Available online at www.sciencedirect.com

ScienceDirect

ISPRS Journal of Photogrammetry & Remote Sensing 63 (2008) 115–127

PHOTOGRAMMETRY & REMOTE SENSING

www.elsevier.com/locate/isprsjprs

CAMPINO — A skeletonization method for point cloud processing

Alexander Bucksch [*], Roderik Lindenbergh

Delft University of Technology — Delft Institute of Earth Observation and Space, Systems, P.O. Box 5058, 2600 GB Delft, The Netherlands

Received 5 December 2006; received in revised form 27 September 2007; accepted 8 October 2007
Available online 21 December 2007

Abstract

A new algorithm for deriving skeletons and segmentations from point cloud data in $O(n)$ time is explained in this publication. This skeleton is represented as a graph, which can be embedded into the point cloud. The CAMPINO method, (C)ollapsing (A)nd (M)erging (P)rocedures (IN) (O)ctree-graphs, is based on cycle elimination in a graph as derived from an octree based space division procedure. The algorithm is able to extract the skeleton from point clouds generated from either one or multiple viewpoints. The correspondence between the vertices of the graph and the original points of the point cloud is used to derive an initial segmentation of these points. The principle of the algorithm is demonstrated on a synthetic point cloud consisting of 3 connected tori. Initially this algorithm was developed to obtain skeletons from point clouds representing natural trees, measured with the terrestrial laser scanner IMAGER 5003 of Zoller+Fröhlich. The results show that CAMPINO is able to automatically derive realistic skeletons that fit the original point cloud well and are suited as a basis for e.g. further automatic feature extraction or skeleton-based registration.

Keywords: Skeletonization; CAMPINO; Point cloud; Terrestrial laser scanning

1. Introduction

Terrestrial Laser Scanning (TLS) is a method for generating point clouds, that is known since the 80's (Blais, 2004). Nowadays it gives the opportunity to generate millions of distance measurements in one scan procedure. Many laser scanners, like the IMAGER 5003 of Zoller+Fröhlich, used for our experiments, produce a panoramic scan. The whole set of measurements, a point cloud, is stored in a data structure containing a vertical angle θ, a horizontal angle ϕ and the distance R between scanner and object. The (ϕ, θ, R)-coordinates are usually transformed into a cartesian (x, y, z)-coordinate system. The instrument's angular resolution determines the amount of gathered data. The combination of several overlapping scans to one point cloud within a common coordinate system is called registration. Often a 3-step procedure is applied for registration and processing point cloud data.

(1) Registration of the single scans obtained from different viewing angles with the "Iterative Closest Point"-algorithm (ICP), (Besl and McKay, 1992; Rusinkiewicz and Levoy, 2001),

* Corresponding author.
E-mail address: a.bucksch@tudelft.nl (A. Bucksch).

0924-2716/$ - see front matter © 2007 International Society for Photogrammetry and Remote Sensing, Inc. (ISPRS). Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.isprsjprs.2007.10.004

(2) Reconstruction of the surface as represented by the point cloud (Mencl, 2001; Edelsbrunner and Mücke, 1994; Amenta et al., 2001),
(3) Extraction of features or topology from the reconstructed object surface (Rabbani, 2006).

Current state of the art algorithms, like the ones given as examples in the three steps, operate on single points. But single points carry no information about the local object structure. This information can also not be extracted from a small neighborhood. In our opinion it is necessary to make a step towards algorithms that consider the links between sets of points. A link will be represented as an edge in a graph, which is derived from the local object structure of the represented object, while each vertex incident to the edge corresponds to one homogeneous subset of points in the point cloud.

The purpose of the algorithm is to obtain sets of points from the point cloud and describe the links between them in order to extract a graph description of the data, which may facilitate further processing like the registration of scans, the measurement of scanned objects or the reconstruction of object surfaces.

A skeleton is defined as a graph containing geometric information with its vertices and edges. While the 3- or more connected vertices represent a topological change in the object, the edges represent the link between them. An example of a skeleton is shown in Fig. 9.

This publication is organized in five sections. Section 2 gives a short introduction to skeletonization algorithms. The newly developed CAMPINO algorithm is presented in Section 3 and is explained with a synthetic data set. Before heading on to the discussion of the feasibility and the future of this algorithm in Section 5, the results of the CAMPINO algorithm on real data are validated in Section 4. In this section also a time analysis is given.

2. Existing skeletonization methods

This section gives an overview of existing skeletonization principles to extract a one-dimensional description from a point cloud representing individual objects. The terms centerline, skeleton and medial axis are all named in literature to indicate one-dimensional descriptors (Greenspan et al., 2001; Palagyi et al., 2001; Cornea et al., 2005). We will use the term skeleton throughout this paper, without distinguishing between the different terms. Some of the given methods may produce surfaces when applied on a 3D object, like the medial axis approach or the distance field method. To obtain a one-dimensional descriptor such surfaces should be further reduced by e.g. thinning, (Palagyi et al., 2001).

A well-known method of skeleton extraction is based on the distance field approach. The set of all distances to an object boundary in a given space defines a distance field. The connection of all local distance maxima within the object interior, forms the skeleton of the object (Satot et al., 2000; Wan et al., 2001). First a main skeleton is extracted, and, in a second step, skeletons of smaller object parts are connected as branches to the main skeleton. The main problems of this approach are its inefficiency for selecting and connecting the local maxima and the non-uniqueness of the derived skeleton (Jiang and Alperin, 2004).

The medial axis transform (Dey, 2007) is another way to extract a skeleton of an object. The medial axis is defined under the assumption that the object is watertight, that is, no holes are allowed in the object surface. A point p_1, belongs to the medial axis of an object, if the set of boundary points most close to p_1, consists of at least two points. Imagine a sphere, which is touching the surface of an object on at least two points, then the center of the sphere is part of the medial axis. The definition guarantees that the axis is always inside the object. The representation of the object is a set of spheres, whose centers describe the medial axis, and whose common closure describes the object boundary. Proofs and further medial axis properties can be found in Amenta et al. (2001) and Dey (2007). To derive a one-dimensional descriptor from 3D data it is necessary to reduce planar parts of the medial axis to a line description. As emphasized above, the medial axis is only defined on closed objects, therefore it is not possible to extract a skeleton from a single scan. These arguments give us some more motivation to directly determine a skeleton out of a point cloud.

