Online Triangulation of Laser-scan Data

Klaus Denker, Burkhard Lehner, and Georg Umlauf

Department of Computer Science, University of Kaiserslautern, Germany.
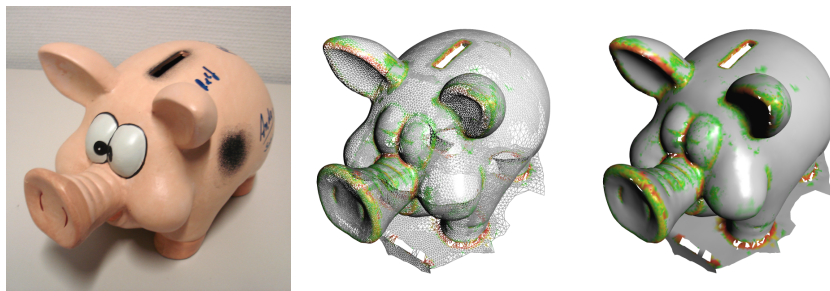`denker|lehner|umlauf@cs.uni-kl.de`

Fig. 1. A piggy bank. Original object (left), wire-frame (center), and smooth shaded triangulation with uncertainty visualization (right).

Summary. Hand-held laser scanners are used massively in industry for reverse engineering and quality measurements. In this process, it is difficult for the human operator to scan the target object completely and uniformly. Therefore, an interactive triangulation of the scanned points can assist the operator in this task.

Our method computes a triangulation of the point stream generated by the laser scanner online, i.e., the data points are added to the triangulation as they are received from the scanner. Multiple scanned areas and areas with a higher point density result in a finer mesh and a higher accuracy. On the other hand, the vertex density adapts to the estimated surface curvature. To assist the human operator the resulting triangulation is rendered with a visualization of its faithfulness. Additionally, our triangulation method allows for a level-of-detail representation to reduce the mesh complexity for fast rendering on low-cost graphics hardware.

1 Introduction

In industry, the scanning of surfaces of 3d objects is used for measurement and analysis of manufactured objects and for reverse engineering. Most scanning devices use a laser to sample points on the surface. Some scanners move
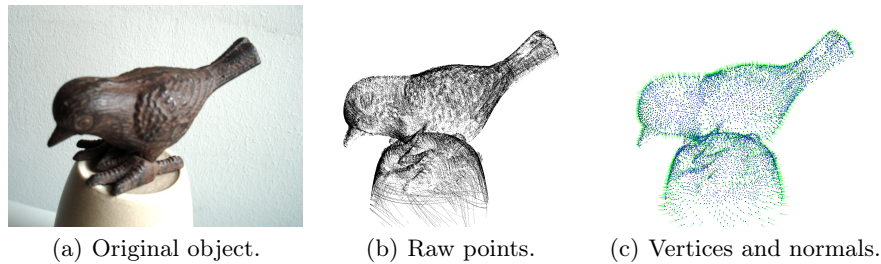
(a) Original object.   (b) Raw points.   (c) Vertices and normals.

Fig. 2. Different levels of data reduction shown with a little bronze bird.

the object while others move the laser device. While some scanning devices measure the sample points in a regular pattern, hand-held laser scanners have a movable scanning device that is moved along the surface by a human operator. These scanning devices generate a vast amount of data in very short time with very high precision. To process and triangulate this data the used method must preserve the precision while reducing the data to an adequate level, see Figures 1 and 2. Thus, it is necessary to allow for heterogeneous triangulations and point densities, especially when areas are scanned multiple times. This is particularly important for hand-held devices, where the operator most likely will scan the object from different directions and with different speeds. This generates disconnected triangulation fragments with highly different point densities.

Since the local point density can become arbitrarily high by multiple scans, the feature size that can be reconstructed is theoretically arbitrary small. Of course, the measuring accuracy/error sets a lower bound for the feature size.

For these hand-held laser scanners, the operator has to cover the complete surface of the object. Because the scanning may take a long time, it is difficult for the operator to keep track of the already scanned area. Furthermore, there is no feedback to tell the operator to re-scan a region to increase the point density to improve the quality of the reconstructed surface. Therefore, a full-automatic real time triangulation and visualization of the scanned points is crucial to assist the operator to improve the scans in less time. Thus, the crucial constraints for our triangulation task are:

A. Handling of large point sets,
B. handling of heterogeneous point densities of incoherently scanned regions,
C. handling of high precision point data with predefined measurement errors,
D. handling of point streams of arbitrary order, i.e., online triangulation,
E. triangulating full-automatically,
F. triangulating in real-time, and
G. assisting the human operator during the scanning process.

In the rest of the paper we first discuss related work in Section 2 and describe the principle of our method in Section 3. Subsequently we discuss

various aspects of our method in detail, i.e., in Sections 4–6 the neighborhood and normal calculations and local triangulation, in Section 7 our octree-based data structure, and in Section 8 local improvements of the approximation quality. In Section 9 we present a method to assist the human operator during the scanning process and in Section 10 the integrated level-of-detail representation. Finally we show results of our method in Section 11 and close with an outlook on our future research plans in Section 12.

2 Related work

To contrast our approach to other methods for surface reconstruction from unorganized point clouds, we briefly describe alternative methods and discuss their pros and cons with respect to the constraints A.–G. given above.

One of the first methods in this field was proposed in [14]. Here, for every point a surface normal is estimated from its k nearest neighbors and its orientation is propagated from the orientation of one particular normal to all other normals using a global minimal spanning tree of the points. This allows to estimate tangent planes defining an estimated signed distance function to the surface. Its zero-set is used to compute a triangulation of the surface using marching cubes. This method can deal with surfaces with boundaries and holes, and no additional information (such as surface normals) is necessary. On the other hand, it is not capable of dealing with incremental insertion of data points, since the orientation propagation is global, and the density of the points on the surface is pre-defined, otherwise spurious holes are introduced. The minimum feature size that can be reconstructed is fixed a priori by the edge length of the marching cubes algorithm and increasing the density of points does not reveal more details. Therefore, at least constraints B. and D. are not satisfied.

