Detecting Holes in Point Set Surfaces

Gerhard H. Bendels   Ruwen Schnabel   Reinhard Klein

Universität Bonn
Institut für Informatik II – Computergraphik
Römerstraße 164
D-53117 Bonn, Germany

ABSTRACT

Models of non-trivial objects resulting from a 3d data acquisition process (e.g. Laser Range Scanning) often contain holes due to occlusion, reflectance or transparency. As point set surfaces are unstructured surface representations with no adjacency or connectivity information, defining and detecting holes is a non-trivial task. In this paper we investigate properties of point sets to derive criteria for automatic hole detection. For each point, we combine several criteria into an integrated boundary probability. A final boundary loop extraction step uses this probability and exploits additional coherence properties of the boundary to derive a robust and automatic hole detection algorithm.

Keywords Point Set Surfaces, Modelling, Filtering, Repairing

1 Introduction

Point set surfaces have become popular with the rise of 3D data acquisition techniques such as laser-range scanning. Their conceptual simplicity makes them suitable for both modelling as well as high quality rendering. Usually, these 3D data acquisition methods deliver unstructured point clouds, possibly equipped with normals and additional surface properties, such as colour. The surface is encoded implicitly therein and can only be extracted using some neighbourhood relation between samples. Compared to mesh based representations, the lack of explicit connectivity information simplifies the definition and implementation of many tasks encountered in geometric modelling, such that for instance free-form deformation techniques for point sets become increasingly popular [PKKG03, BK05]. On the other hand, the detection of holes in the surface – trivial in the case of meshes – becomes an ill-defined problem.
The knowledge of holes in the data, however, is vital for many applications dealing with point set surfaces and it can be exploited in several ways. It can be used to reconstruct surfaces with boundaries or to direct a further scanning step, gathering missing information in holes, either manually or even automatically. In postprocessing, a smoothing step to remove noise profits from boundary information as many smoothing operators usually fail on boundaries and special handling is required at the borders. Identification of points on the boundary of a hole is obviously required before any attempt to algorithmically fill holes, an application useful not only in surface repairing but also in modelling and interactive editing [BSK05, SACO04].
While several authors proposed sampling conditions for surfaces to ensure correct reconstruction (most notably [ABE98]), we are not primarily concerned with undersampling but are interested in holes that a human user might identify when inspecting a point cloud, often unaware of the original surface. Also we want to provide a user with intuitive parameters making it easy to find the holes needed for a given application.

The Journal of WSCG, Vol.14, ISSN 1213-6972
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency–Science Press

2 Previous Work

The problem of detecting holes in point set surfaces is closely related to surface reconstruction as well as feature extraction. Thus, many algorithms in those areas include criteria to identify holes or undersampled surface patches.
[GWM01], [LP02] as well as [CN04] apply what we shall call the angle criterion. The angle criterion considers for each sample point $\mathbf{p}$ a set of neighbouring samples and examines the maximum angle between two consecutive neighbours. [GWM01] also use the

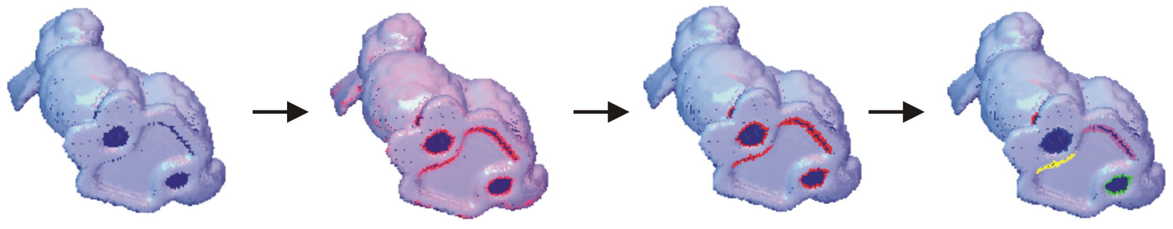

Figure 1: The steps of the boundary detection algorithm. From left to right: A boundary probability $\Pi(\mathbf{p})$ is computed for every point (the points are shaded in red according to their boundary probability). Then points are classified into boundary and interior points, exploiting coherence. Finally, for each hole a boundary loop is extracted.

correlation matrix formed by the neighbourhood. The eigenvectors and eigenvalues of this matrix define a correlation ellipsoid. Its shape, expressed in the ratios of the eigenvalues, is used to identify corner, crease and boundary points and also gives an approximation to crease and boundary direction. In order to find continuous crease lines, a neighbourhood graph on the point set is built and its edges are weighted according to the crease probability. Edges with high probability are then collected and constitute the feature patterns.

In [DG01], undersampled regions are detected using the sampling requirement of [ABE98]. This sampling condition is based on an approximation of the medial axis by so called poles of each sample's Voronoi cell. The distance of each point to the medial axis gives the local feature size. Every point on the true surface needs at least one sample point within a ball defined by the local feature size and a factor r. Consequently, [DG01]'s approach fails to identify holes in flat areas of the surface, where only very few samples are required to fulfill this requirement (in flat areas the medial axis is far away). In these areas, though, often holes are present and clearly visible for a human observer. Similarly, we are not interested in regions declared undersampled at sharp creases where the sampling requirement can never be met (at sharp edges the medial axis touches the surface).

