Consistent orientation of segmented models recovered from digitized data

M. Vančo **Guido Brunnett**

Computer Graphics and Visualization
TU Chemnitz
Germany
Marek.Vanco@informatik.tu-chemnitz.de
Guido.Brunnett@informatik.tu-chemnitz.de

ABSTRACT

Reverse Engineering addresses the problem of creating a CAD model for an existing physical object. In the data acquisition phase of the reconstruction process a discrete set of points on the boundary of the original object is created that serves as the database for all consecutive steps. The authors have developed a method to collect these points into segments that form the basic elements of a B-rep model for the 3D object.
In this paper we consider the important question of how to obtain a consistent orientation of the segmented model. For this three different methods are presented that are based on the following principles: orientation propagation, orientation via the boundary curves and orientation induction from surrounding 3D space. For these three strategies a detailed description of the corresponding algorithms on the segmented model are given. The performance of these methods for several benchmark objects is discussed.

Keywords: Reverse Engineering, Surface Reconstruction, Consistent normal orientation

1 Introduction

The problem of building a 3D model from an unstructured set of points measured from the surface of a given physical object appears in many areas including computer graphics, computer vision and reverse engineering [5]. Building such a model is a problem of growing importance since efficient 3D scanning technologies such as laser range scanning become more and more available. In order to sample 3D objects adequately, multiple scans have to be taken. Merging the measurements of each scan results in a large set of unstructured data points. Now, the challenge is to derive a surface model from the measured points automatically.

Before surfaces can be fitted to the data points, it is necessary to group these points into appropriate subsets, a process which is referred to as segmentation [9]. An automatic segmentation can be based on geometric properties as normals or curvatures, which have to be estimated from the data points. The efficient computation of these properties requires to build up a data structure that allows fast solution of elementary geometric operations as e.g. nearest neighbor search. A common way to achieve such a data structure is to compute a polyhedral approximation of the initial object based on the digitized point set [1, 6]. This approach may also be used to define an initial parameterization in the surface fitting step of the reconstruction process. However, the computation of the polyhedral approximation is expensive with respect to time and memory requirements.

In this paper we follow a different approach that avoids the construction of the polyhedral approximation. For the computation of k-nearest neighbors we employ a very efficient method that is based on median subdivision and a hashing strategy [7]. The corresponding data structure of this method serves as the basic data structure of our segment oriented reconstruction strategy, which is described in section 2.1.

Using the neighborhood information it is possible to estimate the normal vector in each data point. We implemented three different methods for normal vector estimation, because the quality of the segmentation method depends on the accuracy of the estimated normals. The description of these methods together with critical evaluation of the results obtained can be found in section 2.2.

The segmentation method described in section 2.3 follows a region growing approach, i.e. each segment is built from a seed point by recursively adding neighboring points as long as the desired criteria are fulfilled. The criteria involve the angle between the normals of neighboring points as well as the angle between the current point normal and a reference vector of the current segment.

This initial segmentation is not completely satisfying because of the appearance of a large number of small segments accompanying the larger reasonable segments. Therefore we developed a sequence of cleanup procedures in order to remove these flaws in the segmentation. As the result of these procedures small segments are joined with neighboring larger ones until remaining small segments correspond to characteristic features of the surface as sharp edges or regions of high curvature. In order to simplify the segmentation these small segments are then joined into so called 'connected segments'. If the object contains sharp edges (which will be very often the case) it is important to recompute the normals close to the edges. For this purpose we developed a procedure that operates with modified neighborhoods for points that lie close to the edges of the current segmentation. A crucial issue of this method is to avoid the recomputation of normals in situations where neighboring segments occur with a smooth transition.

In section 3 we address the important problem of creating a consistent orientation of the segmented model. Three different principles are considered:

1. **Orientation propagation**. In this approach the orientation of one segment is arbitrarily chosen and propagated over the whole surface.

2. **Orientation based on oriented boundaries**. A boundary model of a 3D object is correctly oriented if the common boundaries of adjacent faces appear in both orientations in the boundary model. In order to utilize this principle for our segment model it is necessary to create oriented boundary curves for the segments. This is done by creating a triangulation of segments and extracting the boundary curve from the triangulation. Note, that the triangulation of a segment is in itself a complex reconstruction problem. However, for the purpose of orientation it is not necessary to reconstruct the exact boundaries of the segment. Therefore, we developed a fast heuristic algorithm that provides an approximation of the segment boundaries.

3. **Orientation induced from 3-space**. The basic idea of this method is to find the extreme points of the data set. Under the assumption that the object to be reconstructed is closed these extreme points can be used to specify the orientation of the corresponding segments. In order to increase the number of extreme points we perform a subdivision of the bounding box of the data set and apply the strategy to the subset in a similar fashion. Note, that this method can also be applied to objects that possess only a few number of small holes, since the orientation flaws will only appear in the neighborhood of the holes and will not spread over the whole surface. Therefore these flaws can be detected and removed by a postprocess.