Alpha shapes were defined in [9, 10]. For a given real number $\alpha > 0$, the alpha shape S_α of a point set P is the set of all k-simplices $T \subset P$ ($k < d$) with vertices lying on a sphere with radius α that does not contain any other point of P. The alpha shape can efficiently be determined from the Delaunay triangulation where α controls how many "details" of the point cloud are "cut" out of the convex hull of P. If α is too large, details remain hidden under larger faces, if it is too small, the object may be cut into disconnected pieces. Therefore, the choice of α is crucial for the optimal reconstruction of a surface from a point cloud. If the variation of the point density is too high, there may not even exist a suitable α value, violating constraint B.

For the so-called *weighted alpha shapes* of [1] every point of P gets an associated weight. This permits using different values of α for different regions of P. The weights have to be tuned to the point density very accurately to achieve a good reconstruction of the underlying surface, making it difficult to achieve B. Another extension was proposed in [20]. The sphere with radius α is deformed anisotropically into an ellipsoid, achieving a more accurate

separation of surfaces close to each other. But their approach relies on user input, violating constraint E.

All methods based on alpha shapes can be used to reconstruct surfaces with borders and holes, an no additional information per vertex other than its position is necessary. But, the correct choice of α or the point weights, respectively, is crucial for the quality of the final triangulation, which is in contrast to constraint C. Furthermore, the computed triangulation is not guaranteed to be a 2-manifold with border. It can contain edges with more than two adjacent faces, or isolated edges and vertices. Therefore, a postprocessing clean-up step is necessary, violating also constraint F.

The so-called *power crust* of [4] is based on an approximation of the medial axis transform of the point set. It is computed from the Voronoi diagram and the poles of the input points. From this, it calculates in an inverse transformation the original surface using the power diagram of the poles and taking the simplices dividing the interior and exterior cells of the power diagram from each other as triangulation for the surface. The power crust approach produces connected surfaces possibly with intentional holes. So, it is not suitable for online triangulations (constraint D.) where the triangulation may consist of disconnected fragments (constraint B.).

The *eigencrust* method proposed in [16] is specialized to produce high quality surface reconstructions on noisy point clouds. It labels all tetrahedra in the 3d Delaunay triangulation of the sample points as either being inside or outside the surface based on a global optimization. The triangulation of the reconstructed surface is the set of faces that are adjacent to one inside and one outside tetrahedron. Because of the global optimization step, the results are of high quality even with the presence of noise and outliers. The final surface is always a 2-manifold without border. Therefore, the eigencrust method cannot be used with constraint D.

The geometric convection approach for surface reconstruction described in [6] starts with a Delaunay triangulation of the point set, and shrinks the boundary surface by removing tetrahedra containing a boundary triangle that does not fulfill the *oriented Gabriel property*, i.e., the half-sphere centered at the triangle's circumcenter and oriented to the inside contains point of the point set. This procedure is repeated until all boundary triangles fulfill the oriented Gabriel property. In some cases, cavities are not opened by this algorithm, so another property has to be defined to remove the involved tetrahedra. The approach requires the Delaunay triangulation of the complete point set, therefore it contradicts constraint D.

An extension of [6] for streams of point sets is proposed in [3]. The point set is divided into slices, and only a limited number of slices is kept in memory. A slice that cannot have impact on the current slice can be removed from memory, storing the triangles found for that slice. For the division into slices, all points have to be known in advance, because they have to be ordered according to one of the spatial coordinates. Therefore, this method is not suitable for constraint D.

A method suitable not only for surface reconstruction but also for re-meshing of an existing mesh is presented in [18]. Using an advancing front approach, triangles are constructed that fulfill two user-defined constraints: the maximum edge length with respect to the curvature, and ratio bounds of adjacent edges. For surface reconstruction, a projection operator $\mathcal{P}$ and a guidance field g have to be defined, contradicting constraints B., D., and F.

A common problem of the methods $[1, 3, 4, 6, 10, 16, 20]$ is their computational complexity, which is too high for real-time applications (constraint F.). Another disadvantage of these methods is the fact that the sample points or the same number of points are used to create the surface mesh. Thus, the complexity of the meshes increases rapidly while scanning, and the measurement errors are not corrected. Furthermore, if a region is scanned multiple times, the additional vertices decrease the area of the mesh faces, but the noise remains constant, leading to a bumpier surface after every scan pass. These aspects are in contrast to constraints A. and C.

In [5] an interactive online triangulation method is proposed. The sampled points are processed in a pipeline. In the first stage the number of points is reduced by dropping every point that is closer than a specified radius from an already existing point. In the second stage, the normal at the point is estimated using the points in a local neighborhood. After another reduction stage with a larger radius, the points with a stable normal are inserted into the surface mesh which is re-triangulated locally with a shortest edge criterion to decide which edges to keep. This approach works well and is fast, but has some major drawbacks:

- A large fraction of the input is ignored and not used to reduce the noise of the input data, violating C.
- The size of the smallest features that can be modeled is fix, violating C.
- The size of the mesh triangles is not adapted to the density of sample points or the curvature of the surface, violating B. and E.

The method proposed in this paper uses some of the ideas of the approach of [5], but satisfies all constraints A.-G.

3 Online triangulation

Our method is based on a laser scanner like the FARO Laser ScanArm [11] as described in Section 11. Scanners of this type generate a stream $D = (d_1, d_2, \dots)$ of *data points* d_i. Each data point is a pair $d_i = (p_i, h_i) \in \mathbb{R}^3 \times \mathbb{R}^3$ of a *raw point* p_i, that is measured by the scanner on the scanned object, and the *scan position* h_i of the laser scanner at the moment of scanning p_i.

