Computer-Aided Design and Applications
© 2008 CAD Solutions, LLC
http://www.cadanda.com

Meshless Extraction of Closed Feature Lines Using Histogram Thresholding

Kris Demarsin[1], Denis Vanderstraeten[2] and Dirk Roose[3]

[1]Katholieke Universiteit Leuven, kris.demarsin@cs.kuleuven.be
[2]Metris, Denis.Vanderstraeten@metris.com
[3]Katholieke Universiteit Leuven, dirk.roose@cs.kuleuven.be

ABSTRACT

In reverse engineering, the reconstruction of a surface model from a point cloud requires the extraction of closed feature lines at the borders of the different surface patches. In this paper we propose a new algorithm to extract such closed polygonal feature lines, representing sharp or smooth edges, from a point cloud. Based on the variation of the normal vectors and a graph approach we extract the sharp edges, which are used to divide the point cloud in *smooth regions*. Smooth edges, such as fillets, are extracted for each smooth region separately using a novel approach for 1D-histogram thresholding: we use the curvature histogram in a multi-resolution manner in order to split a point set in different regions of similar curvature (*patches*). The polygonal smooth edges at the borders between these different patches are extracted by point clustering and processing a graph of the point clusters.

Keywords: reverse engineering, point clouds, histogram thresholding, feature lines.
DOI: 10.3722/cadaps.2008.589-600

1. INTRODUCTION

Feature lines on a surface model are defined via local extrema of principal curvatures along corresponding principal directions. In this paper we classify feature lines into two types (see Fig. 1.): smooth and sharp edges. On a surface model we define a *sharp edge* as a curve dividing two surfaces which are tangent discontinuous. A *smooth edge* on the other hand, separates two surfaces which are tangent continuous but curvature discontinuous. Feature lines are used in many applications, e.g. visualization, improvement of the mesh quality, reverse engineering,... Our algorithm operates on point clouds and we detect feature lines as polygonal lines which represent sharp or smooth edges.

1.1 Related Work

Methods related to the detection of feature lines, can be classified into two groups. The first category consists of algorithms which extract unconnected feature lines from a mesh ([11],[14],[23]) or from a point cloud ([10],[17]). Although such unconnected lines are very useful for e.g. visualization, they make it difficult to construct a curve network defining the surface patches. The second category is known as segmentation, of which recently an overview paper appeared ([1]), although only mesh segmentation is considered in that paper. Segmentation methods are typically divided into two groups: face-based approaches and edge-based approaches. Face-based algorithms ([9],[12],[21]) group points with similar geometric properties, using for example a region growing method or a classification algorithm (e.g. K-Means). When the user is interested in the segment borders, these are typically determined by intersecting adjacent fitted surface patches through these point clusters; however, as noted in [2], surface-surface intersection can become unstable for smooth edges. Edge-based algorithms ([6],[13],[24]) detect border points (or edges in case of a mesh) and connect them to obtain border curves. Most of these algorithms require a threshold for this edge detection, e.g. an angle threshold between two adjacent facets or a curvature threshold. The corresponding segments are usually found by a region growing method.

Some algorithms cannot be classified in any of the two segmentation groups. For example Benko et al. [2] discard triangles in the vicinity of sharp edges or small blends causing disjoint regions which might be further subdivided by smooth edges. To find the sharp edges, surface-surface intersection is used. A more recent example of such a combined approach is described by Várady et al. [22]. This algorithm uses morse theory to separate the relatively flat regions by highly curved transitions, called the separator sets, on the entire mesh. Then, contour lines are extracted and optionally extended. The algorithm results in a well-aligned structure required for surface fitting.

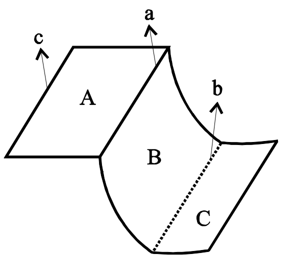
1.2 Our Approach

This paper introduces a new algorithm to extract polygonal feature lines, representing sharp or smooth edges, from a point cloud for the purpose of surface reconstruction. Since the reconstruction of a surface model requires a network of *closed* feature lines indicating the borders of the various surface patches, the extraction of *closed* lines is emphasized.

The presented algorithm is a combination of an edge-based and a face-based approach: a clustering method gives an initial segmentation of the point cloud and extracts candidate feature points. Building and processing a graph of these clusters results in the closed feature lines, which can be used to perform the segmentation.

This paper proposes a multi-step approach, combining the extraction of sharp and smooth edges. First, closed sharp edges are extracted using the variation of the normal vectors and graph processing, as described in [4].

Using these sharp edges (and the external boundary if the point cloud does not represent a solid, as in Fig. 1.), we divide the cloud in subsets enclosed by sharp edges (and optionally an external boundary). These subsets, which contain no sharp edges anymore, are called *smooth regions* (see Fig. 1.) and each smooth region is treated separately to extract the smooth edges. In this paper, we only consider point clouds which can be divided in *patches*, i.e. a patch is a set of points with similar curvature. In a smooth region, a smooth edge is located at the border between two patches. Therefore, the smooth regions are split in sub-regions of similar curvature using one or more curvature thresholds. A new multi-level thresholding approach, based on a curvature histogram, is proposed to determine these thresholds. Each threshold is determined by a multi-resolution analysis of the curvature histogram.



A + B + C = entire point cloud
A, (B + C) = smooth regions
A, B, C = patches
a = sharp edge
b = smooth edge
c = external boundary

Fig. 1: Illustration of a point cloud with the different features used in this paper.

