Eurographics Workshop on 3D Object Retrieval (2009)
I. Pratikakis, M. Spagnuolo, T. Theoharis, and R. Veltkamp (Editors)

SkelTre - fast skeletonisation for imperfect point cloud data of botanic trees

A. Bucksch, R.C. Lindenbergh and M. Menenti[1]

[1]Delft University of Technology, Delft Institute of Earth Observation and Space Systems

Abstract
Terrestrial laser scanners capture 3D geometry as a point cloud. This paper reports on a new algorithm aiming at the skeletonisation of a laser scanner point cloud, representing a botanical tree without leafs. The resulting skeleton can subsequently be applied to obtain tree parameters like length and diameter of branches for botanic applications. Scanner-produced point cloud data are not only subject to noise, but also to undersampling and varying point densities, making it challenging to extract a topologically correct skeleton. The skeletonisation algorithm proposed in this paper consists of three steps: (i) extraction of a graph from an octree organization, (ii) reduction of the graph to the skeleton and (iii) embedding of the skeleton into the point cloud. The results are validated on laser scanner point clouds representing botanic trees. On a reference tree, the mean and maximal distance of the point cloud points to the skeleton could be reduced from 1.8 to 1.5 cm for the mean and from 15.6 to 10.5 cm for the maximum, compared to results from a previously developed method.

Categories and Subject Descriptors (according to ACM CCS): I.4.7 [Computing Methodologies]: IMAGE PROCESSING AND COMPUTER VISION/Feature Measurement—Size and shape

1. Introduction

In recent years instruments capable to measure thousands of distances per second from the instrument to surrounding surfaces became available. One such instrument is a terrestrial laser scanner. These scanners are used to obtain data of large objects, which are typically represented in a point cloud. A point cloud is therefore a sampling of a 3D surface. Extraction of complex botanic tree structures from a point cloud is difficult for several reasons.

1. varying point density caused by the spherical scan geometry of the instrument in a single scan
2. varying point density caused by the alignment of single scans into a common coordinate system
3. undersampling caused by occlusion effects
4. noise and systematic errors masking the object structure

Obtaining Object structure/topology can help in various point cloud applications. Such applications aim at object identification or analyzing the shape parts of a tree in terms of size. A skeleton is a one-dimensional description of the object structure. Skeletons are represented as curves, collections of ordered points or graphs. Their extraction from a point cloud faces several algorithmical challenges. In

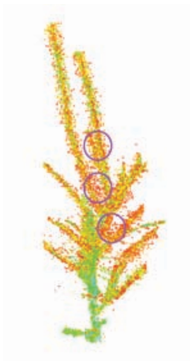
Figure 1: *Point cloud from the inner crown of an leafless orchard tree. The point cloud is colored by intensity.The three marked areas show examples of noise where the separation of branches is even hard by visual inspection*

[CM07] a list of 12 desirable skeleton properties is given. Three of the 12 properties demand special attention, in the context of skeletonising point clouds of botanic trees. First,

DOI: 10.2312/3DOR/3DOR09/013-020

topology preservation of the tree is essential for navigating to a certain position within the tree. Furthermore, proper centering of the skeleton within the point cloud enables the measurement of branch thickness. The third property addresses the computational efficiency, because a point cloud of a small orchard tree already easily consists of 300.000 points. This paper introduces a method for (Skel)etonisation of (Tre)es, here called SkelTre Skeleton, directly from a 3D point cloud by considering all six principal Cartesian directions. The algorithm incorporates three main elements. First an octree is built from which an octree graph is extracted, representing the connectivity between the octree cells with respect to the point cloud. In a second step the octree graph is exploited to retract the point cloud to a skeleton. The third element is a strategy to embed the skeleton graph into the point cloud. To achieve better results on the three properties mentioned above, all three elements have been completely renewed with respect to [BL08].