A laser scanner of this kind scans an object line by line measuring a certain number of data points per scan line. The scanner we used scans up to 30 lines per second measuring up to 640 data points per scan line, see e.g. Figure 2(b). These scan lines are arranged in scan passes that the human operator triggers

by pressing a button. The pauses between two scan passes are usually used by the operator to reposition the scanner for a different scan direction.

In order to triangulate this huge data stream online, the data points need to be reduced. For this the data points are classified by their distance and added to so-called *neighborhood balls* b_j that represent a subset of data points within a certain radius and similar scan positions. The radius depends on the point density and estimated curvature. Subsequently only the averages of data points of the neighborhood balls are used as vertex positions in the triangulation T approximating $\{p_1, p_2, \dots\}$. So, the overall process is described schematically as follows (for the used data structure see Section 7)

ONLINE-TRIANGULATION$(d_1, d_2, \dots)$

Input: Data point stream $D = (d_1, d_2, \dots)$;
Output: Triangulation T approximating $\{p_1, p_2, \dots\}$.
 1: **while** (D not terminated) **do** {
 2:     ADD-TO-NEIGHBORHOOD-BALLS(d_i);   \\ see Section 4
 3:     Update normals of affected neighborhood balls;   \\ see Section 5
 4:     Update local approximation;   \\ see Section 8
 5:     Triangulate area of affected neighborhood balls;   \\ see Section 6
 6:     Render triangulation with uncertainty visualization; \\ see Section 9
 7: }

---

4 Neighborhood balls

A bounding cube of edge length R enclosing the maximal scanning range is

$$O = [x_{\min}, x_{\min} + R] \times [y_{\min}, y_{\min} + R] \times [z_{\min}, z_{\min} + R],$$

i.e., $p_i \in O$ for all i. Furthermore, a *ball* with center $c \in \mathbb{R}^3$ and radius $r \in \mathbb{R}, r \geq 0$, is for the Euclidian norm $\| \cdot \|$ defined as the set

$$\beta(c, r) = \{x \in \mathbb{R}^3 : \|x - c\| \leq r\}.$$

As in [5] we use *neighborhood balls* $b_j = (c_j, r_j, D_j)$ to represent a set of n_j data points $D_j = \{d_{j,1}, \dots, d_{j,n_j}\} \subset D$ contained in the ball $\beta_j = \beta(c_j, r_j)$. Every neighborhood ball corresponds to a local estimate N_j for the oriented surface normal, which might be undefined (cf. Section 5), and a vertex v_j of T. Thus, neighborhood balls can intersect and serve three purposes:

- Collecting n_j data points to reduce the number of visualized data points.
- Estimating a local oriented surface normal.
- Averaging its data points gives the position of a vertex of the triangulation.

The minimal ball radius $R_{\min} = 0.75\,\text{mm}$ prevents ball sizes below scanner accuracy. A neighborhood ball may contain up to $n_{\text{split}} = 40$ data points. Later this value is depending on curvature, see Section 8.2. The set of all neighborhood balls b_j is denoted by B. It is initialized with the first data point $B = \{(p_1, R, \{d_1\})\}$. Then new neighborhood balls are generated by adding one data point after the other with ADD-TO-NEIGHBORHOOD-BALLS(d_i), using the following steps:

1. To add data point $d_i = (p_i, h_i)$, first all k neighborhood balls $b_j = (c_j, r_j, D_j)$ are determined that contain $p_i \in \beta_j$ with normals N_j aligned to the scanning direction, i.e.

$$N_j^T \cdot (h_i - p_i) \geq 0 \qquad \text{(if } N_j \text{ is defined)}. \qquad (1)$$

   a) If $k = 1$, d_i is added to D_j.
   b) If $k > 1$, d_i is added to D_j of the neighborhood ball b_j with largest radius and smallest distance $\|c_j - p_i\|$.
   c) If $k = 0$, a new neighborhood ball $b = (p, r, \{d\})$ is generated, with radius $r = 2^{-\mu}R$ where μ is the smallest integer such that $\beta(p, r)$ does not contain the center of any other b_j. The new ball b is added to B.
2. If in cases a) and b) n_j equals n_{split} after d_i is added and $r > R_{\min}$, the neighborhood ball b_j is removed from B and all data points in D_j are added using Step 1. If in this process a data point $d_l \in D_j$ is not contained in any other neighborhood ball of $B \setminus \{b_j\}$, a new ball $b = (p_l, r_j/2, \{d_l\})$ is generated and added to B.

Remark 1. This definition of neighborhood balls has two advantages over the method of [5]: First all data points are collected in neighborhood balls and not only the first one to support C. (see Sections 5 and 8.1), and second the radii of the balls can be adapted to the density of the data points and the estimated curvature of the surface to support B. and E. (see Section 8.2).

For later triangulation we use the *average* of a neighborhood ball b_j

$$\bar{b}_j = \frac{1}{n_j} \sum_{l=1}^{n_j} p_{j,l},$$

as position of vertex v_j representing b_j in the triangulation, see Figure 2(c).

5 Normal estimation

For every neighborhood ball b_j an estimated surface normal N_j is calculated. First all n_l data points d_l contained in $\beta(\bar{b}_j, 2r_j) \ni p_l$ are determined. This provides a more stable normal estimation than using only the data points in D_j. These data points are used for a principal component analysis as in [5,15]. Computing the eigenvalues $0 \leq e_1 \leq e_2 \leq e_3$ of the 3×3 covariance matrix

$$C = \sum_l (p_l - \bar{b}_j)(p_l - \bar{b}_j)^T$$

using [19], yields for N_j the direction of the eigenvector v_{e_1} of C corresponding to the smallest eigenvalue e_1

$$N_j = v_{e_1}/\|v_{e_1}\|.$$

To get a stable normal estimate it is necessary that $e_2 \geq 2e_1$. Otherwise b_j does not have a normal estimate and all subsequent computations requiring a N_j are rejected. Thus, highly curved regions with too low point density are either not triangulated or marked as regions that need a further scan pass, see Section 9. This ensures a locally rather planar point distribution. To get the orientation of N_j, the average scan direction of all determined d_l

$$\bar{s}_j = \frac{1}{n_l} \sum_l (h_l - p_l).$$

is used. The normal orientation is correct if $N_j^T \cdot \bar{s}_j \geq 0$. Otherwise the orientation is inverted. Finally, all data points of D_j that do not satisfy (1) are removed from D_j and re-inserted using ADD-TO-NEIGHBORHOOD-BALLS. Figure 2(c) shows the estimated normal of each neighborhood ball.