To determine the smooth edges at the borders of the different patches, a region growing method is applied, which results in point clusters indicating the smooth edges and allows building a graph connecting neighboring clusters. After processing this graph, using similar operations as for the sharp edges ([4]), we obtain the smooth edges which separate the different patches.

The advantages of the algorithm are:

- The multi-step approach allows the use of optimal parameter values per smooth region. For point clouds with different types of smooth edges in the separate smooth regions, this approach can be more accurate compared to single-step approaches (e.g. [22]), which extract all edges at once using one set of parameter values.
- While single-step approaches might smooth also the sharp edges, our multi-step approach preserves these edges since we perform smoothing after the sharp edge extraction.
- The required curvature thresholds are automatically estimated.
- Mesh generation is unnecessary.
- The extracted lines are closed, making the algorithm suitable as pre-process for surface reconstruction.
- The graph approach allows to process point clouds very efficiently.

The paper is structured as follows. Section 2 constitutes the main contribution of this paper, i.e. the histogram approach to separate the different patches in a smooth region. The other steps of the algorithm to construct polygonal feature lines representing smooth edges are explained in section 3. Results of the feature line extraction algorithm for real-world point clouds with sharp and smooth edges are illustrated in section 4. The conclusions are formulated in section 5.

2. HISTOGRAM THRESHOLDING

2.1 Introduction

We first define the following notions:

- $curv$: set of curvature values with $curv(p)$ equal to the mean curvature in point *p*. To compute the mean curvature in *p*, we fit a quadric through the *k* nearest neighbors of *p*, as described in e.g. [18].
- τ : curvature threshold, i.e. a curvature value used to divide a smooth region in two different areas *(low* and *high* points, see the next definition). All the curvature thresholds τ_i together divide a smooth region in patches.
- *Low* points: set of points with a curvature below (or equal to) the given threshold τ .
- *High* points: set of points with a curvature above the given threshold τ .
- *L*: resolution of a histogram, i.e. the number of equally sized bins, among which we distribute the given values.
- *h*: 1D curvature histogram $h = (h_i)_{i=1...L}$ with bin size $|bin| = \frac{\max(curv) - \min(curv)}{L}$ and h_i the frequency of curvature values in bin *i,* i.e. if $\min(curv) + (i - 1) * |bin| \leq curv(p) < \min(curv) + i * |bin|$ then $curv(p)$ belongs to bin *i*. The last bin also counts the values that are equal to $\max(curv)$.
- L_{start}: the start resolution, i.e. the histogram resolution from which the thresholding algorithm starts.
- Start histogram: the curvature histogram *h* with $L = L_{start}$.

Although other curvature measures exist, i.e. the principal curvatures and the Gauss curvature, we choose the mean curvature for the following reason. Using one principal curvature does not allow distinguishing between all different patches. For example, for a rounded cube (see Fig. 7.) the fillets and the planar regions have the same minimum curvature. Similar, the maximum curvature cannot distinguish between the fillets and the corners. The Gauss curvature is also unable to separate the fillets from the planar areas, since this curvature is defined as the product of the principal curvatures. Because the mean curvature is defined as the mean of the principal curvatures, this curvature is better suited to distinguish between the different patches.

Using histogram thresholding, a smooth region is segmented in homogeneous point regions with respect to some property. For this property, the mean curvature is chosen, since smooth edges are located at the borders between regions of significantly different mean curvature (*patches*). This definition for 3D point cloud segmentation originates from image segmentation ([8],[16],[20]): an image is segmented into homogeneous pixel regions, i.e. the pixels of one region are similar with respect to e.g. color, intensity, or texture. In image segmentation, histogram thresholding is an extensively used technique. Bi-level thresholding (*binarization*) assumes that the gray levels of pixels belonging to the objects are substantially different from the gray levels of the pixels belonging to the background, such that one threshold separates the objects from the background, e.g. the extraction of text characters from a document image. However, more complex images, e.g. color images, require *multi-level* thresholding to separate all the objects in the image. Thresholding algorithms based on a histogram of pixel intensity values, assume that the histogram peaks (*modes*) represent the different objects that need to be separated. Determination of the segmentation thresholds corresponds to searching the valleys of the histogram. Many thresholding methods view the histogram as a mixture of density functions ([3],[15]), or use functional histogram approximation ([19]). Parametric approaches ([5]) make an assumption on the underlying probability density function, e.g. the Gaussian Mixture Model, and the related parameters are estimated to segment the histogram.
Similar to image segmentation, we determine the valleys of the 1D mean curvature histogram under the assumption that the histogram peaks represent the different patches in the point cloud. Another reasonable assumption is made, namely the width of a histogram valley is at least as large as the width of the preceding peak. As more samples create a clearer image histogram, more points in the point cloud create a clearer curvature histogram. However,

there are also some important differences between a gray-level histogram and a curvature histogram. The first one is the determination of the number of histogram bins. For a gray-scale image the maximum intensity is known in advance (e.g. 256). The number of different possible curvature values in a point cloud is not known in advance and thus the choice for a good resolution of a curvature histogram is far more difficult. Fig. 2. illustrates the histograms of the mean curvature at three different resolutions for a noisy rounded cube (see Fig. 7.). In Fig. 2(b). we see three peaks representing the three different patches (respectively planar regions, fillets and corners). We want to determine the curvature threshold τ_2 , located after the peak of the planar areas, and τ_1 , located after the peak of the fillets, to separate the different patches of the rounded cube. While this resolution clearly shows the valleys we are interested in, a too coarse resolution hides the peaks, see Fig. 2(c). A too fine resolution for a noisy point cloud might cause a histogram without any significant peaks since there is too much variation on the curvature. For the cube, when we take a resolution which is as large as the point cloud size, see Fig. 2(a)., we still see the two largest peaks, but many local minima exist (caused by *noisy peaks*), making it difficult to determine the correct valleys. Related to the unknown boundaries of the curvature values, a second difference with image histograms arises: noisy point clouds typically have points with a curvature which is significantly larger than the curvature of the other points, causing the histogram to deform. Later on, we will discuss this into more detail.