A successful way of skeletonization is explained in Gorte and Pfeifer (2004). In this approach the sequential data thinning method of Palagyi et al. (2001) is used to skeletonize terrestrial laser scanner point clouds. The point cloud is first transformed into a 3D-raster. A raster cell is called a voxel. This method requires a pre-processing step in which the morphological operations opening and erosion are used to fill holes in the object and remove noise (Serra, 1982). The drawback of this algorithm is the large number of parameters that needs to be controlled, like resolution of the 3D-raster, and type and size of the structuring element. Furthermore, when increasing the resolution, an increasing tendency of producing cycles in the resulting skeleton occurs, also depending on the quality of data. The general drawback of thinning methods is that they may extract only parts of the skeleton, which need to be connected in a post-processing step. Moreover, thinning methods are known to be sensitive to changes on the object boundary. This sensitivity occurs, because the thinning methods try to approximate the medial axis, which in itself is already sensitive to small changes on the object boundary.

A skeleton extraction method based on fuzzy clustering is described in Katz and Tal (2003). A skeleton is produced by decomposing a polygonal mesh into its components using a fuzzy clustering strategy. After this decomposition, a vertex for the resulting skeleton graph is placed at the center of mass of the polygon defined by a curve between two decomposed parts. This method needs a segmentation step, based on the polygonal mesh information before the skeletonization procedure can be started. Neither such an initial mesh nor a segmentation is known for unprocessed point clouds.

We conclude from this section that medial axis based methods operate as bottom-up approaches, by starting up on the level of single points. Their major disadvantage is, that it is unknown which points are caused by noise and where object points are missing. Therefore we have developed a top-down approach by looking at the global object structure first, and consecutively descending to finer levels of detail after that. This guarantees a fully connected skeleton as a result. A post-processing step to connect skeleton parts or to further reduce resulting surfaces is in general not needed, because the result is already a one- dimensional descriptor. It will be shown later, that the requirements for noise reduction and hole filling are low.

3. CAMPINO algorithm — Collapsing and merging procedures in octree graphs

This algorithm takes a point cloud as input and returns a skeleton of the object represented by the point cloud, together with a skeleton-based point cloud segmentation. The algorithm consists of four major steps.

(A1) Generation of an octree
(A2) Extraction of a graph from points in the octree cells
(A3) Removal of cycles in this graph
(A4) Splitting of vertices with many incident edges

In this context, a cycle of a graph G is a subset of the edge set of G that forms a path such that the first node of the path corresponds to the last (Wilson, 1985). In the first step an octree is built to contain the point cloud. The two parameters defining the octree are:

(P1) the minimal acceptable cell size and
(P2) a threshold to determine if the point cloud is intersecting a cell side.

In the second step an octree graph is constructed, dual to the octree structure. In general this octree graph will contain cycles, which are removed in the third step by merging certain vertices. This will result in a reduced octree graph that represents a point cloud skeleton. From this skeleton a segmentation is derived.

3.1. Octree generation

To determine if further subdivision is necessary during the octree generation, subdivision termination rules are defined based on how the point cloud intersects the six sides of an octree cell. If an octree cell contains no points, it is simply removed. A cell side is called a touched side, if at least one point cloud point is within threshold distance of the side. This 'touched side' threshold is specified relative to the current level octree cell size. Subdivision stops if one of the following terminating conditions is met, see Fig. 1:

(C1) A cell is touched on three sides or less.
(C2) A cell is touched on four sides and the midpoints of the touched sides are in one plane.
(C3) The cell size is lower than a predefined value.

It is possible that subdivision terminates while a cell contains points of different object parts. To detect such an under-subdivision case, two criteria at the next subdivision level are checked for every cell where the algorithm is about to terminate because condition (C1) or (C2) is met. The first criterion is whether the cells of

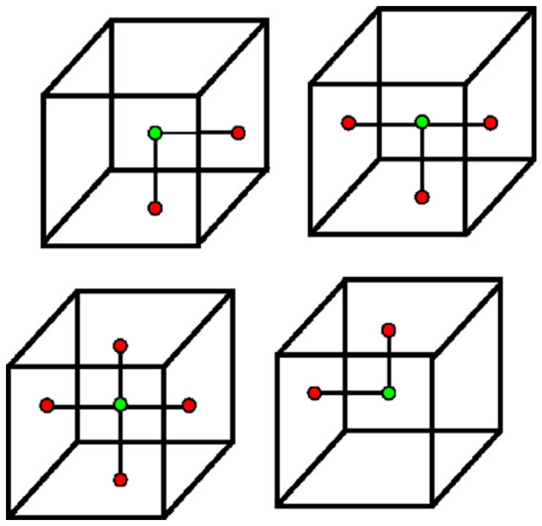
Fig. 1. Examples of the terminating conditions. Condition 1 is visible in the upper two cubes and the right bottom cube. Condition 2 is illustrated in the left bottom corner.

the next subdivision level have new interior sides on the inside of the original cell. The second is whether an untouched side of the original cell gets touched on the next subdivision level. For each kind of side a criterion is formulated. That is, under-subdivision occurs if

(C1) not all 8×3 new interior sides are touched
(C2) the touched side threshold is met for any of the new outer sides.

Ideally, our intermediate result at this stage is a subdivision which separates all parts of the object that are also spatially separated. Clearly, the separating power of the subdivision depends on the maximum resolution of the octree. It should be recognized that the extraction is based on the local object structure in the neighborhood of the octree cells. This property overcomes the typical rotational dependency problems associated with octrees. Also in our case, a rotation of the object in 3D-space can lead to a different octree, but the extracted graph still represents the local object structure. It can only happen that the intermediate octree graph contains a different number of cycles due to the new subdivision.

3.2. Graph extraction

After collecting the information on where the point cloud is intersecting the octree cells via the touched sides, we are able to extract the octree graph from the octree subdivision. The octree graph contains two types of vertices:

M-vertex: An M-vertex represents the geometric midpoint of an octree cell.
With an M-vertex, initially the octree cell it is representing is stored.

T-vertex: A T-vertex represents the center of a touched cell side. It has an edge incident to the M-vertex representing the cell midpoint. With a T-vertex, initially the 3D position of the cell side center is stored.

Each cell is represented by its dual star graph (Wilson, 1985). The adjacency structure of the octree is used to connect the star graphs by merging neighboring T-vertices, as shown in Fig. 2. When two neighboring cells are on the same subdivision level, the T-vertices of the shared side are on the same position and simply merged. If the neighboring cells are on a lower level, the T-vertex is merged with the 1, 2, 3 or 4 neighboring lower level T-vertices. The position of the newly created, merged T-vertex is the mean of the positions of the deleted T-vertices.

The described extraction procedure is only operating in the 6-neighborhood of shared sides of each octree cell. As an extension one can consider neighboring edges and vertices as well.

