Detecting Holes in Surfaces

Ruwen Schnabel*

Institute of Computer Graphics
University of 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. This probability is used to present a robust and automatic hole detection algorithm. The power and feasibility of our algorithm is shown in several examples.

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 color. The surface is encoded implicitly therein and can only be extracted using some neighborhood relation between samples. While the lack of explicit connectivity information simplifies, compared to mesh based representations, the definition and implementation of deformation operations, as, for instance, encountered in geometric modelling, finding holes in the surface becomes an ill-defined problem. In this paper we will assume all surfaces to be 2-manifolds with boundary.

The knowledge of holes in the data 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 can profit 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.

While several authors proposed sampling conditions for surfaces to ensure correct reconstruction (most notably [1]), 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.

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.

[4], [6] as well as [2] apply what we shall call the angle criterion. The angle criterion considers for each sample point p a set of neighboring samples projected into the tangent plane of p. Then the projected samples are sorted radially and the angles between consecutive samples are computed. The maximum of these gaps is used to determine the probability of p being a point on a boundary. In addition [4] uses the correlation matrix formed by the neighborhood. The eigenvectors and values 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. These directions, together with the weights obtained with the angle criterion, are then used to weight edges of a neighborhood graph. In a next stage a minimum spanning graph, describing crease and boundary patterns, is extracted from the neighborhood graph.

In [3], undersampled regions are detected using the sampling requirement of [1]. It is based on the approximation of the medial axis by the 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 original surface needs at least one sample point within a ball defined by the local feature size and a factor r. This approach fails to identify holes in flat areas of the surface where only very few samples are required to fulfill this requirement and

*schnabel@cs.uni-bonn.de

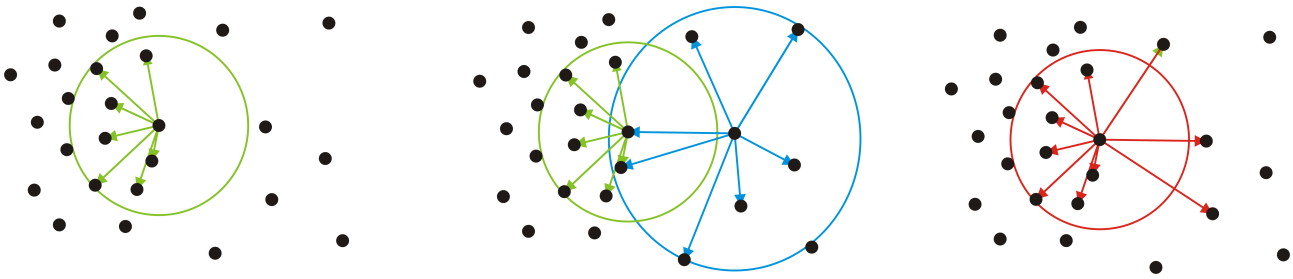

Figure 1: Left: The k-neighborhood of a point is biased towards the densely sampled region. The circles depict the radius of the k-neighborhood. Middle: Neighborhoods of points in the sparsely sampled region contain points in the densely sampled region. Right: The neighborhood of a point in the densely sampled region when the neighborhood graph is made symmetric.

holes are only visible if the surrounding area is oversampled. Also we are not interested in undersampled regions as usually observed at sharp creases where the sampling requirement can never be met.

3 The Algorithm

We use three criteria to identify points lying on a surface boundary. All expect the surface to be a 2-manifold and use a local neighborhood N_p for each sample point p. The widespread k-nearest samples neighborhood, denoted as $N_k(p)$, does not cope well with irregularly sampled surfaces. Therefore we extend this neighborhood definition in order to be able to handle these cases satisfactorily.