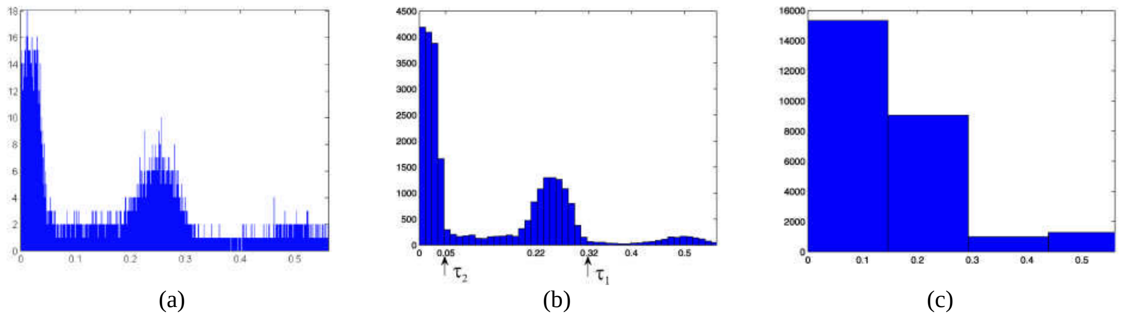


Fig. 2: Mean curvature histogram for a noisy rounded cube at three different resolutions: (a) L = |point cloud| ≈ 30000; (b) L = 50; (c) L = 4.

2.2 Automatic Threshold Detection

We present a novel approach for multi-level threshold selection for a smooth region, using a non-parametric 1D-histogram approach which does not assume an underlying density function and does not rely on functional histogram approximation. The algorithm is robust to noisy peaks and avoids the resolution problem of the curvature histogram: in a fast and memory efficient manner, multiple resolutions of the mean curvature histogram are considered and the best resolution to extract a curvature threshold is automatically detected. Recursively applying this approach results in multiple thresholds.

Since we assume that a smooth region is the union of different patches, the number of points in one patch is much higher than the number of points located between two neighboring patches. Consequently, in the curvature histogram with a good resolution (see Fig. 2(b).) the peaks are alternated by valleys with the thresholds in between. The idea behind the multi-resolution approach is to coarsen a 'fine enough' histogram to obtain a histogram at a resolution for which a peak and the following valley are each represented by one bin such that the threshold after the peak is characterized by a large ratio of the frequencies of the two coarsened bins.

In Fig. 2 we illustrated that a too coarse resolution makes it impossible to detect the valleys and thus we choose a start resolution L_{start} which is 'fine enough'. The only requirement for L_{start} is that the start histogram shows the global peaks and valleys, but noisy peaks are allowed. For the tested models in this paper L_{start} = 50 is sufficient.

Suppose we know the threshold τ we want to determine in the start histogram, e.g. τ_1 or τ_2 for the rounded cube (see Fig. 2(b).). To make the algorithm robust to noisy peaks, the start histogram is coarsened by joining bins. We join k bins starting from τ to the left of the histogram until the peak is represented by one bin, see Fig. 3(a). Assume that the curvature threshold τ corresponds to the right end of bin $p(\tau)$ of the start histogram, i.e.

$p(\tau)=\dfrac{\tau-\min(curv)}{|bin|}$ and let $start\_curv=\tau-k*|bin|$, i.e. $k=p(\tau)-p(start\_curv)$. Then the frequency of curvature values in this coarsened bin is $b_1=\sum_{i=p(start\_curv)+1}^{p(\tau)} h_i$. If we assume that the width of a histogram valley is at least as large as the width of the preceding peak, then by joining *k* bins starting from τ to the right, the valley (or at least a part of it) is represented by one bin with a frequency of curvature values $b_2=\sum_{i=p(\tau)+1}^{p(\tau)+k} h_i$.

Consequently, at the threshold τ we have a large ratio $\dfrac{b_1}{b_2}$.

In our threshold selection procedure, we consider all possible curvature values for $start\_curv$ and τ:

$$start\_curv=\min(curv)+i*|bin| \text{ for } i=0,\dots,L_{start}-2$$
$$\tau=start\_curv+k*|bin| \text{ for k} =1,\dots,\lfloor(L_{start}-p(start\_curv))/2\rfloor.$$

If we define $\varepsilon(\tau,start\_curv)=\dfrac{b_1}{b_2}$, then we are able to formulate the algorithm to determine *one* curvature threshold τ of the start histogram: we search the values $start\_curv$ and τ for which $\varepsilon(\tau,start\_curv)$ attains its maximum.

For the rounded cube, Fig. 3(b) illustrates that $\max_{\tau}(\varepsilon(\tau,start\_curv))$ attains its maximum for $start\_curv=0.22$ and Fig. 3(c) shows that this maximum is attained for $\tau=\tau_1$. For $start\_curv=0$ we have $\tau=\tau_2$. Since the algorithm determines the maximum of $\varepsilon(\tau,start\_curv)$ the threshold $\tau=\tau_1\approx 0.32$, located in the second valley of the start histogram.