In section 3 of this paper we provide a detailed description of the algorithms that correspond to the mentioned orientation principles. The advantages and shortcomings are discussed and a practicable orientation method is obtained by combining these principles. The results of the methods are illustrated for several examples.

2 Creating segmented models from digitized data

2.1 k-nearest neighbors computation

The segmentation process is based on geometric properties of the surface of the object to be reconstructed. In order to estimate these properties it is necessary to know the local neighborhood of each surface point. In our approach we approximate the neighborhood of a point P by the set of k-nearest neighbors of P. This set provides sufficient information on the local behavior of the surface and it can also be used for point-based parsing of the whole object.

Our method for the computation of k-nearest neighbors is based on the following steps:

Data organizing:

- Subdivision of the bounding box of the point set into non-overlapping rectangular regions. During the subdivision a binary tree is created, whose leaves are the regions with points. For the subdivision the median value is used in order to guarantee that every region contains the same number of points (up to one point).
- For every region a hash table is allocated and all points of the region are projected into this table.

Searching:

- Find in the binary tree the region which contains the point P.
- Search in the corresponding hash table for k-nearest neighbors and use the k_{th} neighbor to determine the searching sphere.
- If the searching sphere intersects or contains adjacent regions, search in the hash tables of these regions.

For detailed description see [7].

Note that with our algorithm, we also could compute the approximate nearest neighbors by extending the procedure with a tolerance parameter ε.

There it has been shown that our method computes the k-nearest neighbors very efficiently with linear memory complexity. Compared to other efficient searching method based on KD-Tree [4], our algorithm proved to be superior for $k \leq 25$, which is sufficient for the applications in surface reconstruction.

2.2 Approximation of the normal vectors

In the following we will assume that the set $N(P)$ of k-nearest neighbors of a point P computed with respect to the Euclidean distance is the same as the set of the k-nearest neighbors of P with respect to geodesic distance, i.e. on the surface of the initial object. Note, that it is this assumption that allows to estimate higher order surface properties as normal vectors from the data set.

Since the quality of the segmentation highly depends on the accuracy of the estimated normal vectors, we implemented and compared several methods, in order to find a reliable technique to approximate the normal vectors from the neighborhoods.

A very common approach to approximate the normal in P is to compute the plane of regression R (or best-fit plane), see [2], of the data set $N(P) \cup \{P\}$ and to use the normal of R as the approximation. In [8] we introduced three different methods for the normal vector estimation based on the following principles: local centre triangulation (LCT), local Delaunay triangulation (LDT) and approximation with a analytic surface (AwAS).

In the LCT we compute the centre of mass C_m of the neighborhood $N(P)$ and an appropriate projection plane E (for details for computing E see [8]) . We project $N(P)$ and C_m to E, obtaining planar points. For all planar points we take two adjacent points (in the sense of polar coordinates direction) and create a triangle containing these two points and C_m. We map all triangles back to the 3D space, compute their normal vectors and the estimation of the normal vector in P is the normalized sum of normal vectors of all triangles.

The LDT method is similar as the LCT ($N(P)$ is projected to E), but Delaunay triangulation of the planar points is performed. The normal vector estimation is the normalized sum of the normal vectors of all triangles incident with P.

The AwAS uses a first estimation of the normal vector N_s in P, that can be obtained from LCT or LDT. We define a new local orthogonal coordinate system, so that P is the origin and N_s coincides with the z-axis of this coordinate system. The neighborhood of P and P are transformed from this new coordinate system into the global coordinate system and approximated by a quadratic or cubic surface $z = f(x, y)$ using the least square method. The estimated normal in P is the inverse transformation of the normal vector of the surface in P.

For artificial data set without noise the LDT estimates the normal vectors very well, even if the normal vectors have to be approximated close to the edges. However, for noisy data sets this method showed stability problems. A further disadvantage results from the fact that the complexity increases fast with the size of the neighborhoods.

The LCT method works faster than the LDT but provides normals that are less exact. On the other hand, it works more stable than the LDT for noisy data sets.

The AwAS estimates the normal vectors perfectly, if they belong to a smooth surface without noise. For noisy point sets it works more stable than the LCT.

Our tests have shown, that the appropriate neighborhood size for point sets with noiseless sampling is in the range of $k = 10$. For this neighborhood size the estimation was incorrect only on sharp edges. Increasing of the neighborhood size results in a smoothing effect of the normal vectors in the vicinity of the edges, but did not improve them on smooth surfaces.

For noisy data sets the neighborhood size has to be increased in order to obtain satisfactory results. However, these large neighborhoods result in a strong smoothing close to the edges. Extensive tests showed, that the optimal choice of the estimation method should be made depending on the surface shape. In general the most reliable method for badly scanned objects seems to be the combination of best-fit-plane with AwAS, where the neighborhood size should be chosen in the range of 20-30 neighbors.