Topology preservation is enhanced by using a new noise robust criterion to decide on proper connections in the octree graph instead of simple thresholding. The centering is improved by a new embedding strategy taking the approximate shape of the original object into account. A novel graph reduction method is introduced on the basis of approximate surface directions, which are incorporated as vertex labels. These vertex labels form the basis for an increased efficiency, because the new octree graph reduction rules behave linear in time and need less octree subdivisions to achieve improved results in comparison to a previous method.Although, this paper focuses on botanic trees, no restrictions are expected in applications to general shapes represented as a point cloud.

2. Related work

Skeleton extraction from point clouds, as extensively discussed in [CM07], includes several classes of methods. This section gives an overview of methods and points out briefly their problems and benefits.

Morphological thinning methods organize the point cloud in a 3D raster of equally sized cells. From this raster the outer layer is removed until the skeleton remains. Removing the outer layer uses the morphological operations opening and erosion [Ser82]. This class of algorithms requires a defined inner volume of the object to produce a centered skeleton. [PSB*01] introduced a time linear algorithm using 6 sub-iterations. Its application on tree point clouds was proposed by [GP04] and later extended to achieve better connectedness of the skeleton in case of undersampling in [Gor06].

Medial axis based approaches use approximations of the medial axis from the point cloud, e.g. [ACK01]. This class of algorithms derives a skeleton from a Voronoi space division. The medial axis is in general a set of surfaces in 3D, but can be reduced to a skeleton [DS06]. Another possibility to extract the medial axis is the distance transform [ZKW98],but does not guarantee a connected skeleton. All cells are marked by their distance to the object boundary. The set of neighboring maximal distances form the skeleton. The main disadvantage of the medial axis is its sensitivity to irregularities on the object surface. These irregularities may occur because of noise or unsampled parts within the point cloud.

Geometric methods use a function that is shifted over the represented surface for the extraction of a skeleton.The height function is often used to extract the level sets from a given point cloud, e.g. [VL00]. Placement of additional vertices in the centroid of the extracted level sets detail the extracted skeleton. The resulting graph is often referred to as a Reeb-graph [CMEH*04]. The biggest problems arising with these approaches are the rotational dependency of the height function and the sensitivity of the level set extraction to the sampling density [CM07].

Graph reduction based approaches, as introduced in [BL08], extract an initial graph from a spatial subdivision. This initial graph is reduced by a set of rules to a skeleton. These rules consider the connectivity between different parts of the point cloud. Several advantages of such a approach could be demonstrated: a high robustness to noise on imperfect data, a good centeredness and a good connectivity. Centeredness is achieved by embedding the graph into the point cloud. Topological correctness is achieved by choosing a proper decision criterion to place connections between the different point cloud parts and the careful design of the reduction rules.

3. Mathematical background

To simplify further explanations essential mathematical concepts are introduced in this section.

Let S be a surface of an object represented by its point cloud Σ. A spatial subdivision of Σ into subsets Σ_i can be achieved by an octree.

Definition 3.1 The octree space is modeled as a cubical region consisting of $2^n \times 2^n \times 2^n$ unit cubes, where n is the subdivision parameter and 2^n is the length of the octree space. Each unit cube has value 0 or 1, depending on whether it contains data points or not [CH88].

As this paper introduces a graph-based skeletonisation approach, the notion of the extracted graph to be reduced should be introduced. The graph extracted from the octree organization exhibits local grid graph properties.

Definition 3.2 A three-dimensional grid graph is an $m \times n \times r$ graph that is the graph Cartesian product of path graphs on m, n and r vertices [PS03].