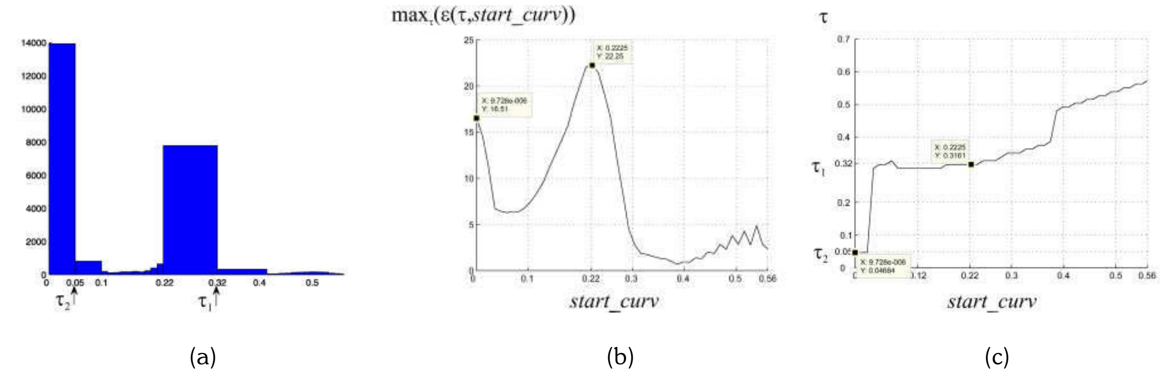


Fig. 3: Illustration for the rounded cube: (a) the histogram with bins joined at τ_1 and τ_2 to represent each peak/valley by one bin; (b) the maximum ratio for each value of $start\_curv$; (c) the values for τ for each value of $start\_curv$ such that $\varepsilon(\tau,start\_curv)$ attains its maximum.

The algorithm to determine the maximum of $\varepsilon(\tau,start\_curv)$ can be implemented very efficiently. We consider all possible values for $start\_curv$ and for a fixed value of $start\_curv$ we join *k* $(k=1,\dots,\lfloor(L_{start}-p(start\_curv))/2\rfloor)$ bins. Since we compute new values for b_1 and b_2 very efficiently from their previous values (b_1_prev and b_2_prev) and since we use L_{start} = 50 in this paper, the algorithm runs fast.

Moreover, experiments indicate that only *one* run of the algorithm is sufficient, i.e. for $start\_curv = \min(curv)$, we obtain similar results than when we consider *all* possible values for $start\_curv$. This leads to a very simple and short implementation:

```
$h = (h_i)_{i=1...L_{start}};$
$k = 0; \max\_ratio = -1;$
$b_1\_prev = 0; b_2\_prev = 0;$
$while(k < \lfloor L_{start} / 2 \rfloor)$
        $k = k + 1;$
        $b_1 = b_1\_prev + h_k;$
        $b_2 = b_2\_prev - h_k + h_{2*k-1} + h_{2*k};$
        $if(b_2 != 0)$
                $ratio = b_1 / b_2;$
        $else\ ratio = b_1;$
        $end\ if$
        $if(ratio > \max\_ratio)$
                $\max\_ratio = ratio; \max\_k = k;$
        $end\ if$
        $b_1\_prev = b_1; b_2\_prev = b_2;$
$end\ while$
$\tau_1 = \min(curv) + \max\_k * |bin|;$
```

Fig. 4(a) illustrates the influence of L_{start} = 2,…,50 on the threshold found for the rounded cube: from L_{start} =13 the fluctuations become less and the threshold τ converges to 0.32. For L_{start} = 50,…,|point cloud|, see Fig. 4(b)., we obtain $\tau = 0.32$ with a precision of 0.6%. This states the assumption we made earlier: L_{start} must be 'fine enough' and typically, L_{start} = 50 is sufficient.

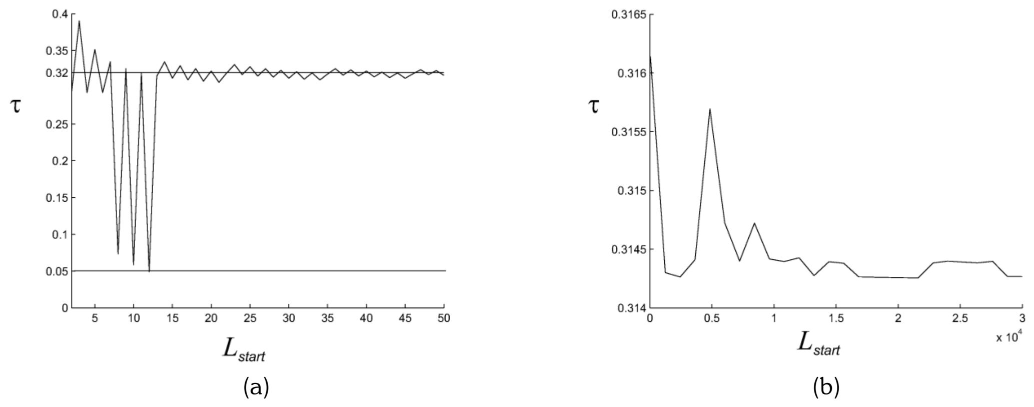


Fig. 4: Illustration of the influence of the start resolution L_{start} on the threshold found for the rounded cube.