3 Overview

Let $\mathcal{S}$ be a 2-manifold surface and let the set of points $\mathcal{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_n\} \subset \mathbb{R}^3$ be a (not necessarily regular) sampling of $\mathcal{S}$. Suppose also that $\mathbf{n}_1, \dots, \mathbf{n}_n$ are the corresponding surface normals. The problem is now to define an operator

$$B_{\mathcal{P}} : \mathcal{P} \to 2^{\mathcal{P}} \; ; \; B_{\mathcal{P}}(\mathcal{P}) \mapsto \{\mathbf{p} \in \mathcal{P} | \mathbf{p} \text{ is boundary}\}$$

that identifies the set of boundary points $\mathcal{B} = B_{\mathcal{P}}(\mathcal{P})$ circumscribing holes in $\mathcal{P}$. We denote the boundary operator with a subscript $\mathcal{P}$ to stress that the assignment *boundary* or *non-boundary* is strictly a property of the point set under consideration itself.

The basic layout of our hole detection scheme (depicted in figure 1) is as follows: For each point $\mathbf{p} \in \mathcal{P}$ we compute a boundary probability $\Pi(\mathbf{p})$, reflecting the probability that $\mathbf{p}$ is located on or near a hole in the surface sampling (section 4). Thereafter, we exploit that the boundary property is coherent, i.e. that boundary points have proximate neighbours that are also boundary, and construct closed loops circumscribing the hole in a shortest cost path manner (section 5). Results and applications of our hole detection scheme are given in sec. 6.

4 Boundary Probability

The property of *being boundary* inherently is a property of the local neighbourhood of $\mathbf{p}$ rather than of the point $\mathbf{p}$ itself. In order to define and evaluate the boundary criteria, we therefore have to seize the local neighbourhood $N_{\mathbf{p}}$ more formally.

4.1 Neighbourhood Collection

A very common definition of local neighbourhoods around a point $\mathbf{p}$ found in the literature is the k-neighbourhood $N_{\mathbf{p}}^k$, consisting of the k nearest samples in $\mathcal{P}$ to $\mathbf{p}$. This simple definition, though, becomes unreliable in areas of varying sampling density. In points lying on the edge of a densely sampled region, the k-neighbourhood will be biased towards the densely sampled region (fig. 2, left).

This problem can be alleviated somewhat by the $N_{\mathbf{p}}^{k\epsilon}$ neighbourhood, that includes not only the k nearest points but also all points inside a small sphere with radius ϵ. By selecting an appropriate value for ϵ, the biasing effect can be reduced, but the neighbourhood of points in densely sampled regions will contain more points than necessary, increasing the cost of evaluating the boundary criteria, which effectively limits the range of a feasible ϵ. For sharp sampling density

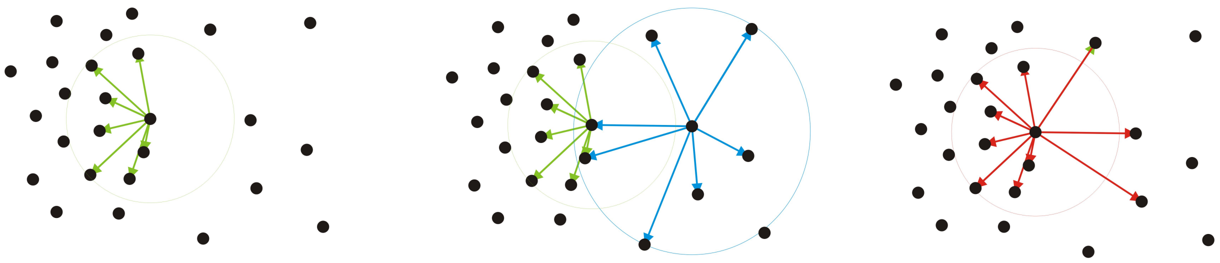

Figure 2: Left: The $k\epsilon$-neighbourhood is biased towards densely sampled regions. Middle: $k\epsilon$-neighbourhoods of points in the sparsely sampled region contain points of the densely sampled area. Right: The symmetric $k\epsilon$-neighbourhood is not affected by the change in sampling density.

changes (as often encountered in point sets stemming from registered range images), this alleviation alone is not sufficient.

However, whereas the $k\epsilon$-neighbourhood for points situated on a sampling density drop will include only points in the densely sampled region, these points will be contained in the neighbourhood of nearby points, located in the sparsely sampled region (fig. 2, middle). To overcome the biasing effect, it therefore typically suffices to include these nearby points in the neighbourhood (fig. 2, right). To complete the neighbourhood for the critical points, we hence define:

$$N_{\mathbf{p}} = \{\mathbf{q} \in \mathcal{P} \mid \mathbf{q} \in N_{\mathbf{p}}^{k\epsilon} \vee \mathbf{p} \in N_{\mathbf{q}}^{k\epsilon}\},$$

i.e. $\mathbf{q}$ is considered one of $\mathbf{p}$'s neighbours, already if $\mathbf{p}$ is one of $\mathbf{q}$'s.