2.3 Segmentation of the surface based on normal vectors

Our segmentation method is based on the normal vectors and uses two angles α, β to subdivide the surface into point clusters such that the following angle criteria are fulfilled:

- the angle between the normal vectors N_r, N_i of two adjacent points P_r, P_i ; $P_i \in N(P_r)$ has to be smaller than α ; $(\angle(N_r, N_i) < \alpha)$.
- the angle between the normal vector N_i of a new point P_i, which is to be added, and the reference vector N_{ref} of a segment has to be smaller than β $(\angle(N_i, N_{ref}) < \beta)$. Note, that the reference vector N_{ref} of a segment is the normalized sum of normal vectors of all points in the segment.

This initial segmentation has the following undesirable property: besides the larger segments (of a size mainly controlled by β) that provide a reasonable segmentation of the data set, a huge number of small segments (number of points < 20) are produced that are certainly unwanted. In order to reduce the number of small segments we implemented three procedures to clean up the segmentation. The first of these procedures joins a small segments with a neighboring larger one, if this process violates the β-criterion only by a prescribed tolerance ε. More precisely we proceed as follows:

Cleaning-up, pass 1:

For all small segments S_f do:

- Consider the set NP of all neighbors of all edge points of S_f. Store for each segment the number of points of NP that it contains. We call this entry the neighborhood index.
- Examine all segments with a neighborhood index bigger than a specified threshold (e.g. 40% of all neighbors). Among these find the segment S with the smallest angle deviation η between its reference normal and reference normal of S_f. If $\eta < \beta + \varepsilon$ join S_f and S. Update the reference normal of S.

Remaining small segments are processed by a second procedure that intends to reduce the size of a small segment by repeatedly extracting its edge points.

Cleaning-up, pass 2:

Let S_1 denote a small segment

- Search for an edge point P_i of S_1 with a neighbor Q_{ij} that belongs to a large (regular) segment S_2 with reference normal N_{ref}.
- If such a point exists and the relations $\angle(N_{P_i}, N_{Q_{ij}}) < \alpha \ \wedge \ \angle(N_{P_i}, N_{ref}) < \beta + \varepsilon$ hold, then add P_i to S_2 and update the list of edge points of S_1
- Repeat until the list of edge points of S_1 is empty or all of its points are marked as not extractable.

Small segments that still remain after execution of the second cleaning-up procedure belong in general to one of the following categories:

a) they describe regions with high curvature or bad sampling, where the normal vector estimation procedure failed even for extended neighborhoods.

b) they occur along sharp edges of the object.

The third procedure intends to simplify the segmentation by joining the remaining small segments. If one small segment has at least one edge point with a neighbor in a second small segment, these segments are connected. Segments that result from the process of joining small segments to one segment are called *connected segments*. Small segments that cannot be joined with other segments are called *isolated*. All other segments are referred to as *regular* ones.

2.3.1 Recomputing of the normal vectors

During the normal vector estimation the normal vectors close to the edges are estimated inaccurately because the neighborhoods contain points on both sides of the edge. This is especially the case if the data set results from a poor sampling that requires the use of large neighborhoods. These normal vectors often strongly affect the segmentation process and can cause a large number of small segments. Therefore we implemented a procedure for normal vector recomputation, which searches for the points in the vicinity of sharp edges and improves the estimation of their normal vectors by modification of the point neighborhood.

Since the current segmentation provides rough information about the object's surface, we can approximately detect edges or regions with high curvature. Therefore the neighborhood of a point P in a regular segment can be temporarily adjusted, so that $N(P)$ does not contain neighbors beyond sharp edges.

For every segment a list of its segment neighbors is created and the normal recomputation procedure proceeds as follows:

- Take a point P_i of a segment S_j.
- Mark all segment neighbors S_m of S_j, with the property $\angle(N^m_{ref}, N^j_{ref}) < \delta$, where N^m_{ref} and N^j_{ref} are the reference vectors of the segments S_m and S_j resp. and δ is a threshold.
- Check if all neighbors of P_i belong to the marked segments. If not, remove them from $N(P_i)$.
- If the neighborhood was not changed, continue with a next point. Otherwise extend $N(P_i)$ to a full neighborhood as follows:
  - Take the neighbors of the points in $N(P_i)$, which belong to marked segments and have not yet been included into $N(P_i)$.
- Estimate the normal vector of P from the modified neighborhood using one of the described estimation methods and discard the modified neighborhood.

The procedure recomputes only normal vectors of regular segments (normal vectors in connected segments can be recomputed e.g. by increasing of the neighborhood). The whole process (segmentation with subsequent recomputing) can be repeated, but as the tests have shown, more than two repetitions gives only very small improvements.