3.1 Neighborhood

We capture the neighborhood relation between samples in a directed graph $N = (P, E)$, P being the set of points. By N_p we denote all points in P adjacent to p in N, also we define the following radii for each point:

$$R_p = \max\{\|p - q\| \mid q \in N_p\}$$

$$\overline{R}_p = \frac{\sum_{i=1}^{k} \|p - q_i\|}{k}, q_i \in N_p$$

Starting point for the construction of N is a graph with E containing all edges (i, j) where p_j is one of the k-nearest samples to p_i. In points lying on the edge of a densely sampled region the k-neighborhood will be biased towards the densely sampled region and thus fails to find a neighborhood homeomorphic to a disc, characterizing interior points (see figure 1). Since we do not consider a drop in sampling density to constitute a hole, we overcome this deficiency by making the neighborhood relation defined by the k-neighborhood symmetric, that is

$$E = \{(i, j) | p_j \in N_k(p_i) \vee p_i \in N_k(p_j)\}$$

This definition is closely related to a measure of the local sampling density gained by the maximum distance of all points in $N_k(p)$ to p. Thus the neighborhood relation stretches over holes in a locally adaptive manner guided by local sampling density, see figure 1.

The above neighborhood definition handles well all situations in which the sampling density falls off sharply, as often observed where patches from different scans overlap. Nevertheless, small holes with dense sampling on all sides will still be detected. Usually this conforms with application requirements together with human intuition, but if the samples occur in small, extremely dense, clusters, or if for some application a minimum hole size is required, we also include in the neighborhood all points within a user defined minimum radius. Then only holes with a diameter larger than that radius will be found.

In unprocessed data gained through a 3D-scanning device there sometimes are outliers located inside a hole. Due to the symmetry of the neighborhood relation they often are included in the neighborhood of many boundary points so that no hole can be detected. This holds for the angle criterion in particular. To avoid such cases an outlier removal step as in [7] should precede the detection of holes.

3.2 The angle criterion

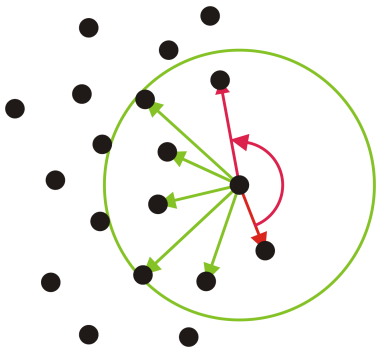

Figure 2: The angle criterion is based on the maximum gap, depicted in red.

As described above the angle criterion projects all neighboring points on the tangent plane and sorts them according to their angle around the center sample. Then the

largest gap g between two consecutive neighbors is computed and the boundary probability is given as

$$P_{angle}(p) = \min(\frac{g - (\frac{2\pi}{k})}{\pi - \frac{2\pi}{k}}, 1)$$

where k is the number of neighbors. Modifying the standard angle criterium described in [4], [6] and [2] we ignore all points in N_p with $\angle(n, p_i - p) < 10$, n being the normal in p. This way the angle criterium becomes less susceptible to small inaccuracies in the normal direction which is especially beneficial in the case of point clouds constructed from multiple range images, as here small errors in normal direction often cannot be avoided.

3.3 The weighted average criterion

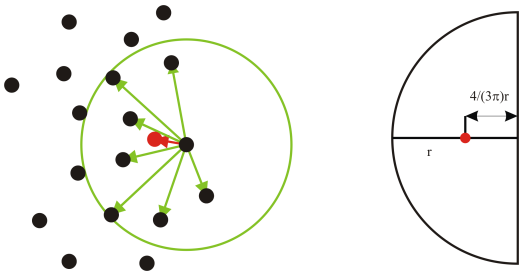

Figure 3: Left: The average criterion computes the average in the tangent plane. The difference vector in red points in direction of the interior surface. Right: The center of mass of a halfdisc is located at a distance of $\frac{4}{3\pi}r$ to center of the full disc.

