surfaces. *IEEE Transaction on Pattern Analysis and Machine Intelligence 24*, 5 (2002), 620–638.

[BC00] Boissonnat J.-D., Cazals F.: Smooth surface reconstruction via natural neighbour interpolation of distance functions. In *16th Annual Symposium on Computational geometry* (2000), ACM Press, pp. 223–232.

[BLCC00] Blane M. M., Lei Z., Civi H., Cooper D. B.: The 3L algorithm for fitting implicit polynomial curves and surfaces to data. *IEEE Transaction on Pattern Analysis and Machine Intelligence 22*, 3 (2000), 298–313.

[BMR*99] Bernardini F., Mittleman J., Rushmeier H., Silva C., Taubin G.: The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics 5*, 4 (1999), 349–359.

[BMV01] Benkö P., Martin R. R., Várady T.: Algorithms for reverse engineering boundary representation models. *COMPUTER-AIDED DESIGN 33*, 11 (2001), 839–851.

[BPK98] Belyaev A., Pasko A. A., Kunii T. L.: Ridges and ravines on implicit surfaces. *Computer Graphics International 00* (1998), 530.

[BV02] Benkö P., Várady T.: Direct segmentation of smooth, multiple point regions. In *Geometric Modeling and Processing - Theory and Applications* (2002), IEEE Computer Society, p. 169.

[BV04] Benkö P., Várady T.: Segmentation methods for smooth point regions of conventional engineering objects. *Computer-Aided Design 36*, 6 (2004), 511–523.

[DB02] Douros I., Buxton B. F.: Three-dimensional surface curvature estimation using quadric surface patches. In *Scanning* (2002).

[DGH01] Dey T. K., Giesen J., Hudson J.: Delaunay based shape reconstruction from large data. In *IEEE symposium on parallel and large-data visualization and graphics* (2001), IEEE Press, pp. 19–27.

[DS05] Dey T. K., Sun J.: An adaptive mls surface for reconstruction with guarantees. In *3rd Eurographics Symposium on Geometry Processing* (2005), pp. 43–52.

[EM94] Edelsbrunner H., Mücke E. P.: Three-dimensional alpha shapes. *ACM Transactions on Graphics 13*, 1 (1994), 43–72.

[GFA04] Guillaume L., Florent D., Atilla B.: Curvature tensor based triangle mesh segmentation with boundary rectification. In *Computer Graphics International* (2004), IEEE Computer Society, pp. 10–17.

[GI04] Goldfeather J., Interrante V.: A novel cubic-order algorithm for approximating principal direction vectors. *ACM Transactions on Graphics 23*, 1 (2004), 45–63.

[Gna77] Gnanadesikan R.: *Methods for statistical data analysis of multivariate observations.* Wiley, 1977.

[Ham93] Hamann B.: Curvature approximation for triangulated surfaces. *Geometric Modelling, Computing Suppl. 8* (1993), 139–153.

[HB04] Helzer A., Barzohar M.: Stable fitting of 2d curves and 3d surfaces by implicit polynomials. *IEEE Transaction on Pattern Analysis and Machine Intelligence 26*, 10 (2004), 1283–1294.

[HDD*92] Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W.: Surface reconstruction from unorganized points. In *SIGGRAPH'92* (1992), ACM Press, pp. 71–78.

[KLM98] Krsek P., Lukács G., Martin R.: Algorithms for computing curvatures from range data. In *The Mathematics of Surface VIII* (1998), Information Geometers. Winchester, UK, pp. 1–16.

[LDB05] Lavoué G., Dupont F., Baskurt A.: A new CAD mesh segmentation method, based on curvature tensor analysis. *Computer Aided Design 37*, 10 (2005), 975–987.

[LZHM06] Lai Y.-K., Zhou Q.-Y., Hu S.-M., Martin R. R.: Feature sensitive mesh segmentation. In *ACM symposium on Solid and physical modeling* (2006), ACM Press, pp. 17–25.

[MLM01] Marshall D., Lukács G., Martin R. R.: Robust segmentation of primitives from range data in the presence of geometric degeneracy. *IEEE Trans. Pattern Anal. Mach. Intell. 23*, 3 (2001), 304–314.