The recomputation works very fast (the worst case complexity is $O(n)$, but in general only normal vectors of edge points are recomputed) and for an object with many sharp edges it provides good improvement of the normal vector estimation. If the transition between two segments is smooth, then the normal vectors of the boundary points of these segment were well estimated with the initial estimation procedure and it is undesirable to recompute these normal vectors. It is the role of the angle δ to avoid the modification of the neighborhoods on this situation.

Figure 1 shows the effect of normal vector recomputation for an object with a hole and many sharp edges.

3 Creating a consistent orientation

In this section we address the problem of creating a consistent orientation of our segmented model. If one considers the possible arrangements of normal vectors of two adjacent points it is apparent, that the orientation problem cannot be solved without additional surface information or some restrictions, see Figure 2: N_{OK} is an already oriented normal vector and $N_?$ is the normal vector to be oriented. In the example a) in both cases the angle between N_{OK} and $N_?$ is the same, but the orientation of $N_?$ on the left side should be inverted, while on the right $N_?$ is oriented correctly. The same problem occurs in the example b) where $N_?$ lies perpendicular to N_{OK}: it cannot be decided directly which orientation of $N_?$ is the correct.

In the following we investigate three approaches for consistent normal vector orientation, which

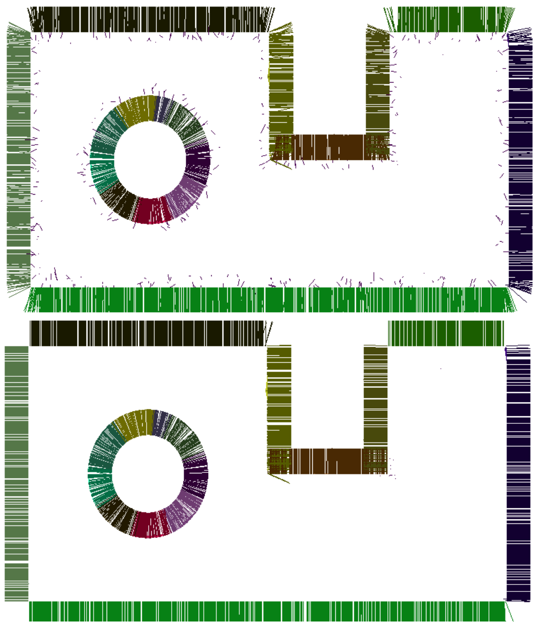

Figure 1: First step normal estimation (upper image) and the normal vectors after recomputing (lower image)

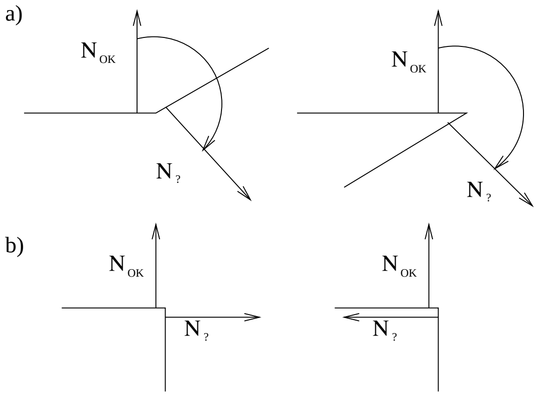

Figure 2: Normal vector orientation problems

are based on the following principles: orientation propagation, orientation via the boundary curves and orientation of extreme points.

3.1 Orientation propagation

The basic idea of the orientation propagation is to choose the orientation of one normal vector arbitrarily and to propagate this orientation over the whole surface following a path of minimal angles between adjacent normal vectors. An implementation of this strategy has been described in [3], where a minimal spanning tree (MST) of all points in the point set is computed and the edges of the tree are weighted by the angles between corresponding normal vectors. For smooth surfaces this method works very well, however, if a sharp edge occurs in the surface it is necessary that a smooth transition of the normals is created artificially. This is done by choosing a huge neighborhood that causes a blending process of the normals on both sides of the edge. In complex objects with many high curvature parts this huge neighborhood can cause wrong normal vector estimation and the propagation procedure fails.

Therefore we try to orient the normal vectors consistently by using the reference vectors of the segments. The procedure works analogously to the method with MST of angles between adjacent normal vectors: we build a minimal spanning tree of the angles between reference vectors of the segments. It is obvious, that we have the same problem as the MST of the normal vector angles: if the minimal angle between the reference vector $N_{ref}(S)$ of a segment S and its neighbors S_n with the reference vector $N_{ref}(S_n)$ falls into an interval $(\gamma, \pi - \gamma)$ it cannot be decided from the MST, which direction of the $N_{ref}(S_n)$ is the correct. The threshold γ divides the whole angle interval $[0, \pi)$ into three subintervals:

- interval $[0, \gamma]$: $N_{ref}(S_n)$ is oriented correctly with regard to $N_{ref}(S)$
- interval $[\pi - \gamma, \pi)$: $N_{ref}(S_n)$ is oriented opposite with regard to $N_{ref}(S)$
- interval $(\gamma, \pi - \gamma)$: it is unreliable to determine to correct orientation of $N_{ref}(S_n)$ with regard to $N_{ref}(S)$

For the third case we implemented another method based on the segment boundaries, which is introduced in the next section.