6 Local triangulation

Because every neighborhood ball corresponds to one vertex in the triangulation, the latter is updated in five steps if a neighborhood ball is added or removed from B:

1. Collect potential neighbor vertices in a set V_j.   (see Section 6.1)
2. Project V_j onto the estimated tangent plane.   (see Section 6.2)
3. Adapt the border of V_j.   (see Section 6.3)
4. Determine the triangulation T_j of V_j.   (see Section 6.4)
5. Insert T_j into the triangulation T.   (see Section 6.5)

6.1 Collecting potential neighbor vertices

Every neighborhood ball b_j corresponds to a vertex v_j in T with position $\bar{b}_j$. Thus, if b_j is added or removed from B, the corresponding vertex v_j is added or removed from T. In both cases the local neighborhood of v_j needs to be re-triangulated. To determine the geometric neighbors of v_j we define a ball

$$\eta_j(r) = \{x \in \mathbb{R}^3 : \|(x - \bar{b}_j) + ((x - \bar{b}_j)^T N_j)(f_\eta - 1)N_j\| \leq r\}.$$

of radius r around $\bar{b}_j$ flattened along the normal N_j by f_η to provide a better separation of close parallel surfaces sheets. Then, the geometric neighbors are all b_l with $\bar{b}_l \in \eta_j(5r_j)$ for $f_\eta = 3$ and $N_l \cdot N_j \geq 0.5$. This yields a set $V_j = \{v_{j_1}, \ldots, v_{j_m}\} \subset V \cup \{v_j\}$ of vertices that will be re-meshed.

6.2 Projection onto the estimated tangent plane

Because the triangulation of the area around b_j is computed in the plane perpendicular to N_j all vertices $v_{j_i} \in V_j$ are projected along N_j onto this plane, i.e. $\bar{b}_{j_i}$ is projected to t_{j_i}. Then, because of the one-to-one correspondence of v_{j_i} to t_{j_i}, triangulating the t_{j_i} is equivalent to triangulating the v_{j_i}. Therefore, we will speak of a triangulation of V_j although the triangulation is computed in the local estimated tangent plane.

6.3 Adapting the border of the local triangulation

Triangulating V_j yields a triangulation T_j of the local neighborhood of v_j. Because most vertices in T_j are also in T, the edges in T_j should match edges in T. Thus, the *border* ∂T_j of T_j has to match edges in T. The border $\partial T_j = (v_{x_1}, \ldots, v_{x_{n_x}})$ is represented as a counter-clockwise oriented, ordered sequence of n_x *border vertices* $v_{x_l} \in V_j$, $l = 1, \ldots, n_x - 1$ with $v_1 = v_{x_{n_x}}$, i.e., the index of border vertices is understood modulo n_x. Every pair $(v_{x_l}, v_{x_{l+1}})$ is a so-called *border edge* and the border ∂T_j is initialized as the convex hull of V_j using "Jarvis' March" [7]. Subsequently V_j and ∂T_j are modified until the border edges match edges in T as good as possible.

If a border edge $e_b = (v_{x_k}, v_{x_{k+1}})$ does not match any edge in T, determine the edge $e_s = (v_{x_k}, v_s)$ or $e_{\bar{s}} = (v_s, v_{x_{k+1}}) \in V_j \times V_j$ in T inside ∂T_j with smallest angle φ to e_b. Then v_s is inserted to ∂T_j between v_{x_k} and $v_{x_{k+1}}$, if e_s respectively $e_{\bar{s}}$ is either an inner edge or $\varphi < 20°$ and if this does not cause proper intersections of the interior of two edges of ∂T_j or any loops containing more than two border edges, see Figure 3. This approach can lead to a border of the form $(\ldots, v_{x_l}, v_{x_{l+1}}, v_{x_{l+2}}, \ldots)$ with $v_{x_l} = v_{x_{l+2}}$, see Figure 3, which is handled by removing v_{x_l} and $v_{x_{l+1}}$ from ∂T_j and $v_{x_{l+1}}$ from V_j.
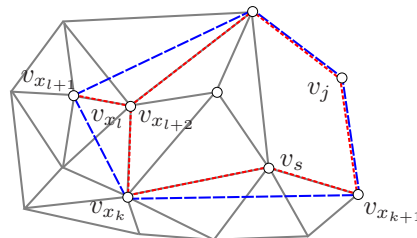

Fig. 3. The border ∂T_j before (blue) and after (red) the modification.

Repeating these operations until there are no more edges that can be removed results in a border that fits the existing triangulation T better. This process terminates because the border shrinks monotonically in each step.

6.4 Triangulation of the border

To triangulate V_j first ∂T_j is split into monotone sub-polygons which are triangulated individually, see e.g. [7]. This determines a triangulation T'_j of

∂T_j. Second all vertices of $v_l \in V_j \setminus \partial T_j$ are added to T'_j successively by splitting the triangle of T'_j that contains t_l at t_l into three new triangles. This is repeated for every vertex of $V_j \setminus \partial T_j$ yielding a triangulation T''_j. Finally, the Delaunay criterion [13] is applied repeatedly constrained by ∂T_j to improve the triangle quality generating a triangulation T_j of V_j.