3.3. Cycle removal rules

Due to the nature of this graph extraction procedure, such a derived octree graph will in general contain cycles. The art of removing these cycles is explained in this section. As a result only the skeleton remains. The removal procedure consists of rule-based merging of M-vertices. A newly created M-vertex, which is the result of merging two M-vertices, will point to the union of all the octree cells its ancestors pointed to. Note that the octree graph, being dual to the octree, is at most six-connected. Therefore the cycles contain a maximum of 4 M- and T-vertices. The following three cases cover all possibilities.

(1) 2 M- and 2 T-vertices.
(2) 3 M- and 3 T-vertices.
(3) 4 M- and 4 T-vertices.

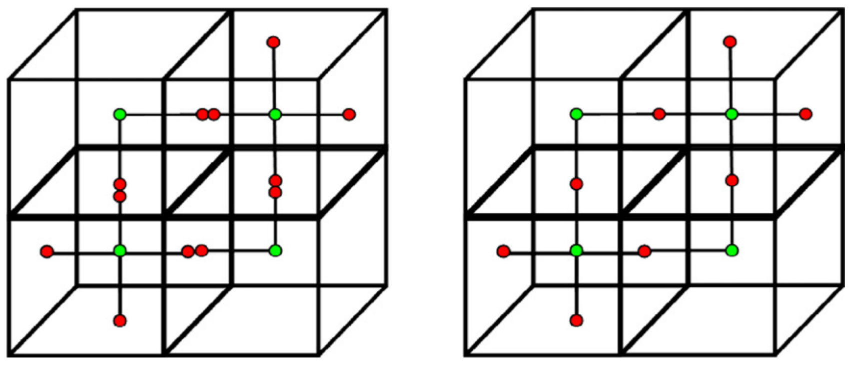

Fig. 2. Merging of T-vertices. The T-vertices are colored red and the M-vertices are colored green throughout all following figures.

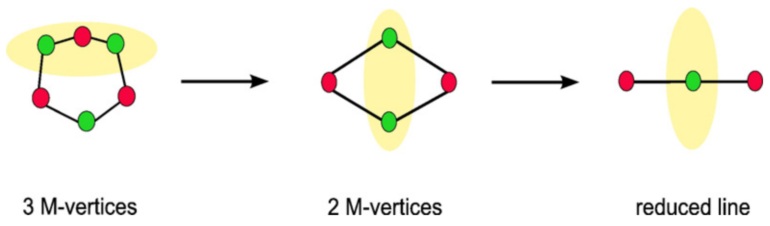

Fig. 3. The basic rules for reducing the 3M/3T-case to the 2M/2T-case and as a result to a line segment.

These cases are subsequently reduced to one or more sequential M–T edges.

3.3.1. *Case 2M/2T*

The case of 2 M- and 2 T-vertices occurs when reducing a 3M/3T case or when reducing certain 4M/4T cases. It is illustrated in Fig. 3. The 2M/2T cycle is simply replaced by one M-vertex. Alternatively, it is possible to merge the two T-vertices. This will lead to a finer skeleton graph, and was preferred in our implementation.

3.3.2. *Case 3M/3T*

The case of three M- and T-vertices does not occur naturally, but only as an intermediate step in the reduction of the next cases. Those two M-vertices that are connected by the shortest path in the cycle are merged to one M-vertex. As a result, a 2M/2T cycle is obtained, compare Fig. 3. This reduction step may lead to wrong connectivity results, if the minimum cell size of the octree is larger than the smallest topological change to be represented. Remember that a topological change is defined as a vertex with 3 or more, if no knot splitting is applied, incident edges.

3.3.3. *Case 4M/4T*

The rule for reducing cycles of 4M- and 4T-vertices is divided into four different sub-rules. These cases are characterized by the number of M-vertices with more than two incident edges in the cycle. Such an M-vertex is called an *x*M-vertex. An overview of all cases is shown in Fig. 4.

1 xM-vertex. This case implies that the cycle is connected to the rest of the octree graph via the unique *x*M-vertex. The three non *x*M-vertices are merged to one M-vertex. As a result a 2M/2T cycle is obtained.

2 xM-vertices. Two combinations exist. The first case occurs, when the point cloud behaves locally like a straight pipe. In this case the two *x*M-vertices are connected

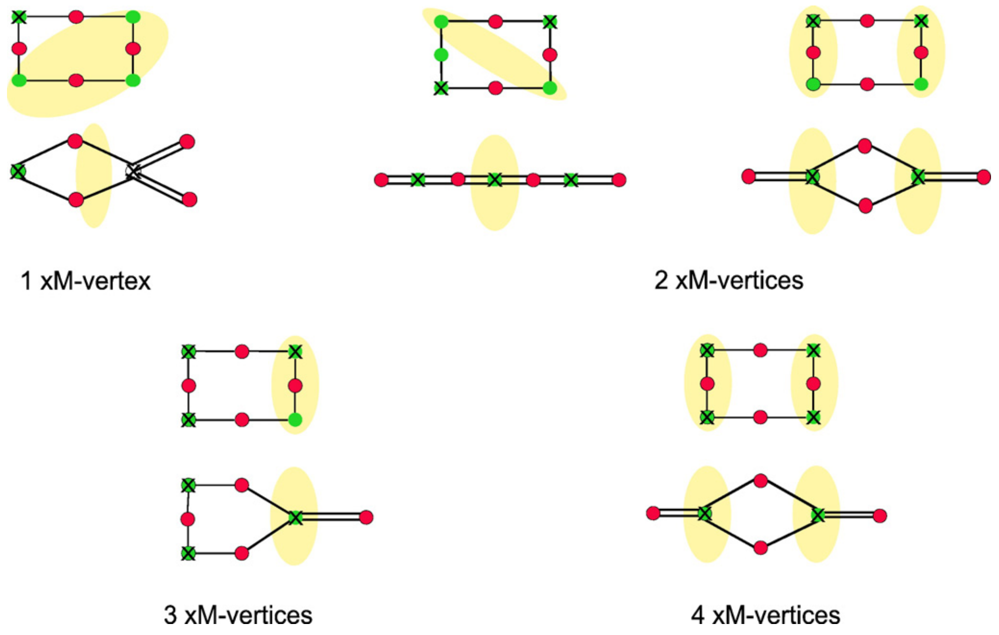

Fig. 4. Reduction scheme of all 4M/4T-cases. Resulting edge collapses are marked with double lines.