2.3 Additional Thresholds

We demand input from the user at two different stages: (a) after the high-low division is made using a curvature threshold and (b) after the feature lines are extracted. The first user-input requires an indication whether the computed threshold is correct. As we will see in the results section, even though the determined threshold is located in a valley, it is not necessarily the desired valley. For example, the point cloud in Fig. 9(a) contains two fillets, but we see that the initial high-low division is not correct and the user indicates that another threshold needs to be determined. Our approach enables to shift the threshold to a previous or to a next valley in a fast and easy manner: the same algorithm is used, but only the curvature values below or above the originally determined threshold are considered. For this purpose, the user has to indicate whether a higher or lower curvature threshold is required, which is not a difficult task assuming that the user has some notion of the different curvature areas. This user-input can reduce the cost of the algorithm drastically: the graph of all neighboring clusters (see section 3) is only built and processed once with the correct threshold. For the rounded cube, the initial threshold τ_1 is accepted since it separates the corner points from the other points, which results in correct lines at the corners.

The second input demands an indication whether the extraction of the feature lines is complete. If not, additional thresholds are determined until the user is satisfied. This avoids the automatic determination of the number of desired thresholds, which is, as in image segmentation, a difficult task. For the rounded cube, a second threshold, lower than the initial threshold, is required to separate the fillets from the planar areas. For this purpose, we build a histogram with only the curvature values below the first threshold τ_1, see Fig. 5. The maximum of $\varepsilon(\tau, start\_curv)$ is attained for $start\_curv = 0$ and $\tau = \tau_2 \approx 0.05$, with τ_2 located in the first valley of the start histogram (Fig. 2(b).). The two thresholds τ_1 and τ_2 result in the separation of the planar regions, the fillets and the corners of the rounded cube. In a similar way, the user can indicate to consider the histogram of the curvature values above the preceding threshold.

Most of the point clouds we considered have a curvature histogram which consists of only one large peak representing the planar areas. Typically, the different patches are surrounded by points located in this large peak and using one threshold immediately after the first peak is sufficient.

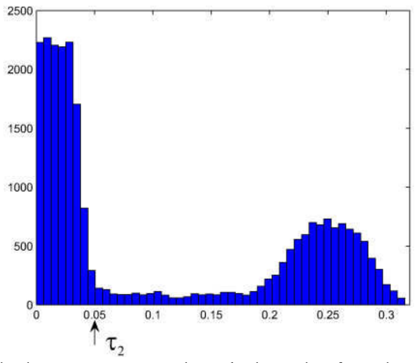
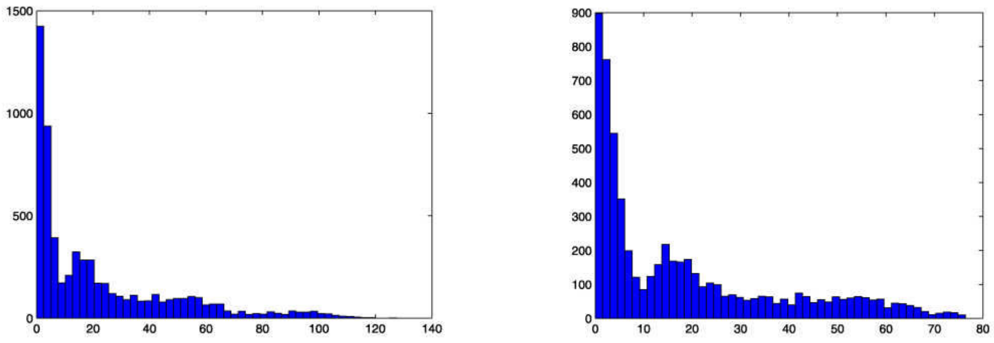


Fig. 5: Histogram with the curvature values below the first threshold (rounded cube).

2.4 Noise

Since the point clouds for which we build the histogram have no sharp edges, Laplacian smoothing is typically performed for noisy point sets before the histogram is built.

Noisy point clouds typically have points with a curvature which is significantly larger than the curvature of the other points. Such outliers have a negative influence on the histogram: the peaks are shifted to the left causing less clear valleys in between, see Fig. 6(a). Removal of a specified percentage of the highest curvature values from the histogram, reduces this effect, see Fig. 6(b).



(a) (b)

Fig. 6: Histogram *before* (a) and *after* (b) removal of outliers.

3. CONSTRUCTION OF POLYGONAL SMOOTH EDGES

The algorithm described in section 2 results in an initial labeling of low and high points. The high points are further divided into points situated inside a higher curvature region and points at the border of such a region:

- *Inner* points: the high points with only high points in the 1-ring neighborhood ([7]).
- *Border* points: the remaining high points with at least two inner points in the 1-ring neighborhood. We request that *two* neighbors are inner points, because in the ring neighborhood of a point *p*, the neighbors are distributed around *p*, i.e. a point on the border of a higher curvature region will typically have some low points and some inner points as neighbors. Since the size of a ring neighborhood is typically 5 or 6, two inner points as neighbors of a border point is an appropriate choice.

This labeling causes the inner points to be separated from the low points by border points. High points that are not inner or border points are considered to be noisy isolated points and are added to the set of low points. Fig. 7(a). illustrates the different labels for the rounded cube at the first threshold: the corner areas consist of inner points and are bounded by border points.