3.2 Orientation based on oriented segment boundaries

3.2.1 Triangulation of the segments

The angle β in the segmentation process limits the maximal curvature of the segment. We use such values for this angle, so that a $2\frac{1}{2}$D triangulation is possible. Our triangulation tool works basically as follows:

A triangulation is created for each of the regular segments. This is done by projecting the points of the segments to a plane, creating a Delaunay triangulation of this point set and modifying this initial triangulation such that the intuitive shape of the point set is approximated by the resulting triangulation. This planar triangulation is then back-mapped into 3D space.

It is known that the Delaunay triangulation triangulates the convex hull of the point set. The segments are not convex in general and therefore some triangles have to be removed from the triangulation in order to approximate the segment shape. As mentioned before, for the purpose of orientation it is not necessary to reconstruct the boundary of the segments exactly. Therefore we developed a heuristic method based on the histogram of the edge lengths, see Figure 3.

- The whole interval of possible edge lengths (the interval between the shortest and longest edge) of the triangulation is divided into subintervals (the number of intervals is a parameter of the procedure).
- Each edge length is projected into the corresponding interval and the occurrences of the projections in every interval are counted.
- The interval containing the *last maximum* number of occurrences (for detailed explanation see paragraph below) is found and from the position of that interval the critical length L_c is computed. L_c is the length of the largest edge, that is not removed from the triangulation.
- If the length of any edge of a triangle is longer than L_c the triangle is removed from the triangulation.
- If the length of any edge of a triangle is longer than $\kappa_2 L_c$ and the triangle lies on the boundary of the triangulation, the triangle is removed from the triangulation.

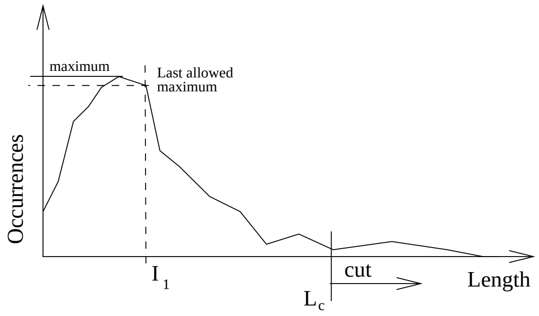

Figure 3: Edge length based triangle removal from a triangulation using histogram

As the last interval containing the maximum number we take an interval whose number of projections deviates from the real maximum value of a constant κ_1, i.e. the last interval with the number of projections greater than κ_1∗`Max`, where `Max` is the maximum number of projections and κ_1 is a parameter. This interval is taken as the position for the computation of the critical length (in Figure 3 the value I_1 is the position, from which L_c is computed). This parameter helps especially for convex segments, where the histogram has a shape similar to a Gaussian distribution of the lengths and it could happen, that the maximum would occur in the beginning of the histogram (in Figure 4 the value M_1) and too short critical length (computed from I_1) could cause removal of many inner triangles, which belong to the segment shape.

The second parameter κ_2 specifies the tolerance of the computed critical length L_c for triangles on the boundary of the triangulation. The critical length is chosen such that no undesirable holes in the segment result, which do not belong to the shape. Therefore from the boundary of the triangulation triangles with shorter edges than L_c can be removed, but not from the inside (i.e. the triangles in the interior of the triangulation have bigger weight than the boundary ones).

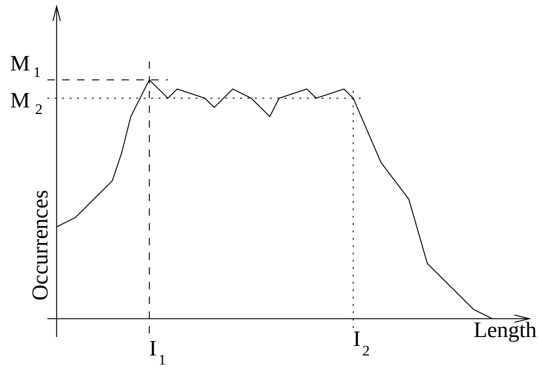

Figure 4: Convex segments length distribution

Figure 5 shows a quite complex segment after triangle removal.

The triangle removal method works very satisfactory and automatic for the most segment types. Only if there are various sampling depths in one segment, which strongly differ each other, this method creates many holes in the interior or leaves many triangles with long edges on the boundary, see Figure 6, where undesirable holes in the interior of the segment were created.