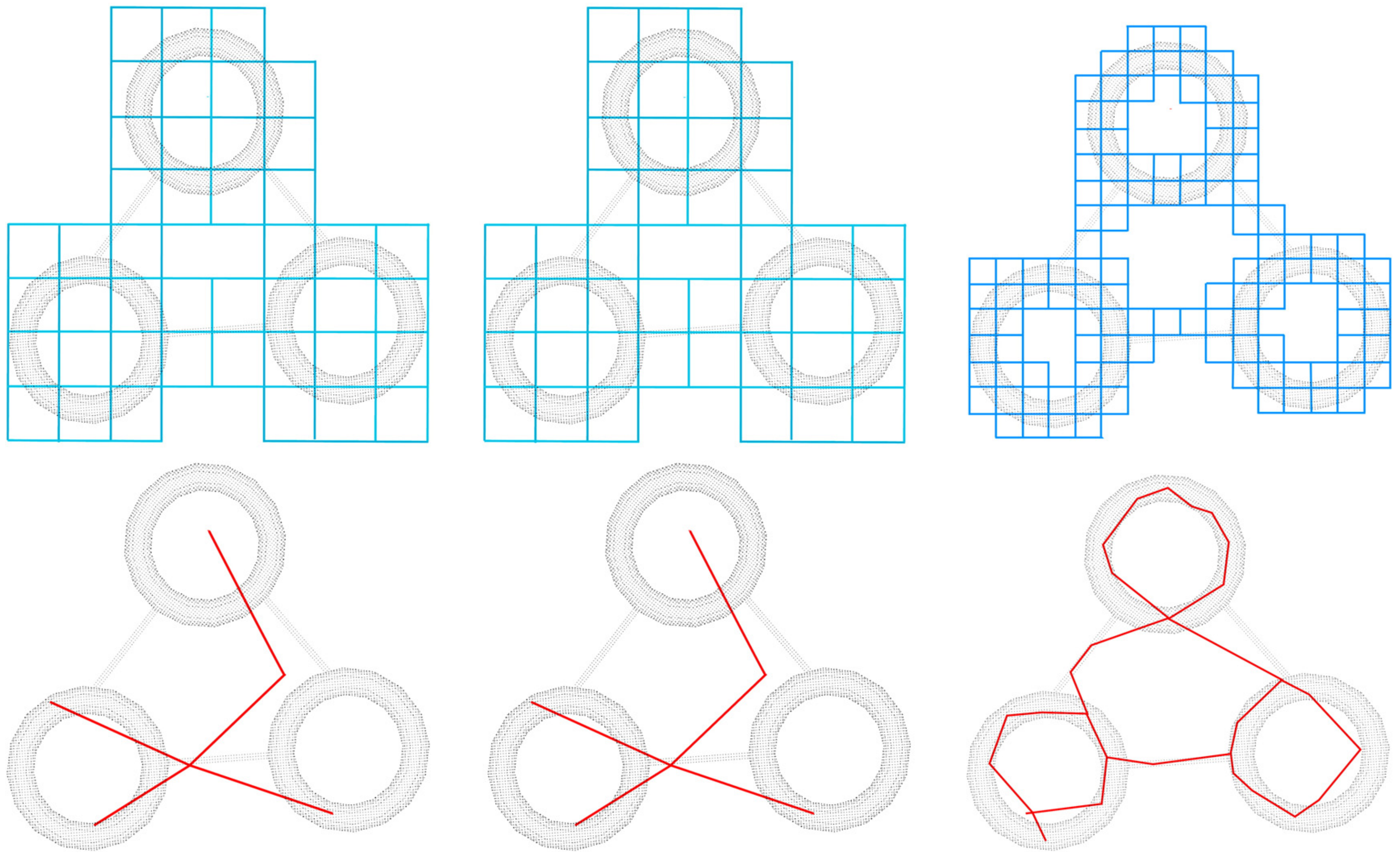
Fig. 5. From left to right: chosen minimum cell sizes of 1.00 m, 0.80 m and 0.60 m. In the top row the octree is shown, while the bottom row shows the resulting skeleton.

through a T–M–T subgraph. By merging the two non *x*M-vertices, the loop is removed, Fig. 4.
In the second case, the two *x*M-vertices have an incident T-vertex in common, that is, the point cloud behaves locally like a bent pipe. Now each of the two non *x*M-vertices are merged with the closest *x*M-vertex, Fig. 4, reducing the 4M/4T cycle to a 2M/2T cycle.

3 xM-vertices. Such a structure only exists on corners of the object represented by the point cloud. The loop is completely removed by merging the one non *x*M-vertex with the *x*M-vertex in the cycle to which the path is shortest.

4 xM-vertices. Every 4 *x*M case is either fully surrounded by other cases or is representing an over-subdivision. The two *x*M-vertices joined by the shortest path are merged, as are the two other *x*M-vertices. This removes the cycle.

We used different strategies to select removable cycles from the octree graph. It never influenced the results significantly. The results demonstrate that the skeleton is always bounded by the initial octree cells. A large difference in processing time using different strategies was experienced though. The fastest convergence to the skeleton is achieved by starting at cycles containing 2-connected M-vertices, and subsequently continue with cycles containing 3-connected until *n*-connected M-vertices. With this procedure we experienced the minimum number of cycle searches among the tested strategies. The algorithm terminates, when all cycles are removed.

3.4. Reference object

As a reference object we have chosen three connected tori arranged in a triangle, Fig. 5. The tori were sampled with 40 mm sampling distance between the points. The algorithm contains two parameters. One is the minimum cell size, and the second one a threshold defining a touched side. Fig. 5 illustrates the influence of the minimum cell size parameter on the skeleton. The under- subdivision test guarantees that small changes of the minimum cell size values have non or only small influence on the result. As can be seen in Fig. 5, a minimum cell size of 1.00 m results in the same octree as obtained when using a minimum cell size of 0.80 m (Fig. 5 left and middle). A count of the resulting octree cells reveals that in these two cases the under-subdivision detection procedure results in octree cells that are smaller than the minimum cell size. If the defined minimum cell size is set to 0.60 m (Fig. 5 right), a skeleton is obtained showing all the object holes correctly. The resolution values given here have to be interpreted with respect to the dimensions of the reference object in *x*-direction of 6.24 m, in *y*-direction of 0.44 m and in *z*-direction of 5.62 m. Decreasing the touched side threshold from 35% to 15% will result in unconnected graph parts.

3.5. Knot splitting procedure

The cycle removal procedure may results in M-vertices with more than 3 incident edges, so-called knots. These knots especially arise in areas where the data is not dense enough. The following procedure is repeated until no M-vertex has more than three incident edges.