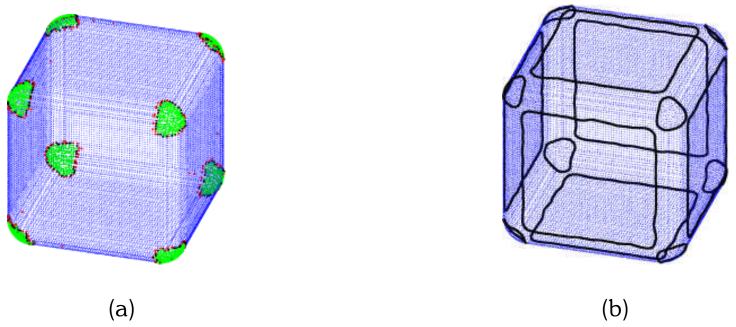


Fig. 7: Rounded cube with (a) labeling using the first threshold: low points in blue, inner points in green, border points in black and the noisy points in red; (b) extracted feature lines using two thresholds.

Afterwards, we divide the smooth region in point clusters with a region growing approach, based on these labels. The purpose of this clustering is similar as described in [4]: we want to create large clusters surrounded by small clusters. These small clusters indicate the smooth edges and thus a graph is built connecting neighboring clusters to extract the smooth edges.

We randomly choose an inner or a low point as a seed to start the growing process. As long as we encounter a point with the same label, we add this point to the current cluster. When we meet a point with a different label than the cluster we are growing, a small cluster consisting of only the encountered point is created. When the growing stops, a new seed is chosen (inner or low point) to restart the growing and this is repeated until all points belong to a certain cluster. In this way, the different patches are split up in large clusters of neighboring points surrounded by small clusters, e.g. for the cube, the patch of corner points is divided into 8 clusters corresponding to the 8 corners.

To construct closed polygonal lines at the smooth edges, we build and process a graph of all neighboring clusters, as described in [4]. Fig. 7(b) illustrates the end result for the rounded cube using two thresholds. The lines at the corners are not connected to the lines between the fillets and the planar areas since we used two thresholds, each resulting in closed feature lines. In addition, the smoothing step increases the distance between the two resulting feature line graphs.

4. RESULTS

We tested the algorithm on several point clouds obtained by scanning actual objects. A first point cloud with three extracted fillets is illustrated in Fig. 8.: one fillet has constant radius and the two other ones have a decreasing radius that vanishes at the end. This point cloud has no sharp edges and thus we immediately start with the threshold determination. After smoothing, the first curvature threshold was sufficient to obtain the perfectly closed feature lines. The boundary of the point cloud is extracted and included in the graph.

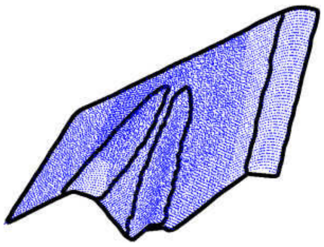


Fig. 8: Noisy point cloud (9202 points) with three extracted fillets.

Fig. 9(c) illustrates the extracted fillet lines for a point cloud of 6440 points. The histogram of this point cloud is shown in Fig. 6(b). Here the difficulties are: the occurrence of noise (smoothing was necessary), two fillets close to each other, low sample density and a fillet neighborhood which is not entirely flat.
The first selected threshold is located in the first valley and results in the high-low division as shown in Fig. 9(a).; however, the desired threshold is located after the second peak. After user-input, we obtain a correct high-low division, see Fig. 9(b)., which results in the extraction shown in Fig. 9(c).

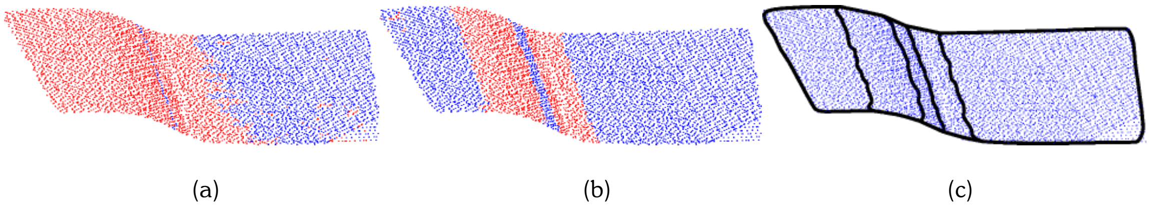


Fig. 9: Noisy point cloud with (a) rejected and (b) accepted high-low division, and (c) the two extracted fillets.

A point cloud, representing a part of the cover of a mobile phone, see Fig. 11(a)., was used in [4] to illustrate the sharp edge extraction. Now, we extract the fillets in the two large smooth regions with the algorithm presented in this paper, see Fig. 11(b-c). This cloud is noisy and has no good point sampling, as illustrated in Fig. 10.

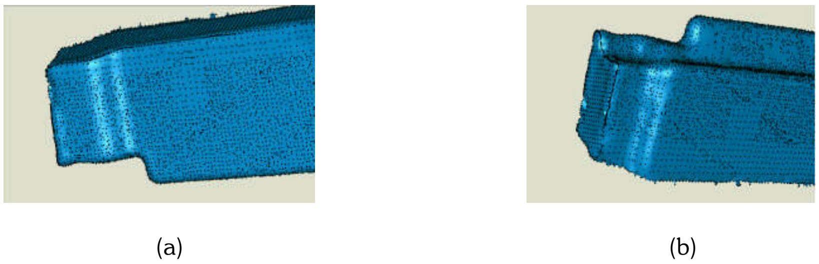


Fig. 10: Two zooms of the mobile phone point cloud (11 034 points) illustrating bad point sampling.