To efficiently find the neighbourhood for each point, a kd-tree is built, containing all points in $\mathcal{P}$. The kd-tree supports the collection of the k nearest neighbours to a point in $O(klog^3|\mathcal{P}|)$ and can also be used to quickly retrieve all points in a sphere of radius ϵ. After the kd-tree has been constructed, we build the *proximity graph* $G(\mathcal{P}, \mathcal{E})$, with $\mathcal{P}$ as vertices and edges

$$\mathcal{E} = \{(i,j) \mid \mathbf{p}_j \in N_{\mathbf{p}_i}\}.$$

Please note that this graph is symmetric, and the adjacency lists of the graph correspond to the $N_{\mathbf{p}}$-neighbourhood of each point.

4.2 The Angle Criterion

The angle criterion projects all neighbouring points contained in $N_{\mathbf{p}}$ on the tangent plane and sorts them according to their angle around the centre sample, see figure 3, and computes the largest gap g between two consecutive projected neighbours. The basic idea is that g will be significantly larger for a boundary point than for an interior point, as illustrated in figure 3. Consequently, the boundary probability is given as

$$\Pi_{\angle}(\mathbf{p}) = \min\left(\frac{g - \frac{2\pi}{|N_{\mathbf{p}}|}}{\pi - \frac{2\pi}{|N_{\mathbf{p}}|}}, 1\right).$$

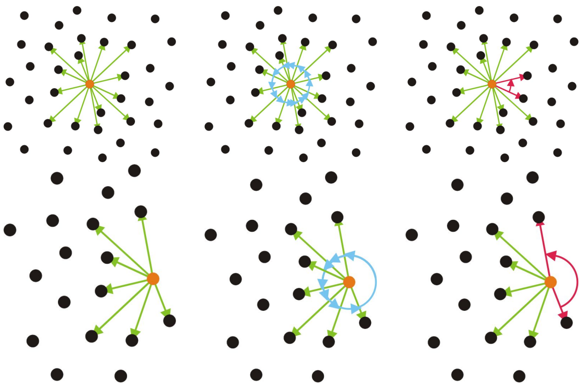

Figure 3: The three steps in the evaluation of the angle criterion for an interior point (top row) and a boundary point (bottom row). After projection into the tangent plane the difference vectors are generated (left). The projected points are sorted according to their angle around $\mathbf{p}$ (middle). The largest angular gap between two consecutive points is used to compute the boundary probability (right).

In contrast to the standard angle criterion, we ignore points $\mathbf{q} \in N_{\mathbf{p}}$ with a small scalar product $\langle \mathbf{n}_{\mathbf{p}}, \mathbf{q} - \mathbf{p} \rangle$. This way the angle criterion becomes less susceptible to small inaccuracies in normal direction.

4.3 The Halfdisc Criterion

In 2D-image processing, edge detection algorithms identify pixels, whose luminance deviates considerably from the average luminance of its neighbouring pixels. The same rationale can also be applied in our problem setting. On a 2-manifold, the neighbourhood of points in the interior of the surface is homeomorphic to a disc such that we can expect the difference between the point $\mathbf{p}$ itself and the average $\mu_{\mathbf{p}}$ of its neighbours to be small, as opposed to points on a boundary. Their neighbourhood is homeomorphic to a halfdisc (see figure 4), such that $\mu_{\mathbf{p}}$ will deviate from $\mathbf{p}$ significantly. Therefore, to derive a boundary probability, we compare $\mu_{\mathbf{p}}$ with the centre of mass of an ideal halfdisc in the tangent plane. To reduce the influence

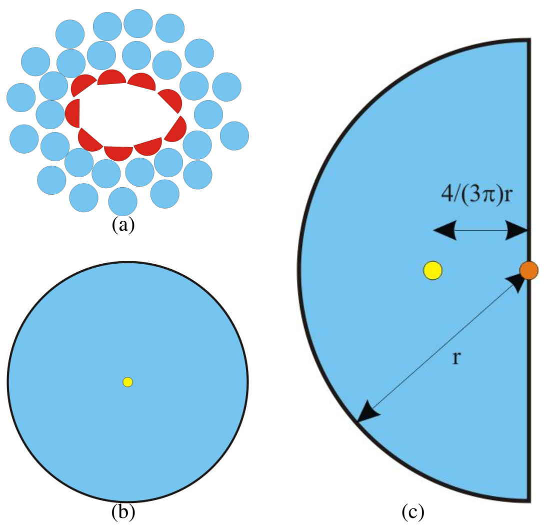

Figure 4: (a) The local neighbourhood of points located on the surface boundary is homeomorphic to a halfdisc as opposed to the full disc of an interior point. (b) For an interior point, the average of the neighbourhood points will coincide with the interior point itself, while for a boundary point (c), it will deviate in direction of the interior surface.