3.2.2 Orientation of segment boundaries

It is obvious that the normal vectors in two neighboring segments are oriented consistently if their

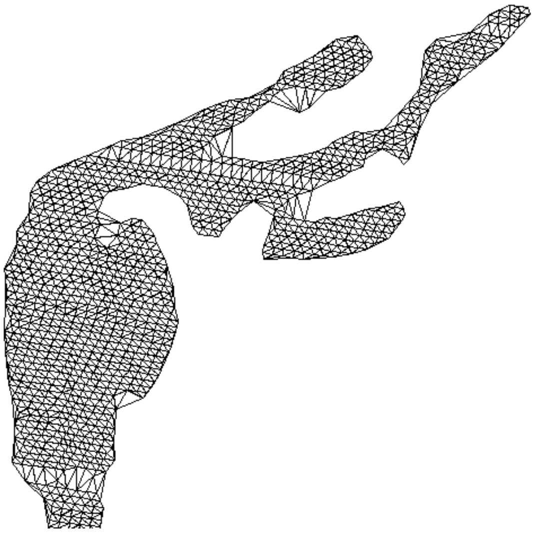

Figure 5: Segment shape of a real object after edge removal

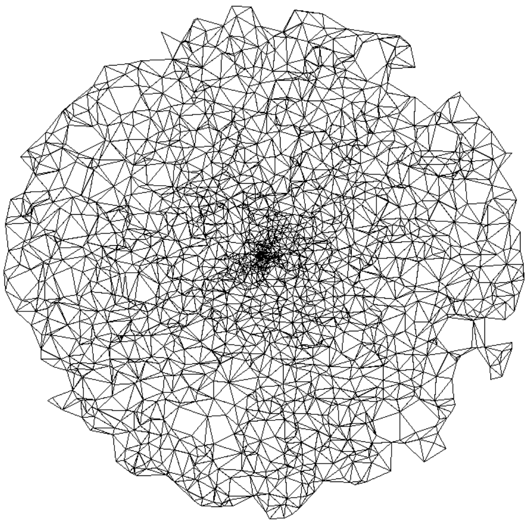

Figure 6: A circular segment with variable sampling depth

triangulations are oriented consistently, i.e. the neighboring parts of the boundaries are traversed in opposite directions (see Figure 7).

Assume the segment S_1 in the Figure 7 was already oriented correctly and we are going to orient the segment S_2:

- Find on the boundary of the S_1 a chain of points (1-2-3-4-5-6; at least two), whose neighbors lie on the boundary of the segment S_2.
- Make from these neighbors a chain of points (a-b-c-d-e-f-g) and compare both directions: $1 \rightarrow 6$ and $a \rightarrow g$.
- If they are not opposite, invert everything in S_2 (normal vectors of points, triangulation, boundary and reference vector).

We tried to orient the segments consistently based on their triangulation boundaries. However, we found that the result depends on the shape of the segment. For several special cases: degenerated segment shapes (e.g. thin triangulation), overlapping triangulations, many small segments, where the common point chains could be found, etc. this strategy was not successful.

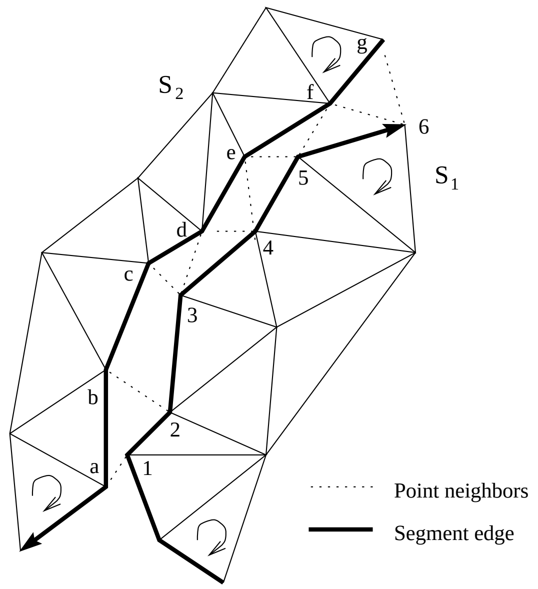

Figure 7: Making consistent orientation of S_2 according to S_1

Therefore we developed a method that combines the principles of orientation propagation and boundary orientation.

If we detect, that the current minimal angle of the MST falls into the third interval (mentioned in the section 3.1), i.e. $(\gamma, \pi - \gamma)$, we take all marked (already oriented) neighbors of a segment S_i, which is to be oriented and sort them in an array by the size (the biggest one in the beginning). Then we take the first neighbor segment S_j from the array and try to orient S_i according to S_j using the boundaries of both segments. If it fails we take the next neighbor segment and so on. If the boundary orientation procedure fails for all neighbor segments in the array, we mark S_i as "hard to orient" segment. At the end of the whole process we take all "hard to orient" segments and try to orient them again (the majority of their neighbors has been marked and the probability for a success is higher).