Since the neighborhood of points on the boundary is homeomorphic to a halfdisc the average of the neighbors will deviate from the center point in direction of the interior surface. In fact we expect it to lie in the center of mass of a halfdisc in the plane, see figure 3. Thus we compute the weighted average

$$\mu_p = \frac{\sum_{i=1}^{k} w_p(q_i) q_i}{\sum_{i=1}^{k} w_p(q_i)}, q_i \in N_p$$

of the neighborhood and project it onto the tangent plane. We use a Gauss kernel

$$g_\sigma(d) = \frac{1}{\sqrt{2\pi}\sigma} \exp \frac{-d^2}{2\sigma^2}$$

with $\sigma_p = \max_{i=1..k} \|p - prj(q_i)\|$ for weighting with

$$w_p(q) = g_{\sigma_p}(\|p - prj(q)\|)$$

This reduces the influence of variations in the sampling density. The boundary probability can then be deduced from the distance between the center point and the projected average in the following manner:

$$P_\mu(p) = \min(\frac{\|p - prj(\mu_p)\|}{\frac{4}{3\pi}r}, 1)$$

where $r = \frac{1}{2k} \sum_{i=1}^{k} \|p - prj(q_i)\|$

3.4 The ellipsoid criterion

As noted in [4] the shape of the correlation ellipsoid of N_p approximates the general form of the neighboring points. The shape of the ellipsoid is encoded in the eigenvalues $\{\lambda_0, \lambda_1, \lambda_2\}$, where $\lambda_0 \geq \lambda_1 \geq \lambda_2$, of the weighted covariance matrix C_p.

$$C_p = \sum_{i=0}^{k} w(q_i)(\mu_p - q_i)(\mu_p - q_i)^t, q_i \in N_p$$

We collect the relative magnitudes of the eigenvalues in a vector $\Lambda_p = (\frac{\lambda_0}{\alpha}, \frac{\lambda_1}{\alpha}, \frac{\lambda_2}{\alpha})$, with $\alpha = \lambda_0 + \lambda_1 + \lambda_2$.

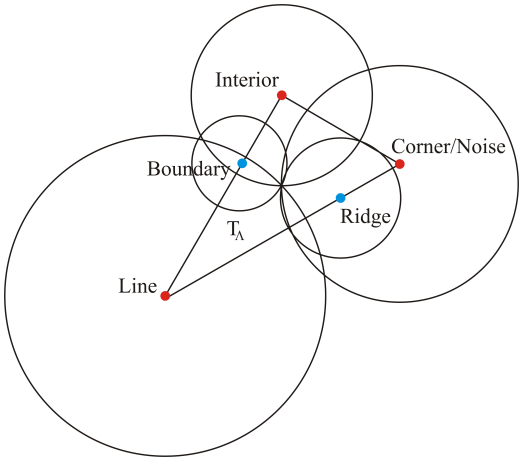

Figure 4: The triangle formed by all Λ values and the characteristic points for certain shapes. The circles show the 2σ radius for each shape.

In the case of a boundary point, the ellipsoid becomes an ellipse in the tangent plane and thus $\Lambda_{boundary} \approx (\frac{2}{3}, \frac{1}{3}, 0)$, for points on a ridge $\Lambda_{ridge} \approx (\frac{2}{4}, \frac{1}{4}, \frac{1}{4})$ applies. In interior points the ellipsoid degenerates to a circle and $\Lambda_{interior} \approx (\frac{1}{2}, \frac{1}{2}, 0)$ holds. For either a corner point or a noisy region there is no preferred direction and $\Lambda_{corner} \approx (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. Finally, for points on a line in space $\Lambda_{line} \approx (1, 0, 0)$. The latter three values of Λ span a triangle T_Λ, depicted in figure 4, containing all possible values for Λ. We can now extract tentative classification probabilities $\widetilde{P}_{shape}$ for each of the shapes described above from Λ_p by evaluating a spatial kernel around the characteristic Λ of the desired shape. We use a Gauss kernel g_σ for this with $\sigma_{shape} = \frac{1}{2}\|\Lambda_{shape} - centroid(T_\Lambda)\|$. Now $\widetilde{P}_{shape}$ is given as

$$\widetilde{P}_{shape}(p) = g_{\sigma_{shape}}(\|\Lambda_p - \Lambda_{shape}\|)$$