of variations in the sampling density, we compute $\mu_\mathbf{p}$ as a *weighted* average of $N_\mathbf{p}$ using the Gauss kernel

$$g_\sigma(d) = \exp\left(\frac{-d^2}{\sigma^2}\right),$$

where σ depends on the average distance to the neighbouring points $r_\mathbf{p}$ (namely $\sigma = \frac{1}{3}r_\mathbf{p}$), such that the influence of points outside the neighbourhood $N_\mathbf{p}$ can be neglected. This delivers:

$$\mu_\mathbf{p} = \frac{\sum_{\mathbf{q}\in N_\mathbf{p}} g_\sigma(\|\mathbf{q}-\mathbf{p}\|)\mathbf{q}}{\sum_{\mathbf{q}\in N_\mathbf{p}} g_\sigma(\|\mathbf{q}-\mathbf{p}\|)}.$$

To filter out properties of the surface itself and to include in our criterion properties of the sampling itself only, we compute the projection $\overline{\mu}_\mathbf{p}$ of $\mu_\mathbf{p}$ into the tangent plane and define the boundary probability as

$$\Pi_\mu(\mathbf{p}) = \min\left(\frac{\|\mathbf{p}-(\overline{\mu}_\mathbf{p})\|}{\frac{4}{3\pi}r_\mathbf{p}}, 1\right).$$

4.4 The Shape Criterion

As noted in [GWM01], the shape of the correlation ellipsoid of $N_\mathbf{p}$ approximates the general form of the neighbouring points. The shape of the ellipsoid is encoded in the eigenvalues $\lambda_0 \geq \lambda_1 \geq \lambda_2$ of the weighted covariance matrix $C_\mathbf{p}$:

$$C_\mathbf{p} = \sum_{\mathbf{q}\in N_\mathbf{p}} w(\mathbf{q})(\mu_\mathbf{p}-\mathbf{q})(\mu_\mathbf{p}-\mathbf{q})^t$$

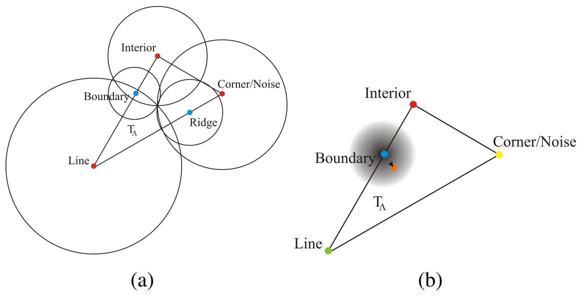

Figure 5: (a) The triangle formed by all Λ values with highlighted characteristic points for certain shapes. The circles passing through the triangle centroid $\mathbf{c}$ are shown for every shape in the respective colour. (b) The tentative probability $\widetilde{\Pi}_{boundary}$ is computed by evaluating the kernel around the characteristic Λ for boundary points.

We collect the relative magnitudes of the eigenvalues in a decision vector $\Lambda_\mathbf{p} = (\frac{\lambda_0}{\alpha}, \frac{\lambda_1}{\alpha}, \frac{\lambda_2}{\alpha})$, with $\alpha = \lambda_0 + \lambda_1 + \lambda_2$. There are four characteristic situations $\phi \in \Phi = \{$*Boundary*, *Interior*, *Corner/Noise*, *Line*$\}$:

$\phi =$ Boundary	$\Lambda_\phi = (\frac{2}{3}, \frac{1}{3}, 0)$
$\phi =$ Interior	$\Lambda_\phi = (\frac{1}{2}, \frac{1}{2}, 0)$
$\phi =$ Corner/Noise	$\Lambda_\phi = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$
$\phi =$ Line	$\Lambda_\phi = (1, 0, 0)$

The latter three values of Λ span a triangle T_Λ containing all possible values for Λ (fig. 5). We can now extract tentative classification probabilities $\widetilde{\Pi}_\phi$ for each of the situations described above from $\Lambda_\mathbf{p}$ by evaluating a spatial kernel around the characteristic Λ_ϕ. Again, we use a Gauss kernel g_σ with $\sigma_\phi = \frac{1}{3}\|\Lambda_\phi - centroid(T_\Lambda)\|^2$, effectively defining a radius of influence for the characteristic point (see figure 5, left). Now $\widetilde{\Pi}_\phi$ is for each shape $\phi \in \Phi$ given as

$$\widetilde{\Pi}_\phi(\mathbf{p}) = g_{\sigma_\phi}(\|\Lambda_\mathbf{p} - \Lambda_\phi\|)$$

Obviously, the regions for different shapes overlap. Therefore we normalise and define

$$\Pi_\phi(\mathbf{p}) = \frac{\widetilde{\Pi}_\phi(\mathbf{p})}{\sum_{\varphi\in\Phi}\widetilde{\Pi}_\varphi(\mathbf{p})}.$$