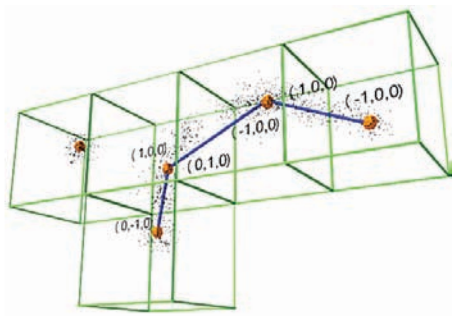

Figure 2: *Connecting and labeling of an octree graph. 5 complete octree cells, containing some black data points. The vertices of the octree graph corresponding to the octree cells are shown in orange. They are positioned in the center of gravity of the local point cloud points. The connectedness of the vertices is based on a robustness criterion.*

4. SkelTre skeletonisation algorithm

This paper introduces a linear-time algorithm for computing a skeleton from an unorganized point cloud of a botanic tree. Unorganized point clouds contain no neighboring information on the points. As an example, consider the alignment of two partly overlapping point clouds into a common coordinate system. The single point clouds are obtained by a terrestrial laser scanner, that measures the points in a line-like order. After alignment of single point clouds in a union point cloud, this order is lost. The new algorithm proposes a noise and systematic error robust process. The resulting skeleton is a geometrically embedded graph. To extract this SkelTre Skeleton, an initial graph, called octree graph, is constructed and further retracted to the SkelTre Skeleton. The edges in the octree graph are labeled to indicate their direction.

4.1. Octree graph extraction

The octree graph extraction is a two step procedure. First an octree is generated as in [BL08], followed by the extraction of an octree graph containing a newly introduced labeling.

4.1.1. Review of the octree generation

Obtaining an octree graph, requires an octree organization as described in [BL08]. To obtain an octree graph, first the point cloud Σ is organized in an octree.The octree subdivides the point cloud by considering how the point cloud is the sides of the octree cells. A cell side E_{ij} between two adjacent octree cells Ω_i and Ω_j is defined to be shared, if both cells contain a point within half the size of E_{ij}.

The subdivision of the point cloud into octree cells terminates if a cell shares three sides or less or if, in case a cell shares four sides, the four midpoints of the shared cell sides are in one plane. Avoidance of disconnected cells is accomplished by defining a minimum cell size.

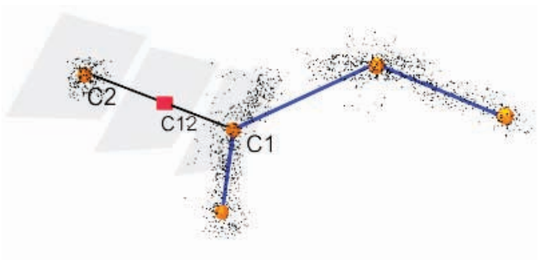

Figure 3: *Robustness criterion to connect octree graph vertices C_1 and C_2 by an edge*

4.1.2. Extraction and labeling of the octree graph

As stated above terrestrial laser scan data is subject to noise, systematic errors, varying point density and undersampling. This can lead to false connections in the octree graph, if the vertices of the octree graph are simply placed at the center of gravity of all points belonging to a cell and connected by an edge, if the common sides are shared. Fig.1 shows typical noise problems of terrestrial laser scan data and Fig.8 shows various occasions of undersampling. The two arising problems for placing connections in the octree graph are wrong additional connections because of noise, which fills the space between shape parts and missing connections because data is strongly undersampled due to occlusions. The criterion to handle undersampling and noise is a decision criterion to place connections between neighboring octree cells.

This robust criterion whether to connect two octree graph vertices, corresponding to two adjacent octree cells, by an edge, is based on the distances of the cell points to three suited planes (Fig.3). Let C_1 and C_2 be the centroids of the point cloud points in two adjacent octree cells Ω_1 and Ω_2. Let C_{12} be the midpoint of the line segment $\overline{C_1C_2}$. The three suited planes P_1, P_2 and P_{12}, are the planes through the points C_1, C_2 and C_{12}, perpendicular to the line through C_1 and C_2. Let d_1, d_2 and d_{12} be the median values of the squared distances of the points in Ω_1, Ω_2 and $\Omega_1 \cup \Omega_2$ to the planes P_1, P_2 and P_{12}. Under ideal conditions the $\sqrt{d_{1,2}}$ of two connected cells would be at least $\frac{1}{4}$ of the distance between C_1 and C_2 to indicate a connection between the two corresponding point cloud parts. Therefore we use $\frac{1}{16}d_{12} \leq \min(d_1, d_2)$ as a criterion to place connections in the octree graph.