Obviously, the regions for different shapes overlap (see figure 4). Therefore we define the final probability P_{shape} as

$$P_{shape}(p) = \frac{\widetilde{P}_{shape}(p)}{\sum_{shapes} \widetilde{P}_{shape}(p)}$$

For the boundary probability, we found it advantageous to combine it with the line probability in the following manner:

$$P_{ellipsoid}(p) = P_{boundary}(p) + \frac{1}{2}P_{line}(p)$$

This is due to the fact that, because of gracing sensor views, close to boundaries, the sampling often is dense along lines, but sparse in between.

3.5 Normal estimation

Both, the angle and the average criterion, depend heavily on the normal in the point p. Therefore, if the data comes without normal information, a good estimation of the normal is mandatory. Similar to [5] the normal is given as the eigenvector corresponding to the smallest eigenvalue of the weighted covariance matrix of N_p. In a first step the points in N_p are weighted according to their euclidian distance to p using the Gauss kernel g_σ as above with $\sigma = \frac{R_p}{2}$. In the following steps, points are weighted using their distance to p in the estimated tangent plane and substituting $\sigma_p = \max_{i=1..k} \|p - prj(q_i)\|, q_i \in N_p$, similar to the weighted average criterion. This is repeated until convergence. In our experiments three iterations usually suffice. In addition, it is possible to integrate the angle

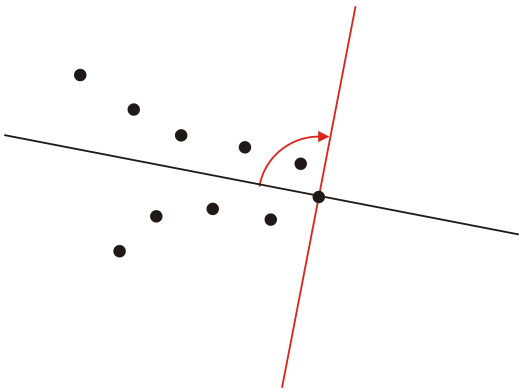

Figure 5: In sharp creases the fitting plane sometimes becomes normal to the surface. These cases can be detected with the angle criterion and the normal can then be flipped.

criterion in the normal estimation process as suggested in [6]. Sometimes, at sharp creases, the fitting plane is normal to the surface, see figure 5. To detect this situation, after the normal has been estimated, the angle criterion is evaluated. If the boundary probability $P_{angle}(p)$ 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 5.

The estimation algorithm does not yield consistently oriented normals. Although none of our criteria requires this, it can easily be achieved by applying the minimum spanning tree traversal introduced in [5] on the neighborhood graph. We use this approach for visualization purposes.

3.6 Combining the criteria

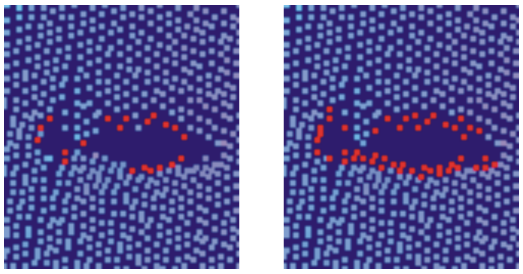

Figure 6: A small hole, that is crossed by some edges of the neighborhood graph. Left: Boundary detected by the angle criterion. Right: Boundary detected by the weighted average criterion.

Every criterion has its own advantages. Compared to the angle criterion, the weighted average criterion is better capable of detecting small holes, especially when the hole is crossed by some edges of the neighborhood graph, see figure 6. On the other hand, while the weighted average and the ellipsoid criterion sometimes find a narrow band of boundary points around holes, especially for larger k, the angle criterion differentiates better, see figure 7. Finally the ellipsoid criterium performs best in the presence of noise, see figure 8. Therefore, we propose two ways to combine the criteria. The first is a weighted average and the second a voting scheme. The combination in a weighted average is given by