4.5 Combining the Criteria

Every criterion has its own advantages. Compared to the angle criterion, the halfdisc criterion is better capable of detecting small holes, especially when the hole is crossed by some edges of the neighbourhood graph, see figure 6.
On the other hand, while the halfdisc and the ellipsoid criterion typically find narrow bands of boundary

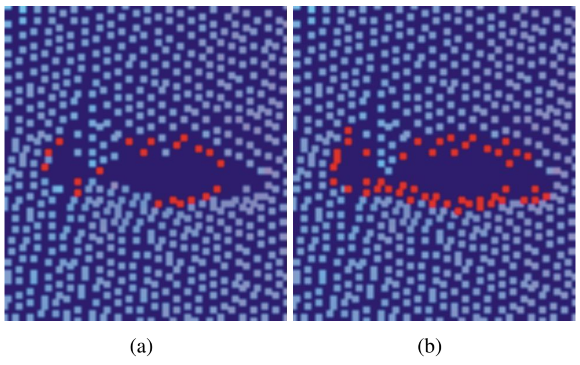
Figure 6: A small hole, crossed by some edges of the neighbourhood graph G. Points with a boundary probability (as computed with the angle criterion (a) and the halfdisc criterion (b)) above a threshold of 0.5 are coloured in red.

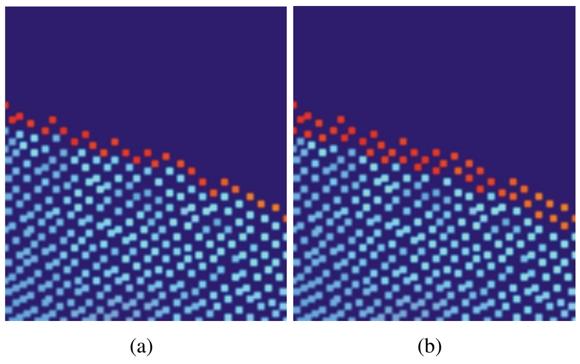
Figure 7: Boundary points detected by the halfdisc criterion form a band of boundary points (b), whereas the angle criterion finds a sharp boundary (a).

points around holes (in particular for larger k) the angle criterion is sharper and better exposes thin lines of boundary points (see figure 7). In the presence of noise, finally, the shape criterion performs best (see figure 8).

In order to make use of the different capabilities of the criteria and to increase the robustness of the boundary probability computation, we derive a combined boundary probability into a weighted sum

$$\Pi(\mathbf{p}) = w_\angle \Pi_\angle(\mathbf{p}) + w_\mu \Pi_\mu(\mathbf{p}) + w_\phi \Pi_{\text{Boundary}}(\mathbf{p}).$$

The weights $w_\angle$, w_μ and w_ϕ, where $w_\angle + w_\mu + w_\phi = 1$, can be adjusted by the user upon visual inspection. As default, a uniform weighting scheme produces good results, but for noisy models, the weight of the shape criterion should be increased.

4.6 Normal Estimation

Both, the angle and the average criterion, depend heavily on the normal at the point p. Therefore, if the data comes without normal information, a good estimation

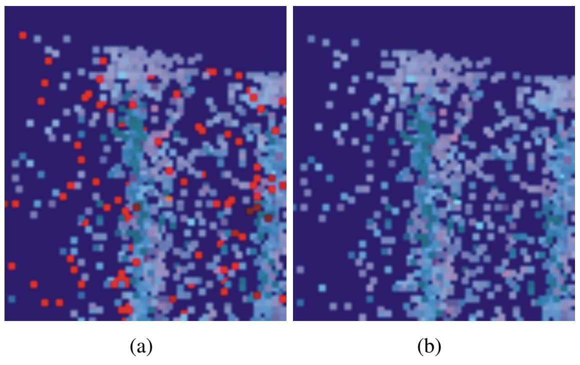
Figure 8: The angle criterion (a) identifies many false boundary points in the presence of noise, while the shape criterion (b) is not affected.

of the normal is mandatory. Following the normal estimation method by Hoppe et al. [HDD+92], the normal is given as the eigenvector corresponding to the smallest eigenvalue of the weighted covariance matrix of $N_\mathbf{p}$. In addition to that, we integrate information gath-

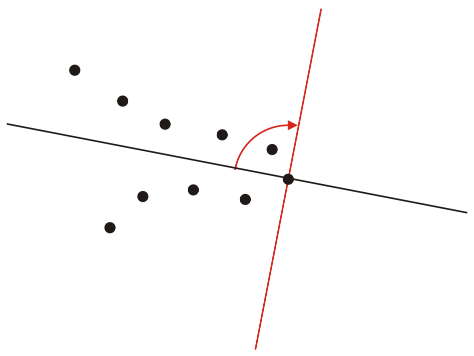
Figure 9: In sharp creases the fitting plane sometimes becomes normal to the surface. These cases can be detected with the angle criterion and the normal can then be flipped.

ered during the computation of the angle criterion in the normal estimation process as suggested in [LP02]. Sometimes, at sharp creases, the fitting plane is normal to the surface, see figure 9. To detect this situation, the angle criterion is evaluated after the normal has been estimated. If the boundary probability $\Pi_\angle(\mathbf{p})$ exceeds a given threshold, the estimated normal is rotated by 90 degrees about the axis defined by the two points on both sides of the maximum gap, projected into the tangent plane. Then the angle criterion is evaluated again, this time using the rotated normal, and the new normal is kept if the boundary probability has been reduced significantly, i.e. by more than 50%. This helps at sharp creases where sometimes the fitting plane is normal to the surface, see figure 9.