[MM97] Mencl R., Müller H.: Interpolation and approximation of surfaces from three-dimensional scattered data points. In *Scientific Visualization* (1997), IEEE Computer Society, pp. 223–232.

[OBA*03] Ohtake Y., Belyaev A., Alexa M., Turk G., Seidel H.-P.: Multi-level partition of unity implicits. In *ACM SIGGRAPH* (2003), ACM Press, pp. 463–470.

[Pet02] Petitjean S.: A survey of methods for recovering quadrics in triangle meshes. *ACM Comput. Surv. 34*, 2 (2002), 211–262.

[Pra87] Pratt V.: Direct least-squares fitting of algebraic surfaces. In *ACM SIGGRAPH: 14th annual conference on Computer graphics and interactive techniques* (1987), ACM Press, pp. 145–152.

[RB05] Razdan A., Bae M.: Curvature estimation scheme for triangle meshes using biquadratic Bézier patches. *Computer-Aided Design 37*, 14 (2005), 1481–1491.

[Rus04] Rusinkiewicz S.: Estimating curvatures and their derivatives on triangle meshes. In *3D Data Processing, Visualization and Transmission* (2004), IEEE Computer Society, pp. 486–493.

[Tau91] Taubin G.: Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transaction on Pattern Analysis and Machine Intelligence 13*, 11 (1991), 1115–1138.

[Tau93] Taubin G.: An improved algorithm for algebraic curve and surface fitting. In *IEEE International Conference on Computer Vision* (1993), pp. 658–665.

[TT05] Tong W.-S., Tang C.-K.: Robust estimation of adaptive tensors of curvature by tensor voting. *IEEE Transaction on Pattern Analysis and Machine Intelligence 27*, 3 (2005), 434–449.

[VB04] Vančo M., Brunnett G.: Direct segmentation of algebraic models for reverse engineering. *Computing 72*, 1-2 (2004), 207–220.

[VB07] Vančo M., Brunnett G.: Geometric preprocessing of noisy point sets: an experimental study. *Computing 79, Special issue on Geometric Modeling* (2007).

[VBS99] Vančo M., Brunnett G., Schreiber T.: A hashing strategy for efficient k-nearest neighbors computation. In *International Conference on Computer Graphics* (1999), IEEE Computer Society, pp. 120–127.

[VFT07] Várady T., Facello M. A., Terék Z.: Automatic extraction of surface structures in digital shape reconstruction. *Computer-Aided Design 39*, 5 (2007), 379–388.

[VM02] Várady T., Martin R. R.: *Reverse Engineering.* Springer, 2002, ch. 26, pp. 651–681.

[VMC97] Várady T., Martin R. R., Cox J.: Reverse engineering of geometric modelsan introduction. *Computer Aided Design 29*, 4 (1997), 255–268.

[VS05] Vieira M., Shimada K.: Surface mesh segmentation and smooth surface extraction through region growing. *Computer Aided Geometric Design 22*, 8 (2005), 771–792.

[WAFR99] Werghi N., Ashbrook A., Fisher R. B., Robertson C.: Faithful recovering of quadric surfaces from 3d range data. In *2nd International Conference on 3D Digital Imaging and Modeling* (1999), IEEE Computer Society, pp. 280–289.

[WFRA98] Werghi N., Fisher R., Robertson C., Ashbrook A.: Modelling objects having quadric surfaces incorporating geometric constraints. In *5th European Conference on Computer Vision-Volume II* (1998), Springer-Verlag, pp. 185–201.

[WK05] Wu J., Kobbelt L.: Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum 24*, 3 (2005), 277–284.

[YBS05] Yoshizawa S., Belyaev A., Seidel H.-P.: Fast and robust detection of crest lines on meshes. In *ACM Symposium on Solid and physical modeling* (2005), ACM Press, pp. 227–232.

[YLL*05] Yamauchi H., Lee S., Lee Y., Ohtake Y., Belyaev A., Seidel H.-P.: Feature sensitive mesh segmentation with mean shift. In *International Conference on Shape Modeling and Applications 2005* (2005), IEEE Computer Society, pp. 238–245.

[YLW06] Yan D.-M., Liu Y., Wang W.: Quadric surface extraction by variational shape approximation. In *Geometric Modeling and Processing* (2006), pp. 73–86.