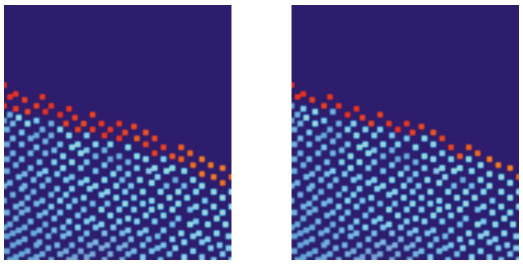

Figure 7: Left: The weighted average criterion finds a band of boundary points. Right: The angle criterion finds a sharp boundary.

$$P_{average}(p) = w_a P_{angle}(p) + w_\mu P_\mu(p) + w_e P_{ellipsoid}(p)$$

The weights for the average, w_a, w_μ and w_e, where $w_a + w_\mu + w_e = 1$, can be chosen by the user upon visual inspection. Usually a uniform weighting produces good results, but for noisy models the weight of the ellipsoid criterion should be increased since it is most robust to noise.

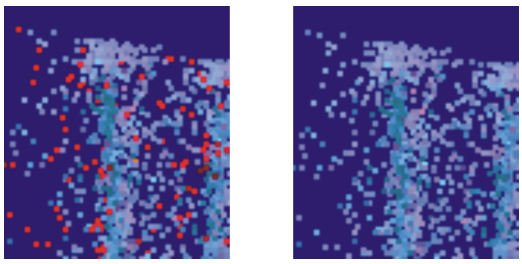

Figure 8: Left: The angle criterion identifies many false boundary points in a region of noise. Right: The ellipsoid criterion is not affected.

The voting scheme is based on a user defined threshold. A point is declared a boundary point only if at least two, or, even more restrictive, all three, probabilities exceed the threshold.

3.7 Extracting the boundary

The extraction stage of the algorithm aims at producing a classification for each point, stating if it is a boundary or an interior point. Here we will use the coherence between boundary points, having been neglected so far. This greatly improves the robustness of our method. Moreover, connected loops of points, circumscribing a hole, can be found, providing immediate access to the boundary.

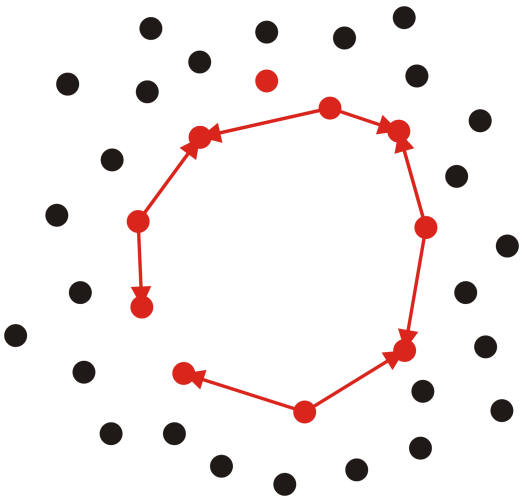

Figure 9: For points on a boundary loop, the two neighbors comprising the maximum gap will also be boundary points.

All holes in a manifold surface possess a closed boundary that we shall call a boundary loop. Thus any point on such a loop has at least one neighbor to each side, also belonging to the boundary (see figure 9). This property can easily be exploited using a simple and efficient 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 neighbors 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 neighboring 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. Obviously only the neighbors to points that did change previously have to be reconsidered in every iteration.

For extraction of the actual loops we use an algorithm that is similar to the one presented in [4]. The algorithm builds a minimum spanning graph on the neighborhood graph N. A minimum spanning graph (MSG) is based on a minimum spanning tree, in fact the minimum spanning tree is a subgraph of the MSG, but the MSG does contain loops under certain circumstances. These loops are what we will be interested in.