Please note that the estimation algorithm does not yield consistently oriented normals. Although this is required for neither of our criteria, it can easily be achieved by applying the minimum spanning tree traversal introduced in [HDD+92] on the neighbourhood graph. We use this approach for visualisation purposes.

5 Boundary Loops

The extraction stage of the boundary detection algorithm aims at producing a classification for each point, stating if it is a boundary or an interior point. Here, in addition to the boundary probability computed with the scheme described in the last section, we will exploit the coherence between boundary points. This greatly improves the robustness of our method. Moreover, connected loops of points, circumscribing a hole, will be found, providing immediate access to the boundary.

5.1 Boundary Coherence

Any point on a boundary loop has at least one neighbour to each side also belonging to the boundary. This property can easily be exploited using a simple iterative classification step. First, all points with a boundary probability greater than a user defined threshold are declared boundary points. Then, for each of these points, the two neighbours forming the maximum gap in the sense of the angle criterion are found. A point stays classified as boundary point if and only if both of these neighbouring points have also been declared boundary points. All other points are marked as interior points. This process is repeated until no more points change their status. Note that only the neighbours of points that did change status in the previous iteration have to be reconsidered in the following step, making the classification very efficient.

After the classification, we use an algorithm that is built upon the one presented in [GWM01] to construct a minimum spanning graph (MSG) based on the proximity graph G. By construction, this MSG will contain loops if and only if they correspond to the boundary loops we are interested in.

To this end, we use an extension of Kruskal's minimum spanning tree algorithm. The required *edge weights* $w(i,j)$, are derived similar to [GWM01] in two parts. The first component penalises the boundary probability of the adjacent points:

$$w_{\text{prob}}(i,j) = 2 - \Pi(\mathbf{p}_i) - \Pi(\mathbf{p}_j).$$

The second component incorporates the local sampling density measured by $r_{\mathbf{p}}$ (defined as the average distance to $\mathbf{p}$'s neighbours (see sec. 4.3) and penalises long edges so that the boundary loops will contain as many boundary points as possible:

$$w_{\text{density}}(i,j) = \frac{2\|\mathbf{p}_i - \mathbf{p}_j\|}{r_{\mathbf{p}_i} + r_{\mathbf{p}_j}}.$$

The total weight is then given by

$$w_{\text{total}}(i,j) = w_{\text{prob}}(i,j) + w_{\text{density}}(i,j).$$

The construction of the MSG uses an extension to Kruskal's minimum spanning tree algorithm. In the beginning, every vertex of G comprises a stand-alone component in G. Then all eligible edges are processed in ascending order, according to their weight. Here, an edge (i,j) is considered eligible only if $w_{\text{prob}}(i,j)$ and $w_{\text{total}}(i,j)$ are below pre-defined thresholds. A threshold combination of 1.1 and 3 proved good in our experiments and was used for all the examples given in this paper.

If an edge (i,j) connects two distinct components of G, the edge is added to the MSG and the two components are joined. If, on the other hand, the edge connects two vertices of the same component, it is included in the MSG only if the emerging loop is longer than a predefined minimum loop length e, measured as the number of edges in the loop. Similar to the radius ϵ in the construction of the neighbourhood graph G, the minimum loop length e steers the minimal hole radius to be found. Therefore, we link these two parameters and set $e = \frac{2\pi\epsilon}{d}$, where d is the average edge length in the graph.

5.2 Loop Extraction

With the MSG at hand, the boundary loops can be extracted using a breadth first search. The search is started once for each vertex in the MSG, unless it has become part of a loop already. The algorithm maintains for all vertices a colour value signaling one of three states: white (untouched vertices), grey (queued for visitation) or black (already processed). In the beginning, all vertices are white, except the origin, which is grey (see figure 10). In every step the vertex on the front is marked black, removed from the queue of grey vertices, and all its white adjacent vertices are marked grey and appended to the queue. If an adjacent vertex is black, it is ignored, but if one of the adjacent vertices encountered is grey, a loop has been found and can be extracted by tracing back the steps of the breadth first search. In a final step, points belonging to a loop are marked as boundary points. The process is illustrated in figure 10.