The basic idea is to reduce the number of edges incident to an M-vertex by merging adjacent T-vertices. This does not violate the M–T structure while the position of the M-vertices, important for the spatial embedding of the skeleton, is maintained. The procedure is illustrated in Fig. 6. For a given M-vertex M with more than three incident edges, the two incident T-vertices T_1 and T_2 are selected that minimize the angle between the edges MT_1 and MT_2, among the pairs of M-incident edges, Fig. 6a. Vertices T_1 and T_2 are merged to a new T-vertex T_M. The position of T_M is at the so-called Steiner point of the triangle $\Delta(M\, M_1\, M_2)$. The Steiner point is placed at the intersection of the

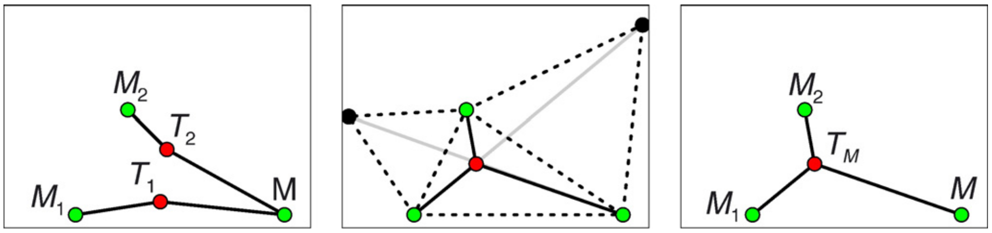

Fig. 6. Merging two T-vertices T_1 and T_2 to a new T-vertex T_M via Steiner construction.

lines between the points M, M_1 and M_2 and the outside points of the equilateral triangles placed on the edges of triangle $\Delta(M\,M_1\,M_2)$, Fig. 6b, Chang (1972). The position of T_M is such, Fig. 6b, that the angles between its consecutive edges are at 120°, which will ensure a well-balanced skeleton in the end.

In our implementation we always start at the M-vertex with the highest amount of incident edges and redo the splitting procedure until the 3-connectedness is achieved. To guarantee an alternating T–M–T structure in the resulting skeleton graph, it is necessary to insert M-vertices between any TT-connection that is created by the knot-split procedure.

3.6. Derivation of the segmentation

During the whole cycle removal procedure M-vertices are merged. Initially, every M-vertex corresponds to one initial octree cell. The duality between the initial octree graph and the octree cells implies, that every M-vertex of the skeleton graph represents a connected set of neighboring octree cells after the cycle removal. The incident T-vertices indicate the border to neighboring connected sets.

For the generation of the segmentation, all adjacent 2-connected M-vertices are merged to one. A point cloud part belonging to one M-vertex is considered a segment of the octree graph derived segmentation. According to the properties of the cycle removal rules, two neighboring segments are topologically different in the sense that their branching characteristics of corresponding M-vertices differs.

4. CAMPINO evaluation

In this section we analyze the CAMPINO algorithm in terms of temporal complexity, skeleton complexity and skeleton validity. As measure for temporal complexity an analysis in O-calculus is given as a machine independent description, (Sipser, 2001). Several skeletons of one natural tree are compared to the original point cloud to show validity and complexity. In the last paragraph not only the skeleton representing a different natural tree, but also the corresponding segmentation is discussed.

4.1. Time analysis

In this paragraph the time behavior of the CAMPINO algorithm is discussed. Let n denote the number of points in the point cloud. Determining a starting cell for the octree, like the smallest enclosing cube of the point cloud, is done in $O(n)$, by selecting the minima and maxima of each coordinate. The first subdivision step is $O(n)$ again, because we only have to check each point against the boundary of one of the new eight cubes, whose dimensions directly follow from the starting cell. For the whole octree with all subdivision levels, the time complexity is therefore $O(dn)$, where d denotes the depth of the octree. In our case $d \ll n$, because we define a minimum box size, which should contain several points. We conclude that building up the octree takes $O(n)$ time.

To extract the octree graph, every point per cell is checked against the boundaries of the octree cell it is belonging to. For each cell side, the distance to the closest point is validated against the intersecting-cell-side threshold. This is done during the subdivision and has a maximum time complexity of $O(n)$.

In general, the number of vertices in the octree graph is smaller than n. In the worst case however, a point is alone in an octree cell, which may result in 7 vertices per point. Therefore the number of vertices is bounded by $7n$.

In the subsequent processing step an adjacency list is built to store the octree graph. This can be done in $O(v)$ time, where v is the number of vertices in the octree graph. In the following step all octree graph cycles need to be found. The known optimum solution for that is $O(v+e)$, where e is the number of edges (Leisserson et al., 1989). The worst case occurs when $e=v-1$. But usually e is much smaller than v in the case of this algorithm.

The required time to sequentially check if an edge exists between two vertices is proportional to the largest number of edges incident to one vertex in the list. The same holds for the operations deletion and replacement. The maximum number of edges incident to one vertex is in the worst case 6 for the initial octree graph. The number of edges e in the whole octree graph is at most $e=6v$, because the collapsing rules only remove edges from the graph. Cycles only occur in squares of four connected octree cells. As every cycle comes from a different square, the number of cycles is bound by the number of octree cells and is therefore bound by $O(n)$.

We conclude that the overall time complexity of the algorithm is $O(n+v+e)$. By replacing the e with worst case $6v$, and v with the worst case of n, we get a worst case complexity of $O(8n)$. Because we are not interested in the constant, the algorithm is $O(n)$.

4.2. Skeletonizing natural trees

In the remaining of this section several results are presented on analyzing point clouds representing natural trees. Constraints exist between geometric

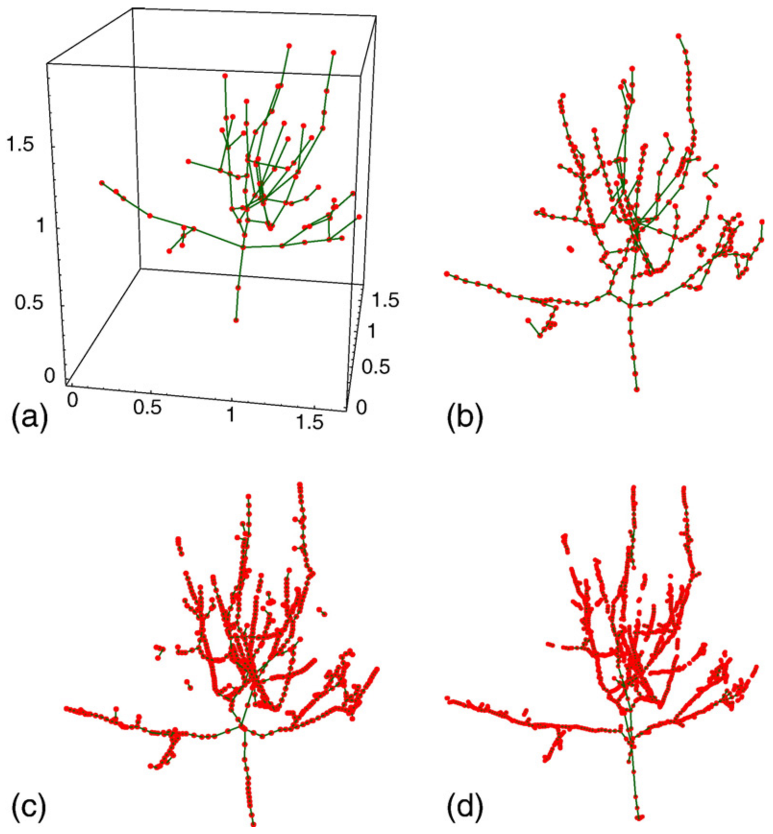

Fig. 7. Natural tree skeletonization of 7219 points. (a) Skeleton obtained by hand. The bounding box visualizes the dimensions of the tree. Units are in meter. Skeletons derived with a minimal octree cell size of (b) 20 cm, (c) 10 cm, and (d) 5 cm.