To construct the MSG, every edge in N needs to be assigned a weight $w(i,j)$. Similar to [4] we derive the weight in two parts. The first part uses the boundary probability of the adjacent points:

$$w_{probability}(i,j) = 2 - P(p_i) - P(p_j)$$

Only edges with $w_{probability}$ less than a user defined maximum are considered for inclusion into the MSG (we found 1.1 to be a good choice, suitable in most of our test cases). The second part incorporates the local sampling density measured by $\overline{R}_p$:

$$w_{density}(i,j) = \frac{2\|p_i - p_j\|}{\overline{R}_{p_i} + \overline{R}_{p_j}}$$

The total weight is then given by

$$w_{total}(i,j) = w_{probability}(i,j) + w_{density}(i,j)$$

Again, only edges with w_{total} less then a threshold become eligible for selection into the MSG. This second threshold is less critical though and we typically use a value of 3 for it. Alternatively, only edges running between points marked as boundary points in a previous classification step can be used. In contrast to [4], we do not use any vector valued penalties, rewarding edges in the direction of the boundary, approximated by the eigenvector corresponding to λ_1 in the ellipsoid criterium, since they did not improve loop quality in our experiments.

The construction of the MSG uses an extension to Kruskal's minimum spanning tree algorithm. In the beginning, every vertex of N is a stand-alone component of N. Then the edges are processed in ascending order, according to their weight. If an edge (i,j) connects to distinct components of N, the edge is added to the MSG and the two components are joined. If, though, the edge runs between two vertices of one component, it creates a cycle in the MSG. Such edges are included in the MSG only if the loop they create is longer than a predefined minimum loop length ρ, taken as the number of edges in the loop. To find out the loop length a breadth first search is spawned on the MSG that has been constructed so far, originating in p_i and counting the number of edges on the shortest path to p_j.

When the MSG is complete, the boundary loops can be extracted. Again breadth first search is applied. The algorithm maintains for all vertices a color value signaling one of three states: white, grey or black. White vertices have not been touched by the search yet. Grey vertices are queued for visitation while the search has already extended beyond black ones. In the beginning, all vertices are white, except the origin, which is grey. All grey vertices are managed in a queue. In every step the vertex on the front of the queue, becoming black, is removed 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 is grey already, a loop has been found. The loop can be extracted by tracing back the steps of the breadth first search along the two branches originating in the grey vertex and the vertex just removed from the queue. For this purpose we save with every vertex its predecessor. This search is started once for each vertex in the MSG, unless it has become part of a loop already.

In a final step, points belonging to a loop are marked as boundary points.

4 Results

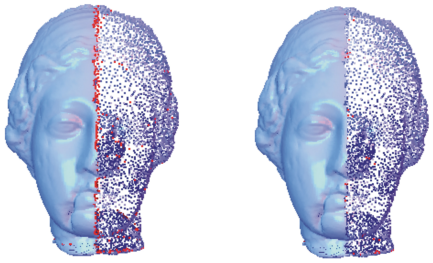

Figure 10: The effect of the symmetric neighborhood relation. Left: k-nearest neighbors Right: Symmetric neighborhood graph

We applied our algorithm to a variety of models. Figure 10 illustrates the effect of the symmetric neighborhood graph when using the angle criterion. Note how well the drastic change in sampling density is handled. Figure 11 shows that our method extracts holes similar to that found in [3]. A classification step with a threshold of .3 was applied. Figure 12 shows boundary points found by a weighted combination of all three criteria after a simple threshold value of .6 has been applied. Figure 13 and 14 demonstrate the effect of including all points within a user defined radius in the neighborhood relation. This way we are able to ignore the many small holes in the dragon. Figure 15 shows the performance of the ellipsoid criterium in the presence of heavy noise and many outliers. We used a value of $k = 40$. Note that our algorithm was unable to produce a consistent normal orientation due to the noise. In figure 16 the boundary of a single scan of the bunny has been extracted as a loop. A minimum loop size of

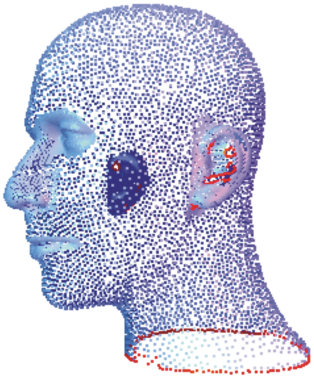

Figure 11: Boundary points identified in the mannequin model when a symmetric neighborhood graph constructed with $k = 15$ is used.