6.5 Insertion of the local triangulation

Before T_j can be inserted into T, the triangles of T in conflict with T_j must be removed. We first remove all triangles of T incident to a vertex of $V_j \setminus \partial T_j$.

Remark 2. Note that this also removes triangles from vertices in $V_j \setminus \partial T_j$ to vertices in $V \setminus V_j$ outside of $\eta_j(5r_j)$. Thus, the global topology of the surface is corrected due to the increased local point density.
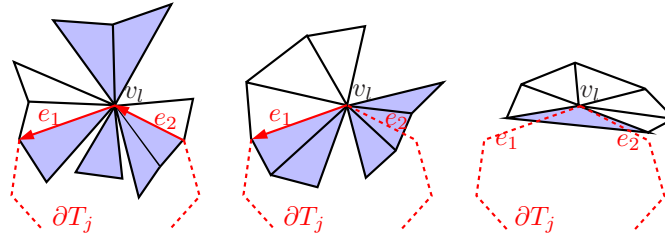

Fig. 4. Topologically preparing the border ∂T_j (red) for the new triangulation T_j by deleting the blue faces. All triangles shown in the figures are in T, T_j is not shown.

At this stage T contains no triangles connected to vertices of $V_j \setminus \partial T_j$. All triangles remaining in T conflicting with T_j involve only vertices on ∂T_j. For such a vertex $v_l \in \partial T_j$ with border edges $e_1 = (v_{l-1}, v_l)$ and $e_2 = (v_l, v_{l+1})$, all faces potentially pointing to the inside of the border are deleted. Thus, there are three different cases that are solved topologically:

1. If both edges e_1 and e_2 belong to T, all faces of the one-ring of v_l are removed if they are left of e_1 or e_2 or if they are not connected to the right faces of e_1 or e_2. This removes triangulated areas of T that are also covered by T_j and reduces the complexity of the one-ring, see Figure 4 (left).
2. If only e_1 belongs to T, the face of the one-ring of v_l left of e_1 and all faces not connected to the right face of e_1 are deleted, see Figure 4 (middle).
3. If both e_1 and e_2 do not belong to T and v_l has a closed one-ring, the face pointing the most inside the triangulated area is removed, which is e.g. the triangle of the one-ring of v_l intersected by the bisector of e_1 and e_2 in the local estimated tangent plane, see Figure 4 (right).

Finally, if there are two vertices v_{l_1} and v_{l_2} from ∂T_j that are connected by an edge e that does not belong to T_j and lies inside of the polygon

spanned by ∂T_j, the corresponding triangles are removed. To test if such an edge is inside the polygon spanned by ∂T_j the angle β, the inner angle of the polygon at v_{l_1}, must be larger than the angle α between the incoming border edge at v_{l_1} and e, see Figure 5.
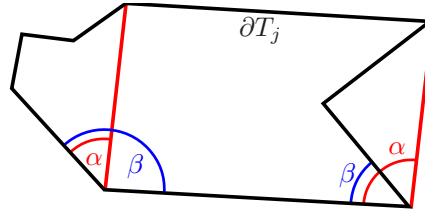
Remark 3. This also changes the global topology of the surface.

Fig. 5. Border ∂T_j (black) with edges (red) not connected by faces to the border.

7 The octree

To support the geometric neighborhood searches in Sections 5 and 6.1 efficiently we use an octree data structure to manage the neighborhood balls. The root node of the octree represents the cube O. Every node in the tree represents a sub-cube

$$o = [x, x+\delta] \times [y, y+\delta] \times [z, z+\delta]$$

of O and has no or exactly eight child-nodes holding the eight sub-cubes $o_1, \ldots, o_8$ with side length $\delta/2$, see Figure 6. To accelerate searches in the local neighborhood, every node stores

Fig. 6. The cube o of an octree node and its child-node's sub-cubes $o_1, \ldots, o_8$.

additional links to the 26 face-, edge- and corner-neighbor nodes $o_{n_1}, \ldots, o_{n_{26}}$ on the same level, as in [5].

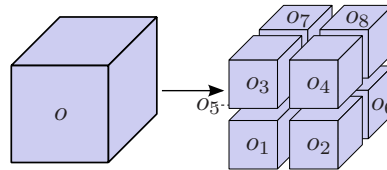*Remark 4.* The main advantage of using an octree instead of a grid as in [5] is the use of different levels of detail corresponding to the levels of the octree to create a finer triangulation in regions with higher point density.

Every neighborhood ball b_i belongs to one cube in the octree which contains its center point c_i. The radius r_i equals the edge length of the cube. Every cube in the octree has a list of the neighborhood balls it contains.

Every data point $d_j = (p_j, h_j)$ is inserted into the octree by searching for the neighborhood ball b_i containing the raw point $p_j \in \beta_i$ and, if b_i is found, inserting it to that ball as in Section 4. To search for b_i the octree is descended from the root node O traversing on any level of the octree the cube o containing p_j. For the traversal all 26 neighbor cubes are tested to find the ball with c_i closest to p_j. If no b_i is found, the search descends one level in the tree and repeats the neighbor traversal. This is repeated until a ball is found or the leaf nodes are searched unsuccessfully. In the latter case a new

neighborhood ball b containing d_j is created in leaf node cube containing p_j, see Section 4.

To make b as large as possible the highest level in the octree with sufficient space is determined, such that the ball β_i does not contain the center of any other ball of B. These centers can only be in the siblings of the 26 indirect neighbor cubes, which share at least one corner with the actual cube o. The set of these cubes is denoted by $S(o)$. In $S(o)$ the center c_k that is closest to p_j is determined with $\Delta := \|c_k - p_j\|$.

1. If Δ is smaller than the radius of b, i.e. the edge length ℓ of o, the cube o is split into sub-cubes until the radius of b is small than Δ. Thus, b is added to a sub-cube o' that is $\lceil \log_2(\ell/\Delta) \rceil$ levels below the node of o in the octree.
2. If Δ is larger than the radius of b, it can be added to the cube o or one of its ancestors. Thus, the ancestors o' of o are tested if their siblings of $S(o')$ contain a center too close to p_j. Finally, b is added to the highest ancestor above o for which this test is negative.