features, such as branch length and width and the fruiting and healthiness of trees (Fleck et al., 2004). But the collection of these features is a time consuming task. TLS is a fast and powerful method to collect such geometric data. All point clouds discussed here were obtained by the IMAGER 5003 terrestrial laser scanner of Zoller+Fröhlich.

The CAMPINO skeletonization method presented in this publication is able to immediately extract features like length and number of branches. Diameters of branches at certain branch positions can be extracted, by fitting ellipses to the point cloud part surrounding the skeleton at a certain position. The points representing the branch are given by the CAMPINO segmentation. The CAMPINO skeleton allows us to identify the forks in the tree by selecting the vertices in the skeleton with 3 or more (if no knot splitting is applied) incident edges.

For this particular application it is sufficient to know the correspondence between the skeleton graph and the initial octree cells. It is further possible to enforce embedding of the skeleton graph into the object by placing T-vertices in an appropriate ellipse, if this becomes necessary for future applications.

4.3. Skeleton analysis

For the purpose of analyzing the skeleton complexity and validity, a point cloud of an orchard apple tree of about 2 m height was skeletonized. This point cloud consists of 7219 points. As input parameters for the algorithm minimum cell sizes of 20 cm, 10 cm and 5 cm are used. As threshold for determining touched sides 35% of the actual box size was chosen. Except for the CAMPINO skeletons, also one manually derived skeleton is included in the evaluation. First we compare

Table 1
Skeleton properties for different resolutions

Resolution	Edges	Components	Joints	Max $d(S, P)$	Mean $d(S, P)$
Manual	83	1	25	0.226 m	0.023 m
0.20 m	281	2	23	0.168 m	0.034 m
0.10 m	647	9	51	0.156 m	0.018 m
0.05 m	1364	40	120	0.230 m	0.012 m

properties of the different skeletons, then we compare the skeletons to the original point cloud.

4.3.1. Skeleton complexity

The skeleton graphs, as obtained for different minimal octree cell sizes, are shown in Fig. 7(b)–(d). As a reference, a manually derived skeleton graph is shown in Fig. 7(a) With increasing resolution, the number of edges increases as well, allowing for a more detailed representation of the skeleton. Table 1 shows that the number of edges in the skeleton graph about doubles with doubling resolution. In contrast with the manual skeleton, the CAMPINO skeletons are suited for obtaining direct approximations of branch lengths.

A fork in the skeleton graph is a vertex with at least three incident edges. Ideally, forks should correspond to a branching point in the natural tree. Table 1 shows that also the number of forks approximately doubles as resolution doubles. Large number of forks may indicate over-skeletonization due to the presence of outliers An example of over-skeletonization is visible in Fig. 7(d), where some edges are leaving the main trunk at the bottom of the tree. Such small erroneous skeleton parts can easily be removed by filtering them based on their length. In comparison, when using a medial axis approach, such outliers would result in a skeleton with irregular curvature behavior. Such a skeleton is more difficult to correct in a post-processing step.

Note that with increasing resolution the number of connected components grows from 2 for the 20 cm resolution case to 40 for the 5 cm resolution case. Parts of thin branches that are not sufficiently represented by the point cloud cause discontinuities in the skeleton graph. We expect that these discontinuities can be partly removed by extending the CAMPINO algorithm by a hierarchical step that maintains connections that exist at a coarser resolution, in order to connect parts that become separated at finer resolutions.

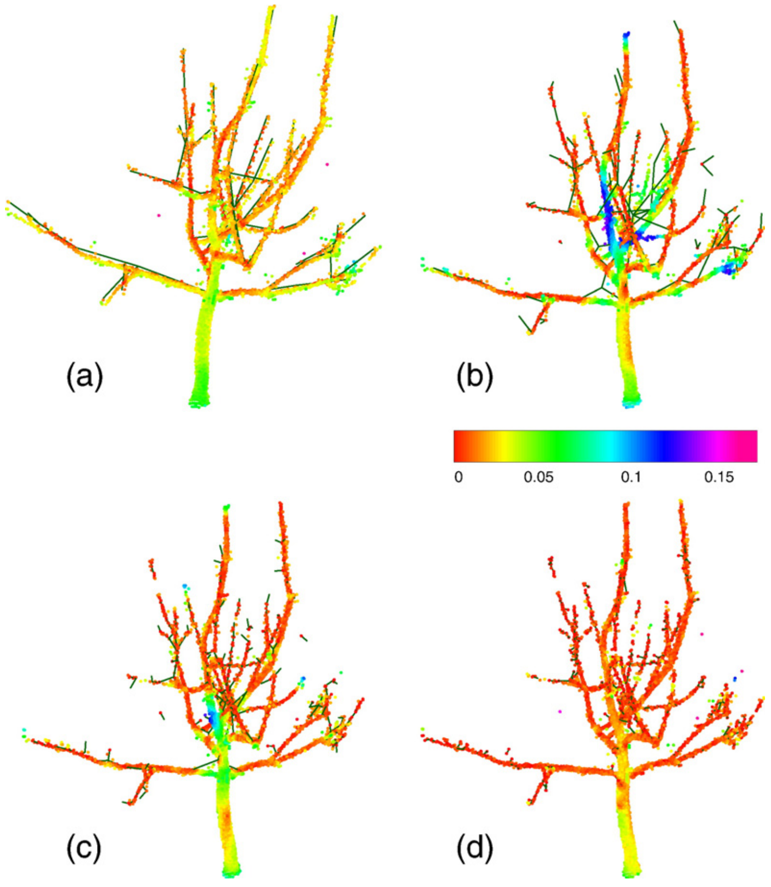

Fig. 8. Original point cloud, superimposed on the CAMPINO skeleton graphs, as obtained with a minimal octree cell size of (b) 20 cm, (c) 10 cm, and (d) 5 cm. Fig. 8(a) shows the manually derived skeleton graph. The point cloud points are colored with respect to their distance to the skeleton. 3D-versions of these figures are available at http://enterprise.lr.tudelft.nl/~lindenbergh/CAMPINO.