6 Results and Conclusions

We applied our algorithm to a variety of models. Figure 11 illustrates the effect of our hole detection method using the symmetric neighbourhood graph that is designed particularly to filter out even abrupt sampling density changes, a situation which causes even well-established hole criteria to fail. For this example,

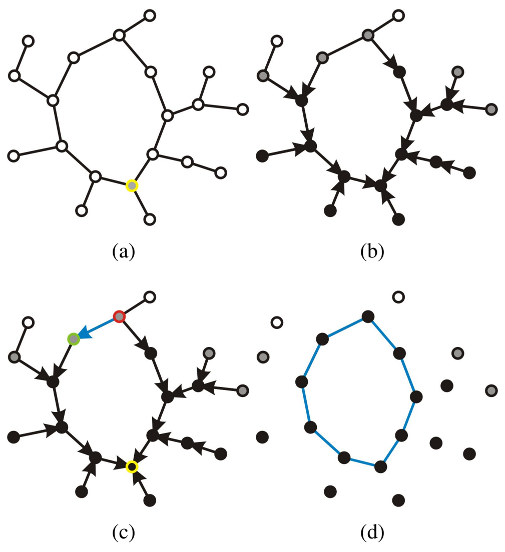

Figure 10: The extraction of a loop in one component of the MSG. (a) The breadth first graph traversal is spawned at the highlighted vertex (grey in the beginning). (b) The state of the vertices after four steps of the search. All grey vertices are queued for visitation, black vertices have been visited. Arrows indicate the vertices' predecessors. (c) When the adjacent vertices of the green vertex are examined, the grey vertex (red) is encountered and a loop has been found. The loop is extracted by tracing back the predecessors of both vertices. (d) The extracted loop.

one half of the depicted data set was heavily downsampled and only the angle criterion employed. Note how well the drastic change in sampling density is handled. Although this novel neighbourhood construction already considerably improves the performance of the so-called angle-criterion, the robust detection of holes in the presence of noise or also of holes of small size remains a challenge using only this criterion. To overcome this, we presented two novel boundary criteria: The halfdisc criterion is the 3d-analogue to the well-known border detection in images, whereas the ellipsoid criterion exploits a classification scheme based on local data analysis.
The notion of a hole is inherently and per-se ill-defined in the context of point set surfaces, and hence any classification ultimately needs to adapt to the application's (or rather the user's) interpretation. Consequently, our probabilistic approach can be trimmed using intuitive parameters, rendering the method easily adjustable to the task at hand. The parameter k of the neighbourhood definition determines the size of the local neighbourhoods. If k is increased, only larger holes can be detected, as smaller holes will be crossed by edges of G. We typically used a value between 12 and 25 for our test cases, depending on the amount of noise

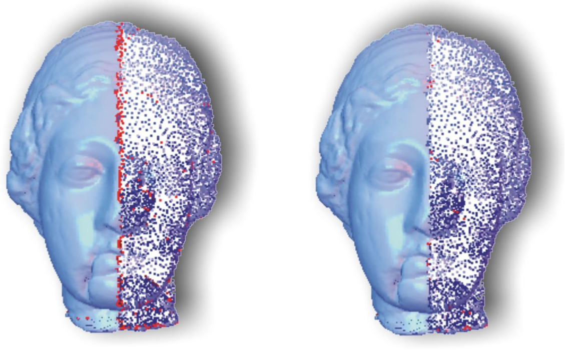

Figure 11: The effect of the symmetric neighbourhood relation. Left: k-nearest neighbours Right: Symmetric neighbourhood graph

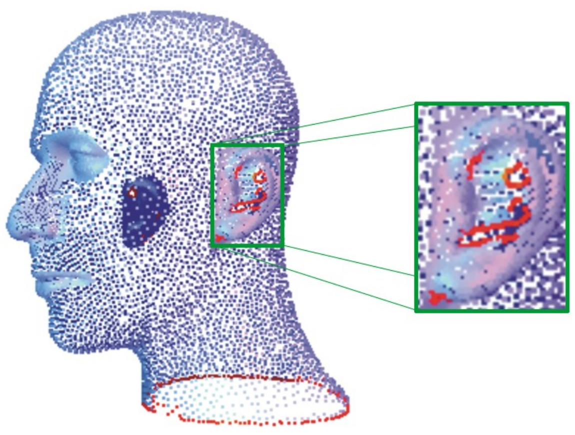

Figure 12: Boundary points identified in the mannequin model (points only) with $k = 15$. The top right image is taken from [DG01].

present in the data. If there is considerable noise, larger values of k can be used to improve the robustness of the hole detection, while the parameter ϵ can be used to define a minimum hole size, since the neighbourhood will stretch over all holes with a diameter less than ϵ. This way the user is enabled to focus on the important holes in the dragon for instance, as demonstrated in figure 14.
By making use of the coherence between boundary samples, the robustness of the hole detection is further increased. As a by-product of this stage, boundary loops are extracted, delivering subsequent processes direct access to the contours of the holes.
For many applications, such as automatic hole filling, the detection of holes has to be repeated after filling part of the hole. A reasonable efficiency of the hole detection is therefore desirable. In the dragon example (containing over 400000 points) the holes depicted in figure 14 (right) were detected in less than two minutes on a AMD Athlon 2.21 GHz processor. Specifically, the timings were: Construction of the kd-tree and the symmetrised proximity graph $23s$, computation of the

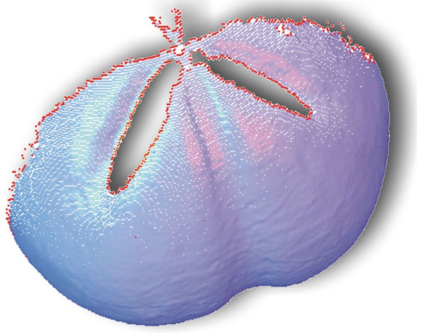
Figure 13: Boundary found in a scan of an echinite. All three criteria were combined with equal weight.