3.3 Orientation induced from 3-space

To make our orientation procedure more robust and stable we search for places on the object, where we can determine the normal vector orientation without neighborhood information.

The main idea is based on the fact that the normal vector orientation on the surface of a closed object can be determined uniquely for 6 extremal points $(X_{\min}, X_{\max}, Y_{\min}, Y_{\max}, Z_{\min}, Z_{\max})$. In ideal case we could start with 6 correctly oriented segments, which is not enough for complex objects, therefore we subdivide the bounding box of the object into non-overlapping boxes until every box contains maximally m points. During the subdivision for every box its 6 neighbors are stored in a appropriate data structure. If a box contains zero points and it has less than 6 box neighbors, i.e. it lies on the boundary (exterior box), this box is deleted. Except for boxes with nonzero number of points, the boxes with zero points and 6 box neighbors remain (interior boxes), see Figure 8 for an example for a 2D subdivision.

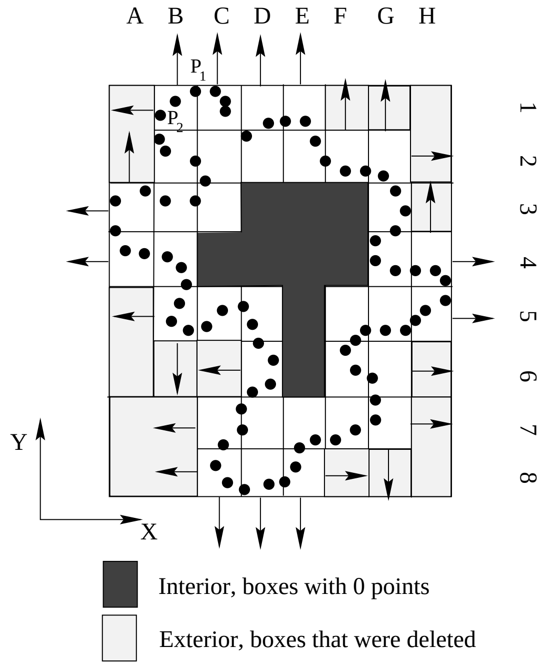

Figure 8: Subdividing the 2D bounding box into boxes

After the space subdivision has been created, the procedure proceeds as follows:

- We denote the *current minimal angle (M_{CA})* as a segment variable, which specifies the minimal angle between a point of the segment and a unit direction $((1,0,0),(-1,0,0),(0,1,0),\cdots)$. In the beginning this variable is initialized with the γ threshold, see section 3.1.

- Take a first box B_i with nonzero number of points and check its box neighbors. If in some direction the neighbor is missing, i.e. B_i is a boundary box, find the extremal point in B_i in the specified direction: in the Figure 8 e.g. the box B1 has two free directions: the positive Y and the negative X. Thus, two points are searched: the first one with the maximal Y coordinate (P_1) and the second one with the minimal X coordinate (P_2).

- The angle between the normal vector of P_1 and the vector $(0,1)$ is computed (thereafter analogously between the normal vector of P_2 and the vector $(-1,0)$).

- If the angle is smaller than the M_{CA} of the segment containing P_1, the M_{CA} is updated with the new angle.

- If the angle is greater than $\pi - M_{CA}$, everything in the segment is inverted (normal vectors, the reference vector and the triangulation) and the M_{CA} is updated.

- Otherwise continue with the next box.

Note that this procedure works only for **closed** objects. The extension of the procedure for open 3D objects is the topic of our present and future work.

4 Results

The next three figures (Figure 9, Figure 10, Figure 11) illustrate the results of the segmentation and orientation procedures. All objects consist of triangulated Gouraud-shaded segments with the same color. Connected segments were not displayed. Fandisk and Dino are closed point sets with sharp edges (Dino in the area of its feet). Although Thor is an open point set (at the bottom of his feet), we performed tests for it because of its complexity (orientation with the MST of the normal vectors based method was not successful). Because this point set is open our orientation procedure oriented a few segments on his both feet incorrectly (on his right foot marked with black half-circles, on the left foot the segments cannot be seen).