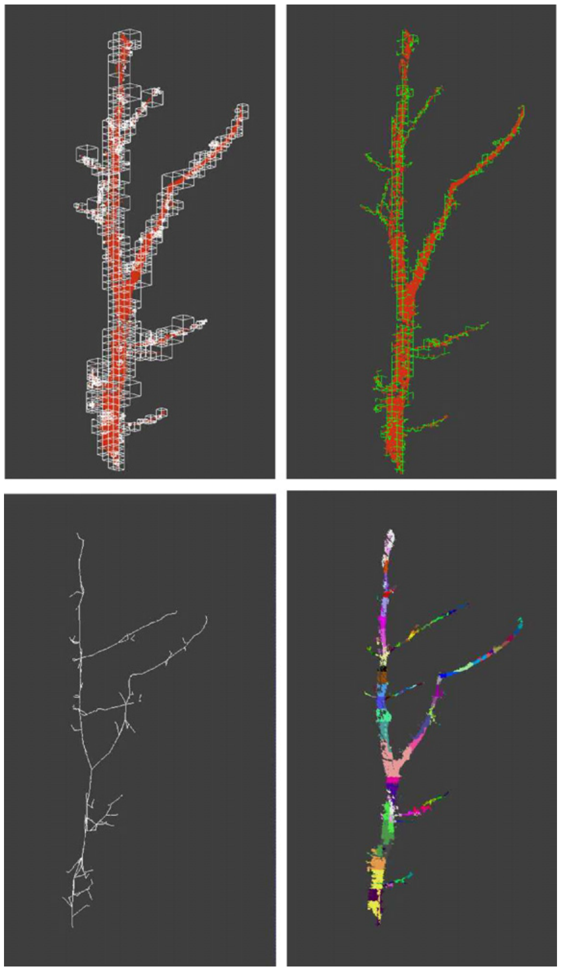

Fig. 9. Natural tree skeletonization of 99 339 points. The tree is 4.01 m high. Top left. Octree. Top right. Derived octree graph. Bottom left. The skeleton graph. Bottom right. The derived segmentation. Gaps are easily visible in the most top segment or the big pink fork segment in the middle.

4.3.2. Skeleton validity

The validity of the skeleton graphs to represent the object as sampled by the point cloud, is assessed by comparing the point cloud to the different skeletons. Except for a visual inspection, we use for this purpose the Euclidian distance between the points in the point cloud and the skeleton. Note that the distance $d(P, S)$ of a point P to a skeleton S by definition equals the minimum of the distances between P and the edges $e_i \in S$:

$$d(P,S) = \min_{e_i \in S} d(P, e_i) \tag{1}$$

The distance $d(P, e)$ of a point P to and edge $e=[A, B]$, with A and B being the end points of the line segment e, equals the distance of P to the line ℓ_{AB} if the line through P and perpendicular to ℓ_{AB} intersects ℓ_{AB} between A and B. In the other case $d(P, e)=\min\{d(P, A), d(P, B)\}$.

Table 1 shows that the mean distance of the 7219 points to the skeleton, reduces from 3.4 cm in the 20 cm resolution case to 1.2 cm for 5 cm resolution. The spatial distribution of the distances to the skeletons is shown in Fig. 8(b)–(d) together with the skeletons itself. Also the distances to the manual skeleton are given in Fig. 8(a) The skeleton matches the point cloud well with a 5 cm resolution. At more coarse resolution forks of larger branches tend to be located outside the point cloud. This effect is caused by an octree cell that still contains geometrically different parts of the point cloud. Therefore the representing vertex is not placed correctly. This happens for example at the blue points in the middle of the tree at 10 and especially 20 cm resolution. Still the distance between the points and skeleton are within the range of the chosen minimum box size.

At the first branch on the right, some blue points occur at 20 cm resolution, because of the trees bad representation through the point cloud. At 5 cm resolution these problems are solved by the algorithm by creating large numbers of short edges. The relative large distances of point cloud points to the 5 cm skeleton at the bottom of the trunk are just caused by the larger

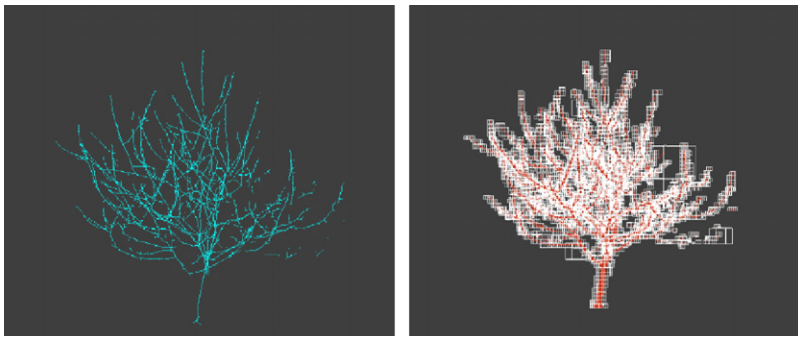

Fig. 10. Skeletonization of one scan consisting of 322 863 points representing one side of a cherry tree of 3.57 m height. On the left the derived skeleton and on the right the octree (white boxes) and corresponding point cloud (red) are shown.

diameter of the tree at its bottom. Although the manual skeleton looks satisfactory, the mean distance of the point cloud points to the manually derived skeleton is twice as high as to the automatically derived 5 cm skeleton.

4.4. Natural tree segmentation

In Fig. 9 the whole CAMPINO procedure is illustrated on a small apple tree. The top left figure illustrates the octree subdivision, which was generated by the conditions respecting the local object structure, compare Section 3. This subdivision differs from the standard octree subdivision, which depends on the number of points per cell. The top right of Fig. 9 shows the octree graph as derived from the octree cells. On the bottom left the skeleton after applying the reduction rules is given. The simultaneously obtained segmentation, reflecting locally unique branching characteristics, defines the point cloud coloring in the bottom right illustration.

Varying point density because of different measurement distances to the branches, or small gaps because of missing points, can be closed with a large octree cell (Fig. 9). The small gaps are best visible in the bottom right picture of Fig. 9.

The same procedure was applied on a point cloud consisting of 322 863 points, representing a large cherry tree. The results are shown in Fig. 10. The difference to the previous examples is the complexity of the object and the fact, that this skeleton is based on one scan from just one view point.

5. Conclusions and future work

We presented a skeletonization algorithm, which delivers both a skeleton graph and an initial segmentation in $O(n)$ time as a result. In this section an evaluation of the new algorithm is given.