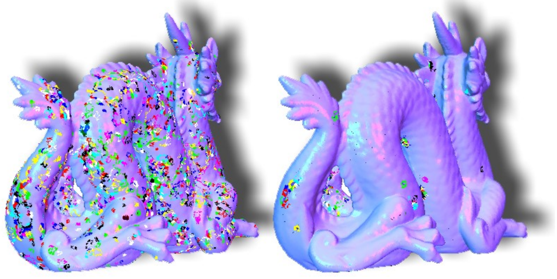
Figure 14: Numerous small holes are detected in the dragon model for $k = 15$, but larger holes can be isolated if all points within 0.01 of the bounding box diagonal are also included in the neighbourhood.

integrated boundary probability $46s$, extraction of the boundary loops $36s$. In the context of hole filling, the update of the boundary loops can naturally be performed incrementally, such that here timings can be expected to be even considerably faster; this has not been in the scope of this paper, though.
Figure 12 shows that our method extracts holes in the mannequin point cloud comparable to those identified for the corresponding mesh in [DG01]. Here, a classification step with a threshold of .3 was applied. In figure 15 the boundary of a single scan of the bunny has been extracted as a loop. A minimum loop size of $e = 1000$ was used to suppress the detection of loops around the smaller holes.

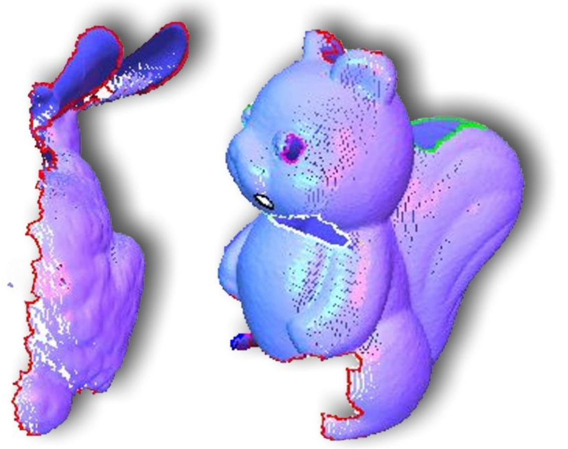
Figure 15: Detected boundaries in a single scan of the bunny and in the registered, yet incomplete squirrel data set.

References

[ABE98] N. Amenta, M. Bern, and D. Eppstein. The crust and the -skeleton: Combinatorial curve reconstruction. *Graphical models and image processing: GMIP*, 60(2):125–135, 1998.

[BK05] Mario Botsch and Leif Kobbelt. Real-time shape editing using radial basis functions. *Computer Graphics Forum*, 24(3):611 – 621, 2005.

[BSK05] Gerhard Heinrich Bendels, Ruwen Schnabel, and Reinhard Klein. Fragment-based surface inpainting. Poster proceedings of the Eurographics Symposium on Geometry Processing 2005, July 2005.

[CN04] Moenning C. and Dodgson N.A. Intrinsic point cloud simplification. In *Proc. 14th GrahiCon*, volume 14, Moscow, September 2004.

[DG01] Tamal K. Dey and Joachim Giesen. Detecting undersampling in surface reconstruction. In *Proceedings of the seventeenth annual symposium on Computational geometry*, pages 257–263. ACM Press, 2001.

[GWM01] Stefan Gumhold, Xinlong Wang, and Rob McLeod. Feature extraction from point clouds. In *Proceedings of 10th International Meshing Roundtable*, pages 293–305, Sandia National Laboratories, Newport Beach, CA, October 2001.

[HDD$^+$92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 71–78. ACM Press, 1992.

[LP02] Lars Linsen and Hartmut Prautzsch. Fan clouds - an alternative to meshes. In T. Alano, R. Klette, and Ch. Ronse, editors, *Proceedings Dagstuhl Seminar 02151 on Theoretical Foundations of Computer Vision - Geometry, Morphology and Computational Imaging*, page [10]. Springer-Verlag Berlin Heidelberg, 2002.

[PKKG03] Mark Pauly, Richard Keiser, Leif P. Kobbelt, and Markus Gross. Shape modeling with point-sampled geometry. *ACM Trans. Graph.*, 22(3):641–650, 2003.

[SACO04] Andrei Sharf, Marc Alexa, and Daniel Cohen-Or. Context-based surface completion. *ACM Trans. Graph.*, 23(3):878–887, 2004.