8 Improving the Approximation

For every neighborhood ball b_i a least square fit f_i for its raw points is computed to smooth the triangulation and to reduce noise in the raw points. The fit f_i is a cubic approximation of the raw points of b_i parametrized over the estimated tangent plane of b_i as in [2] computed by a singular value decomposition. It serves two purposes:

- correction of vertex positions and
- curvature dependent ball sizes.

8.1 Correction of mesh points

The position of a vertex v corresponding to a neighborhood ball b_i is approximated by the arithmetic mean of all its raw points $\bar{b}_i$. On curved surfaces this results in a displaced position. In local coordinates of the estimated tangent plane $\bar{b}_i$ has coordinates $(0,0,0)^T$. Thus, the point $\widetilde{b}_i$ with local coordinates $(0, 0, f_i(0,0))^T$ is a better approximation of the raw points. In order to guarantee that the vertices of T are within scanner precision the raw point p_j closest to $\widetilde{b}_i$ is determined. If ε is the scanner precision and $\widetilde{b}_i$ is not contained in $\beta(p_j, \varepsilon)$, $\widetilde{b}_i$ is projected onto $\beta(p_j, \varepsilon)$ in direction $p_j - \widetilde{b}_i$. This new point is the position of the corresponding vertex v_i.

Remark 5. Because the laser scanner has different precisions in different directions, i.e. along the scan line, between scan lines, and in laser beam direction, $\widetilde{b}_i$ is projected onto an ellipsoid around p_j .

8.2 Curvature dependent ball size

The fits f_i are also used to estimate the curvature of T in the vertex v_i. Then the size of the neighborhood balls is controlled by the curvature measure $C_i := (|\kappa_1| + |\kappa_2|)/2$ at v_i, where κ_1 and κ_2 are the principle curvatures of f_i at $\widetilde{b}_i$ before the projection onto $\beta(p_j, \varepsilon)$. A small value of C_i indicates that the region is rather flat. So, for each neighborhood ball with a valid normal the curvature C_i is computed and the neighborhood ball is split if

$$\arctan\left(4r_j C_j\right) \cdot 2n_j/\pi \geq n_{\text{split}}.$$

This results in larger triangles in flat regions.

9 Uncertainty visualization

Regions with a high uncertainty in the triangulation should be highlighted, to enable the operator of the laser scanner to increase the point density by multiple scan passes. To measure the uncertainty, the stability of the normal estimation is used. It can be calculated for each neighborhood ball b_i by the two smallest eigenvalues e_1 and e_2 of the principal component analysis in Section 5. The uncertainty u_i is defined as

$$u_i = \arctan\left(\left(e_2/e_1 - 2\right)/20\right) \cdot 2/\pi.$$

It is restricted to $[0, 1]$ and visualized by coloring the vertices using a transition from red ($u_i = 0$), yellow ($u_i = 0.25$), green ($u_i = 0.5$) to white ($u_i \geq 0.75$).

10 Level of Detail

For very complex or large objects it may be necessary to use a further reduced mesh for fast rendering on low cost graphics hardware. The octree contains different levels, each containing neighborhood balls b_j of a certain size. These levels can be used for a reduced level of detail on the mesh.

To make use of the levels in the octree, the data points d_i are also added to so-called *LOD balls* b_k^{LOD} on the levels above the enclosing neighborhood ball. These LOD balls b_k^{LOD} are modified neighborhood balls containing all data points d_i of the neighborhood balls b_i on the levels below. Each data point d_i belongs to one neighborhood ball $b_j = (c_j, 2^{-l}R, D_j)$ on the level l and one LOD ball $b_k^{\text{LOD}} = (c_k, 2^{-\lambda}R, D_k)$ for all $0 \leq \lambda < l$ in each level above.

The *global level of detail* l^{LOD} is the depth of the deepest level in the octree used to determine T. If l^{LOD} changes, the complete mesh has to be *re-triangulated*. First all faces of the mesh are deleted. Then the neighborhood

balls or LOD balls necessary for re-triangulation are selected according to their depth in the octree. An ordinary neighborhood ball is used if its depth is less or equal to l^{LOD}. If the depth of a neighborhood ball is larger than l^{LOD} its corresponding LOD ball on level l^{LOD} is used. With these balls the triangulation is computed as described above. Figure 7 shows a schematic illustration of neighborhood balls (white) and LOD balls (gray) used for re-triangulation on different levels of detail.
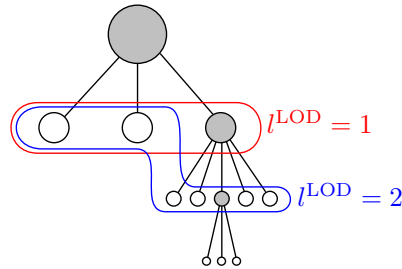
Fig. 7. Schematic illustration of the level of detail re-triangulation in the octree.

11 Results

We used a hand-held laser scanner "Laser ScanArm" from Faro [11] (Figure 8). That is a measurement arm with seven joints and an assembled laser scanner "Laser Line Probe". The scanner driver provides 3d point data relative to the foot of the measurement arm. Lines of up to 640 points can be scanned up to 30 times per second. For each of these lines the position and the viewing direction of the laser scanner is tracked.

Fig. 8. The measuring arm "FaroArm" with laser scanner "Laser Line Probe" [11].

In Figure 1 a scanned piggy bank is shown, while Figure 11 shows the result of scanning the bronze bust "Bildnis Theodor Heuss" by Gerhard Marcks. The uncertainty visualization in the wire-frame and smooth shaded representations reveal the regions that can be improved by additional scan passes.