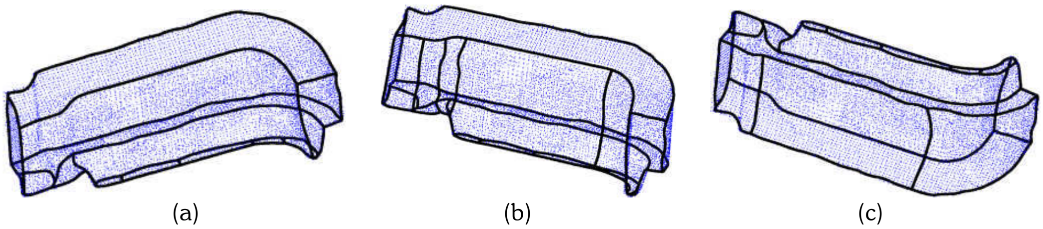


Fig. 11: Mobile phone point cloud with the sharp edges (a) and with the extracted fillets visualized separately for the two large smooth regions enclosed by sharp edges (b-c).

The point cloud used in Fig. 12 illustrates the advantage of our approach to treat each smooth region separately to detect the fillets. The sharp edges are detected without smoothing, resulting in a correct extraction, as illustrated in [4]. The smooth region of the two fillets is noisy and requires smoothing for a good extraction. Since this smooth region contains no sharp edges anymore, smoothing can be performed more safely than when we should smooth the entire point cloud.

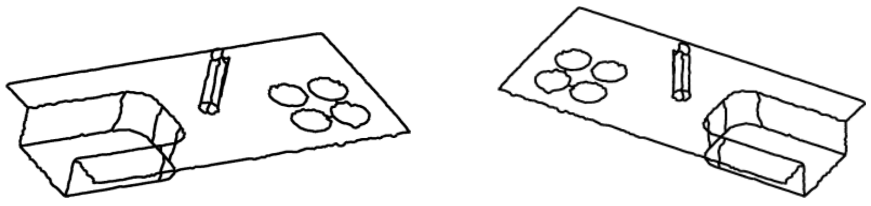
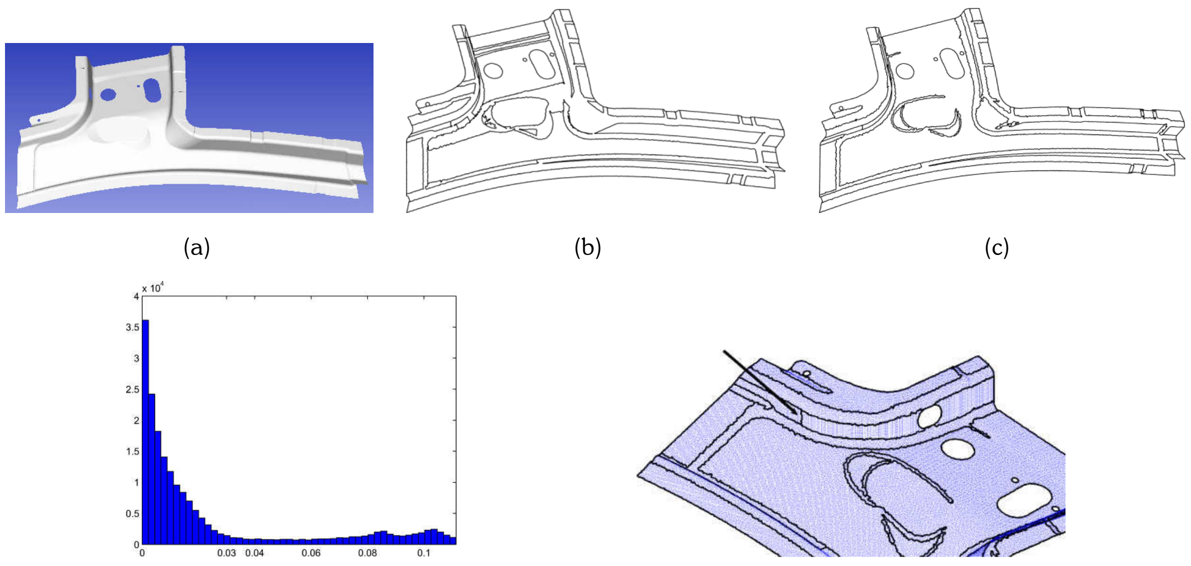


Fig. 12: Two different views of the fillets of the brick cloud (41 432 points). The boundaries are not considered.

The last model we will discuss is a large point cloud of 201 387 points, representing a sheet metal part (see Fig. 13(a)). The smooth edge extraction of this noise-free point cloud is shown in Fig. 13(b)., illustrating the accuracy of our method for noise-free models. After adding Gaussian noise (σ = 10µm) in the direction of the normal vectors, the result in Fig. 13(c). is obtained. We note that, due to the noise, some lines are now missing.
When point clouds become larger and the complexity increases, typically more patches are present and the histogram becomes less clear due to partial overlaps of histogram peaks. A multi-step approach is then appropriate: after the first smooth edge extraction using an initially determined threshold (typically after the first and only large peak), we apply the histogram approach to the various large clusters, which result from the growing approach based on labels (see section 3). Further subdivision can be applied according to the user's needs. For example, after an initial extraction from the noisy sheet metal cloud, an additional fillet line (indicated by the arrow in Fig. 13(e).) has been determined by applying the algorithm to a part of the point set. This extra fillet line is based on τ = 0.002, which is not a visible valley in the histogram of the entire point cloud, see Fig. 13(d).