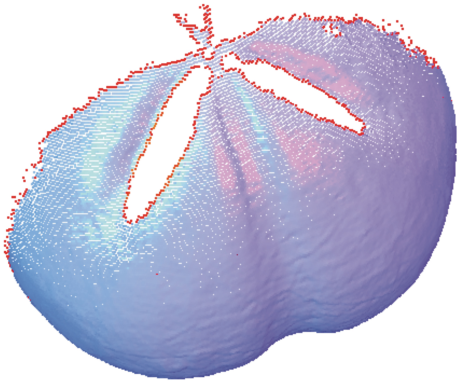

Figure 12: Boundary found in a scan of a sea urchin. All three criteria were combined with equal weight.

$\rho = 1000$ was used to suppress the detection of loops in the smaller holes.

5 Conclusions

We have presented a method to identify holes in point set surfaces. Our probabilistic method is based on the combination of three different criteria that assign to every point in the data set a boundary probability, i.e. the probability of being situated on a boundary of the point set.

Starting point of our hole detection method is a novel neighborhood construction that is designed particularly to filter out even abrupt sampling density changes, a situation which causes even well-established hole criteria to fail. Although this already considerably improves the performance of the so-called angle-criterion, holes cannot be robustly detected in the presence of noise or in cases when also holes of small size are to be detected.

To this end we presented two novel boundary criteria: The weighted average criterion is the 3d-analogue to the well-known border detection in images, whereas the el-

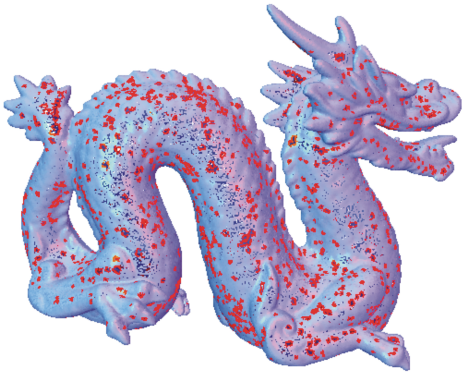

Figure 13: Numerous small holes are detected in the dragon model for $k = 15$.

Figure 15: In this extremely noisy scan, the ellipsoid criterion was used followed by a classification step. Due to the many outliers scattered in the holes not all of them could be found.

Figure 14: Only larger holes remain if all points within 0.01 of the bounding box diagonal are also included in the neighborhood.

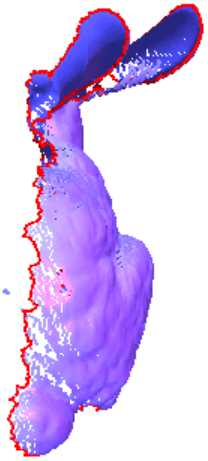

Figure 16: The boundary of a single scan of the bunny has been extracted as a loop.

lipsoid criterion exploits a classification scheme based on local data analysis.

As the notion of a hole is per-se ill-defined in the context of point set surfaces, any classification ultimately needs to adapt to the application's (or rather the user's) interpretation, and therefore our approach can be trimmed using intuitive parameters, rendering the method easily adjustable to the task at hand.

Also, in a second stage, our algorithm makes use of the coherence between boundary samples, further increasing robustness. As a by-product of this stage, boundary loops are extracted, delivering subsequent processes direct access to the contours of the holes.

References

[1] 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.

[2] Moenning C. and Dodgson N.A. Intrinsic point cloud simplification. In *Proc. 14th GraphiCon*, September 2004.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] Tim Weyrich, Mark Pauly, Simon Heinzle, Richard Keiser, Sascha Scandella, and Markus Gross. Post-processing of scanned 3d surface data. In *Symposium on Point-Based Graphics*, pages 85–94, 2004.