The reduction of the input points is demonstrated in Figure 2, where Figure 2(b) shows the data generated by the scanner, and Figure 2(c) shows the averages $\bar{b}_j$ of the neighborhood balls, together with the estimated normal for every ball.

Figure 9 shows how the triangulation is adaptively refined for several scan passes of the same region. The edge of the protruded digit becomes more precise after each scan pass. Furthermore, the wire-frame representation (bottom row) shows that the triangles near the protruded edge are smaller, because the neighborhood balls are dissolved earlier in this region of higher curvature.

Another example of increasing accuracy for multiple scan passes is shown in Figure 10. The reconstructed surface of a piece of paper with holes of
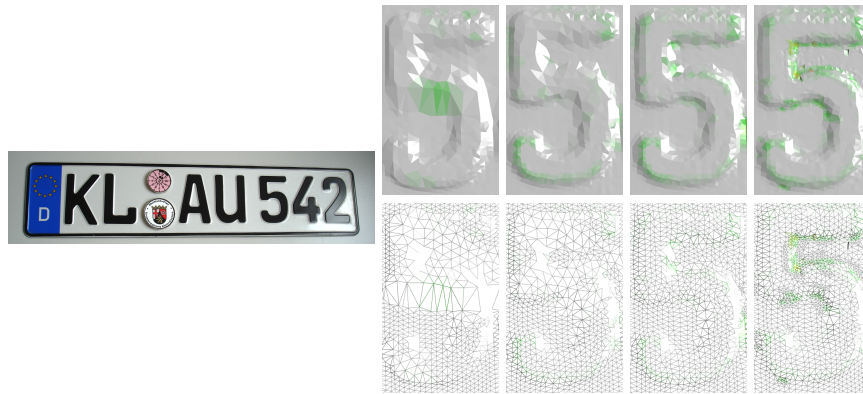
Fig. 9. Flat shading (top) and wire-frame (bottom) of triangulation of a license plate after 250, 2000, 5000, 9000 scan lines with uncertainty visualization.
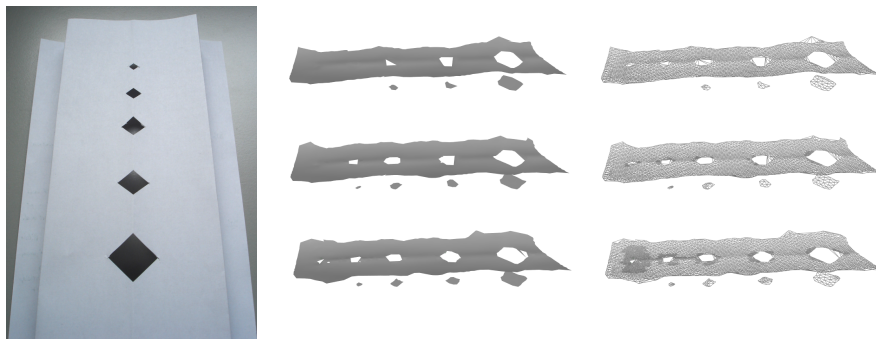

Fig. 10. Smooth shading and wire-frame representation of triangulation of a sheet of paper with holes after 585, 1895, and 3149 scan lines.

different size is shown after the first, the second and the third scan pass. While after the first scan pass two small holes are still closed by the triangulation procedure, after the third scan pass all holes are correctly detected.

Four levels of detail of a scan of a mug are shown in Figure 12. The number of triangles is 20,758 for the finest level of detail (left), 17,678 for the next coarser level, and 11,277 and 3,584 triangles for the two subsequent levels. Choosing a coarser level of detail reduces the complexity of the triangulation, but can result in more reconstruction errors, especially at sharp edges.

All examples are computed on an Intel Core2 Quad Q6600, 2.4 GHz computer with 4 GB RAM. To demonstrate the efficiency of the proposed method we list in Table 1 the sizes of the models on the fines level in terms of data points d_i and vertices v_i and the times spend for the different computations: number of data points processed per second, number of vertices processed per second, and the overall times for the computation of the triangulation
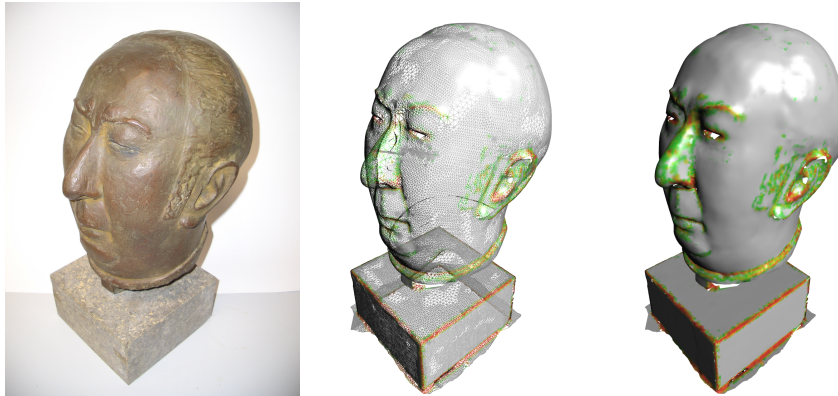
Fig. 11. Gerhard Marcks - Bildnis Theodor Heuss, [17]. Original object (left), wire-frame (center), smooth shaded triangulation with uncertainty visualization (right).
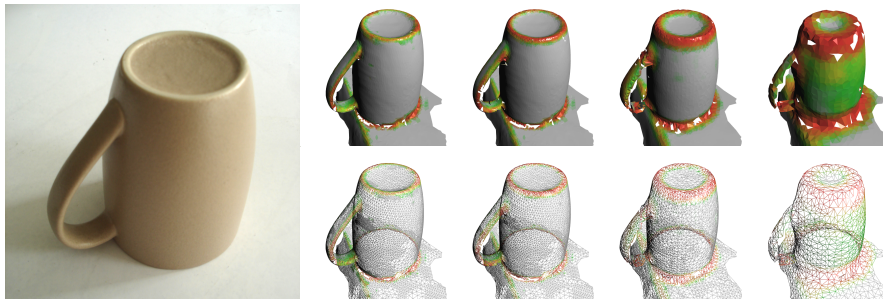

Fig. 12. Flat shaded mug with different levels of detail (top row) and wire-frame (bottom row).

and the scanning process. The times for the scanning process do not include the pauses between scan passes. It is apparent from Table 1 that our method works in real time even for complex objects. A video of a live scanning process is available at [8].