(d) (e)

Fig. 13: (a) Sheet metal model; (b) smooth edges of the noise-free model; (c) smooth edges of noisy model; (d) histogram of entire noisy cloud; (e) additional fillet line extracted by considering a smaller part of the noisy point cloud, i.e. one large cluster resulting from the grow step (based on labels).

5. CONCLUSION

We presented an algorithm to extract closed polygonal feature lines from point clouds. Sharp edges are extracted using the algorithm described in [4]. For the extraction of the smooth edges, we introduced a multi-level threshold selection method based on a curvature histogram to determine the curvature thresholds which separate the different patches. The threshold selection is based on a multi-resolution approach to find a suitable resolution for a curvature histogram. This multi-resolution approach makes the threshold selection method robust to noisy peaks in the histogram and is very fast. Additional thresholds are also computed very fast using the same method acting on a part of the histogram.

The feature line extraction algorithm is fast since we avoid the generation of a mesh, we use an efficient method to set the curvature thresholds, we cluster the points before building a graph and we use efficient graph processing algorithms. We obtained good results for real-world (noisy) data sets from industrial applications.

In the future we will consider a 2D-histogram of the mean and Gauss curvature, and the construction of a curve network, which defines the relation between the different segments and boundary loops.

6. ACKNOWLEDGEMENTS

The used point clouds in this paper are courtesy of Metris.

We would like to thank Hendrik Speleers (K.U. Leuven) for the many useful discussions on the histogram approach.

7. REFERENCES

[1] Agathos, A.; Pratikakis, I.; Perantonis, S.; Sapidis, N.; Azariadis, P.: 3D mesh segmentation methodologies for CAD applications, Computer-Aided Design and Applications, 4(6), 2007, 827-841.

[2] Benko, P.; Martin, R.; Varady, T.: Algorithms for reverse engineering boundary representation models, Computer-Aided Design, 33, 2001, 839-851.

[3] Chang, C.-I.; Du, Y.; Wang, J.; Guo, S.-M.; Thouin, P. D.: Survey and comparative analysis of entropy and relative entropy thresholding techniques, Vision, Image and Signal Processing, 153(6), 2006, 837-850.

[4] Demarsin, K.; Vanderstraeten, D.; Volodine, T.; Roose, D.: Detection of closed sharp edges in point clouds using normal estimation and graph theory, Computer-Aided Design, 39(4), 2007, 276-283.

[5] Duda, R.; Hart, P.; Stork, D.: Pattern classification, John Wiley and Sons, 2000.

[6] Fan, T.-J.; Medioni, G.; Nevatia, R.: Segmented descriptions of 3-D surfaces, IEEE Journal of robotics and automation, RA-3(6), 1987.

[7] Floater, M. S.; Reimers, M.: Meshless parameterization and surface reconstruction, Computer Aided Geometric Design, 18(2), 2001, 77-92.

[8] Fu, K. S.; Mui, J. K.: A survey on image segmentation, Pattern Recognition, 13, 1981, 3-16.

[9] Gelfand, N.; Guibas, L. J.: Shape Segmentation Using Local Slippage Analysis, Eurographics Symposium on Geometry Processing, 2004, 219-228.

[10] Gumhold, S.; Wang, X.; MacLeod, R.: Feature Extraction from Point Clouds, Proceedings of the 10th International Meshing Roundtable, 2001, 293-305.

[11] Hildebrandt, K.; Polthier, K.; Wardetzky, M.: Smooth feature lines on surface meshes, Symposium on Geometry Processing, 2005, 85-90.

[12] Lavoué, G.; Dupont, F.; Baskurt, A.: A new CAD mesh segmentation method based on curvature tensor analysis, Computer-Aided Design, 37, 2005, 975-987.

[13] Meyer, A.; Marin, P.: Segmentation of 3D triangulated data points using edges constructed with a C1 discontinuous surface fitting, Computer-Aided Design, 36, 2004, 1327-1336.

[14] Ohtake, Y.; Belyaev, A.: Automatic Detection of Geodesic Ridges and Ravines on Polygonal Surfaces, The Journal of Three Dimensional Images, 15(1), 2001, 127-132.

[15] Otsu, N.: A threshold selection method from grey-level histograms, SMC, 9(1), 1979, 62-66.

[16] Pal, N. R.; Pal, S. K.: A Review on image segmentation techniques, PR, 26(9), 1993, 1277-1294.

[17] Pauly, M.; Keiser, R.; Gross, M. H.: Multi-scale Feature Extraction on Point-sampled Surfaces, Computer Graphics Forum, 22(3), 2003, 281-290.

[18] Petitjean, S.: A survey of methods for recovering quadrics in triangle meshes, ACM Computing Surveys, 34(2), 2002, 211-262.

[19] Ramesh, N.; Yoo, J.-H.; Sethi, I. K.: Thresholding based on histogram approximation. IEE Proceedings - Vision, Image and Signal Processing, 142(5), 1995, 271-279.

[20] Sezgin, M.; Sankur, B.: Survey over image thresholding techniques and quantitative performance evaluation, Journal of Electronic Imaging, 13(1), 2004, 146-165.

[21] Vanco, M.; Brunnett, G.: Direct Segmentation of Algebraic Models for Reverse Engineering, Computing, 72(1-2), 2004, 207-220.

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

[23] Watanabe, K.; Belyaev, A. G.: Detection of Salient Curvature Features on Polygonal Surfaces, Computer Graphics Forum, 20(3), 2001, 385-392.

[24] Yang, M.; Lee, E.: Segmentation of measured point data using a parametric quadric surface approximation, Computer-Aided Design, 31, 1999, 449-457.