5.1. Discussion of the results

The CAMPINO skeleton gives a suitable solution for objects that are only partly represented by a point cloud, like in the case of a single scan. It directly extracts a skeleton, which is represented as a graph, whose vertices point to connected sets of octree cells. It is in general not necessary to connect skeleton parts to each other like in thinning-based approaches. It is robust to varying point densities and missing points in the point cloud due to filtering or occlusion. This is because we take the local object structure into account by considering where a point cloud part is intersecting with our organizing structure, the octree. In this way a lower sensitivity to noise is achieved.

The final result depends on the depth of the octree. The octree approach allows to close the gaps in the point cloud due to occlusions, if the octree cell in this region is big enough to cover this gap. It was experienced, that the CAMPINO skeletonization runs into topological faults, if the resolution of the octree is not sufficient. This results in the connection of neighboring elements, that should not be connected. Unwanted connections can occur because of the minimum-angle-selection step in the knot splitting procedure or because of a not properly dimensioned threshold for determining the touched sides. These problems cannot be fully eliminated, but steps towards a better solution are already made by investigating the topological/branching structure of the octree graph.

Insight into the quality of the CAMPINO method is obtained by considering the distances of the original point cloud to the resulting skeleton graphs. Once a suitable resolution is found, the results of the skeletonization are satisfactory. Except for its high tolerance with respect to quality of the input point cloud, the most important benefit of CAMPINO algorithm is that, when compared to medial axis based methods, it is hardly influenced by small changes on the object boundary. Even in higher resolutions the propagated skeleton is represented in the skeleton graph, although noisy points will result in the creation of small extra skeleton edges that stick out the main skeleton graph. Such erroneous edges can be automatically removed in a post-processing filtering step.

Future research should focus on how to create optimal conditions for this algorithm. Suitable settings for the CAMPINO method can directly be linked to the resolution of the input point cloud. On the algorithmic side the problem of topological correctness is not yet fully solved.

5.2. Future missions

One extension of the CAMPINO algorithm is to optimize the current search strategy. This is expected to improve the processing time because less resolution is needed. An application of the CAMPINO algorithm that will be considered is registration of scans based on their skeleton graphs only rather than on all the individual points in the respective scans. Robust skeleton graphs are expected to be useful in various fields like e.g. 3D animation or computer games. Even more abstract topics like storage description in GIS databases are conceivable. Furthermore these descriptions can serve as input to simulations of wind or fluids if the describing graph is labeled with values describing elasticity or translucency properties. Further validations are planned on actual applications like the proposed tree measurements.

Acknowledgements

This research is partly financed by the ”Arbeitsgemeinschaft industrieller Forschungsvereinigungen” (AiF) in Germany. We would like to thank Stefan Fleck from the Albrecht-von-Haller-Institut at the University of Göttingen, Ben Gorte from the Delft University of Technology and Tahir Rabbani for their valuable comments.

References

Amenta, N., Choi, S., Kolluri, R.K., 2001. The power crust, unions of balls and the medial axis transform. Computational Geometry: Theory and Applications 19 (2–3), 127–153.

Besl, P.J., McKay, N.D., 1992. A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (2), 239–256.

Blais, F., 2004. Review of 20 years of range sensor development. Journal of Electronic Imaging 13 (1), 231–240. January.

Chang, S.-K., 2001. The generation of minimal trees with a steiner topology. Journal of the ACM 19 (4), 699–711.

Cornea, N., Sliver, D., Min, P., 2005. Curve-skeleton applications. Proceedings IEEE Visualization 2005, Minneapolis, USA, pp. 95–102. 23–28 October.

Dey, T.K., 2007. Curve and Surface Reconstruction. Cambridge University Press, London.

Edelsbrunner, H., Mücke, E.P., 1994. Three-dimensional alpha shapes. ACM Transactions on Graphics 13 (1), 43–72.

Fleck, S., van der Sande, D., Schmidt, M., Coppin, P., 2004. Reconstructions of tree structures from laser-scans and their use to predict physiological properties and processes in canopies. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 36 (Part 8/W2), 119–123.

Gorte, B., Pfeifer, N., 2004. Structuring laser-scanned trees using 3D mathematical morphology. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 35 (Part B5), 929–933.

Greenspan, H., Laienfeld, M., Einav, S., Barnea, O., 2001. Evaluation of center-line extraction algorithms in quantitative coronary angiography. IEEE Transactions on Medical Imaging 20 (9), 928–941 September.

Jiang, H., Alperin, N., 2004. A new automatic skeletonization algorithm for 3D vascular volumes. Proceedings of the 26th Annual International Conference of IEEE EMBS, San Francisco, USA, pp. 1565–1568. 1–5 September.

Katz, S., Tal, A., 2003. Hierarchical mesh decomposition using fuzzy clustering and cuts. ACM Transactions on Graphics 22 (3), 954–961.

Leisserson, C.E., Corman, T.H., Rivest, R.L., 1989. Introduction to Algorithms. The MIT Press, Cambridge.

Mencl, R., 2001. Reconstruction of surfaces from unorganzied three dimensional point clouds. Phd thesis, University of Dortmund.

Palagyi, K., Sorantin, E., Balogh, E., Kuba, A., Halmai, C., Erdohelyi, B., Hausegger, K., 2001. A sequential 3D thinning algorithm and its medical applications. Proceedings of the 17th International Conference on Information Processing in Medical Imaging. Springer-Verlag, London, UK, pp. 409–415. 18–22 June.

Rabbani, T., 2006. Automatic reconstruction of industrial installations using point clouds and images. Ph.D. thesis, Delft University of Technology.

Rusinkiewicz, S., Levoy, M., 2001. Efficient variants of the ICP algorithm. Proceedings of the Third International Conference on 3D Digital Imaging and Modeling. Quebec City, Canada, pp. 145–152. 28 May – 1 June.

Satot, M., Bittert, I., Bendert, M.A., Kaufmant, A.E., Naka jima, M., 2000. Teasar: tree-structure extraction algorithm for accurate and robust skeletons. Proceedings of the Eighth Pacific Conference on Computer Graphics and Applications, Hong Kong, China. 3– 5 October.

Serra, J., 1982. Image Analysis and Mathematical Morphology. Academic Press, London.

Sipser, M., 2001. Introduction to the Theory of Computation. PWS Publishing Company, Boston.

Wan, M., Dachille, F., Kaufman, A., 2001. Distance-field based skeletons for virtual navigation. Proceedings of the Conference on Visualization ’01. IEEE Computer Society, Washington, DC, USA, pp. 239–246. 21–26 October.

Wilson, R.J., 1985. Introduction to Graph Theory, third ed. Longman Scientific & Technical, New York.