12 Conclusion and Outlook

Our experiments show that the proposed method is suitable to assist the operator of a hand-held laser scanner to produce fast and complete high quality triangulations satisfying all constraints A.–G. of the Introduction. The online visualization helps to reduce the time for the scanning process significantly. The final surface mesh is a correct triangulation that can be used without further postprocessing for measurement, surface analysis or reverse engineering.

Table 1. Table of number of data points d_i and vertices v_i in the final triangulation, times for processing data points and vertices, overall time for the computation of the triangulation and the scanning process for all models presented in this paper.

	Fig.	Data points	Vertices	Data points per second	Vertices per sec.	CPU time [sec]	Scan time [sec]
Piggy bank	1	1280387	30130	4198	98.9	304.6	318
Bronze bird	2	181731	6247	4432	152.0	41.1	143
License plate	9	1866413	29703	4392	69.9	424.8	504
Sheet of paper	10	478900	2538	7483	39.9	63.6	105
Theodor Heuss	11	2451014	35412	4300	62.1	570.0	623
Mug	12	358232	8866	5778	143.7	61.7	105

The robustness of our method is derived from the fact that the human operator can increase the point density by additional scan passes in regions that are not yet reconstructed topologically correct, or where important features are still missing.

Nevertheless, there are some requirements for the proposed method to work satisfactorily. Only those parts of the object that are covered by scan lines can be reconstructed. Because of the interactive rendering the human operator can easily detect uncovered regions, fill the remaining holes and scan a topologically correct reconstruction of the target object. Regions of the object that cannot be scanned, because of occlusions or limitations of the scan arm, cannot be reconstructed and remain as holes in the triangulation.

Some extensions to improve the usability are worth further investigation. The detection of sharp features and surface borders could improve the final mesh. The method of [12] can be used to detect sharp features within the scan and move the vertices onto these features.

Experiments show that the orientation of the scanning device during the scan has an impact on the quality of the input data. Scanning the same region with different orientations improves the stability of the computation significantly. Additionally to the uncertainty visualization the optimal orientation for the next scan of an uncertain region could be visualized to aid the operator to achieve better and faster results.

Acknowledgments

References

1. N. Akkiraju, H. Edelsbrunner, M. Facello, P. Fu, E.P. Mücke, and C. Varela. Alpha shapes: Definition and software. In *1st Int. Computational Geometry Software Workshop*, pages 63–66, 1995.
2. M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C.T. Silva. Computing and rendering point set surfaces. *IEEE Trans. Vis. Comput. Graphics*, 9(1):3–15, 2003.
3. R. Allègre, R. Chaine, and S. Akkouche. A streaming algorithm for surface reconstruction. In *Symp. Geom. Proc.*, pages 79–88, 2007.
4. N. Amenta, S. Choi, and R.K. Kolluri. The power crust. In *6th ACM Symp. on Solid Modeling and Applications*, pages 249–266, 2001.
5. T. Bodenmüller and G. Hirzinger. Online surface reconstruction from unorganized 3d-points for the DLR hand-guided scanner system. In *2nd Symp. on 3D Data Processing, Visualization and Transmission*, pages 285–292, 2004.
6. R. Chaine. A geometric convection approach of 3-d reconstruction. In *Symp. Geom. Proc.*, pages 218–229, 2003.
7. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry*. Springer-Verlag, Heidelberg, 2000.
8. K. Denker, B. Lehner, and G. Umlauf. Live scanning video. `www-umlauf.informatik.uni-kl.de/~lehner/scanning.avi`, 2008.
9. H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Trans. Inf. Theory*, 29(4):551–559, 1983.
10. H. Edelsbrunner and E.P. Mücke. Three-dimensional alpha shapes. *ACM Trans. Graph.*, 13(1):43–72, 1994.
11. Faro Europe GmbH & Co. KG. `http://www.faro.com`, 2008.
12. S. Fleishman, D. Cohen-Or, and C.T. Silva. Robust moving least-squares fitting with sharp features. *ACM Trans. Graph.*, 24(3):544–552, 2005.
13. Ø. Hjelle and M. Dæhlen. *Triangulations and Applications (Mathematics and Visualization)*. Springer-Verlag, Heidelberg, 2006.
14. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Comput. Graph.*, 26(2):71–78, 1992.
15. I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, Heidelberg, 2002.
16. R. Kolluri, J. Richard Shewchuk, and J.F. O'Brien. Spectral surface reconstruction from noisy point clouds. In *Symp. Geom. Proc.*, pages 11–21, 2004.
17. G. Marcks. Bildnis Theodor Heuss. In *"Pfalzgalerie Kaiserslautern", permanent loan from "Vereinigung Pfälzer Kunstfreunde (VKP)"*. 1952. Bronze sculpture, 31.5 x 21 x 23.5 cm.
18. J. Schneider, C. E. Scheidegger, S. Fleishman, and C. T. Silva. Direct (re)meshing for efficient surface processing. *Comp. Graph. Forum*, 25(3):527–536, 2006.
19. O.K. Smith. Eigenvalues of a symmetric 3×3 matrix. *Comm. ACM*, 4:168, 1961.
20. M. Teichmann and M. Capps. Surface reconstruction with anisotropic density-scaled alpha shapes. In *IEEE Conf. Vis.*, pages 67–72, 1998.