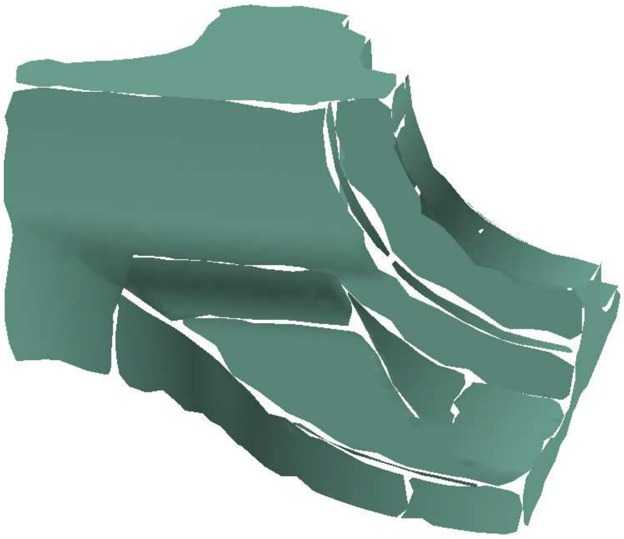

Figure 9: Consistent orientation of a data set 'Fandisk', 16,475 points

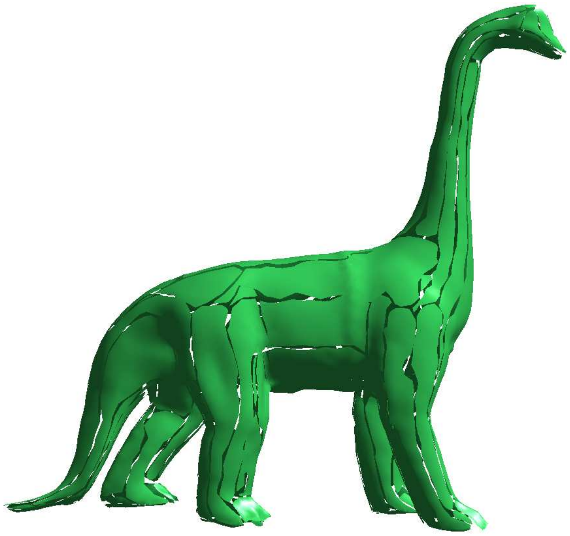

Figure 10: Consistent orientation of a data set 'Dino', 19,521 points

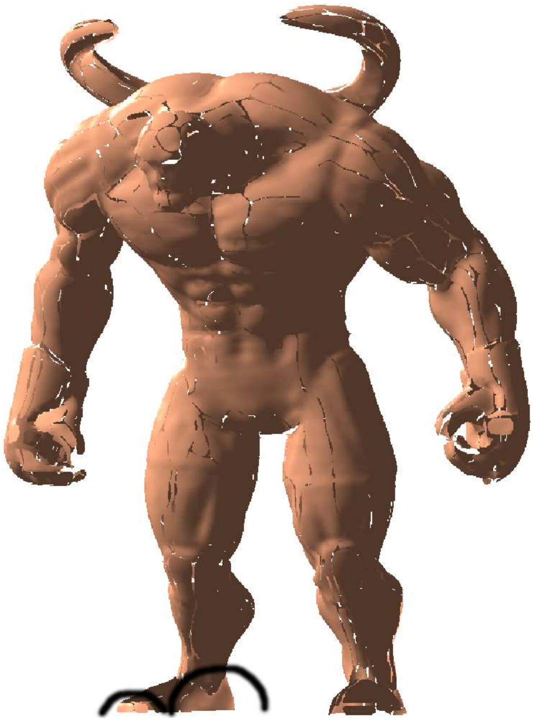

Figure 11: Consistent orientation of a data set 'Thor', 71,292 points

REFERENCES

[1] **Bajaj, Ch., Bernardini, F. and Xu, G.:** Automatic reconstruction of surfaces and scalar fields from 3D scans, *Computer Graphics Proceedings, SIGGRAPH '95, Annual Conference Series*, 1995, 109–118

[2] **Faugeras, O.D. and Herbert, M.:** The representation recognition and locating of 3D objects, *International Journal of Robotics Research 5*, 1986, 27–52

[3] **Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. and Stuetzle, W.:** Surface Reconstruction from Unorganized Points, *Proceedings SIGGRAPH '93*, 1993, 19–26

[4] **Jensen, H. W.:** Global illumination using photon maps, *Proceedings Seventh Eurographics Workshop on Rendering*, June 1996

[5] **Mencl, R., Müller, H.:** Interpolation and Approximation of Surfaces from Three-Dimensional Scattered Data Points *Research Report No. 662/1997*, December 1997

[6] **Schreiber, T., Brunnett, G. and Isselhard, F.:** Two approaches for polyhedral reconstruction of 3D objects of arbitrary genus, *International Journal of Vehicle Design, Vol. 21*, 1999, 292–302

[7] **Vančo, M., Brunnett, G. and Schreiber, Th.:** A hashing strategy for efficient k-nearest neighbors computation, *Proceeding CGI 1999*, 1999, 120–128

[8] **Vančo, M., Brunnett, B. and Schreiber, Th.:** A Direct Approach Towards Automatic Surface Segmentation of Unorganized 3D Points, *Proceedings SCCG 2000*, 2000

[9] **Várady, T., Martin, R.R. and Cox, J.:** Reverse Engineering of Geometric Models - An Introduction *Computer-Aided Design, Vol. 29*, April 1997, 255–268