Now that the octree graph is extracted and defined, the graph should be labeled. A label belongs to an edge, but is always associated to a vertex. Therefore the label corresponds to the unique Cartesian direction of the edge from the view point of one of the two incident vertices v_i and v_j. Let v_i be a vertex in some graph, and v_j a neighboring vertex connected by an edge e_{ij} to v_i.

Definition 4.1 A label associated with an edge of the octree

graph indicates the direction of the edge with a direction vector. The labels for all 3D-directions are:
Left/Right: $(\pm 1, 0, 0)$,
Up/Down: $(0, \pm 1, 0)$,
Front/Back: $(0, 0, \pm 1)$.

The resulting octree graph should be interpreted as a bidirectional graph, as every edge gets two labels (Fig.2). Suppose, that two vertices v_i and v_j are connected, $i = 1, 2$ with Cartesian coordinates (x_i, y_i, z_i) are connected. Suppose that $x_1 < x_2$ and that both $y_1 = y_2$ and $z_1 = z_2$. Then the edge e_{ij} gets the label $(1, 0, 0)$ and the edge e_{ji} the label $(-1, 0, 0)$. Note that the sum of the labels belonging to one edge is the zero-vector (0,0,0) in 3D.

4.2. Computation of the SkelTre Skeleton

In the following it is described how the octree graph is retracted to a SkelTre Skeleton by merging suited pairs of neighboring vertices. These pairs are first defined as *V-Pairs* and *E-Pairs* along with the notion of the vertex merging, and secondly the extraction process is explained.

4.2.1. Definitions

The operations on the graph are intuitively explained as the collapse of two edges incident to the same vertex. To quantify the reduction we introduce the vertex dimension *vdim*.

Definition 4.2 The number of distinct associated edge labels of a vertex v_i is called the dimension of a vertex.

Example: In Fig.2, the left outer vertex has dimension 0, the right outer vertex, and the vertex in the second row have dimension 1. The two middle vertices in the upper row have both dimension 2 Note that one vertex have opposite direction labels, and the other vertex has labels of different principal directions.

In 3D, the dimension of a vertex is at most 6. The convergence towards the skeleton and the preservation of the shape part elongation is assured by defining a notion of a local direction. This direction is defined per vertex as vertex direction *vdir*.

Definition 4.3 The sum $vdir(v_i)$ over the distinct associated edge labels of a vertex v_i is called the vertex direction.

Each label in 3D is a 3D vector, which allows adding up the labels. **Example:**In Fig.2 the left 2-connected vertex v_i, with the associated labels $(0, -1, 0)$ and $(1, 0, 0)$ has vertex direction $vdir(v_i) = (1, -1, 0)$. The right two-connected vertex v_j has $vdir(v_j) = (0, 0, 0)$.

The merging of vertices v_i and v_j along a common edge is denoted by $v_i \oplus v_j$. The merged vertex inherits all incident edges from its ancestors v_i and v_j. If v_i and v_j were both incident to a common vertex v_c, then the two edges $v_i v_c$ and $v_j v_c$ are collapsed to a common edge $(v_i \oplus v_j)v_c$. Under ideal conditions these edges represent the connection between two connected subsets of Σ. Therefore, the operation $v_i \oplus v_j$ is only performed on vertices representing two neighboring subsets of Σ with the same direction characteristic, as indicated by the identical edge labels of $v_i v_c$ and $v_j v_c$.

Definition 4.4 Two vertices v_i and v_j both incident to a vertex v_c are called a V-Pair if,

1. the labels of edges $v_i v_c$ and $v_j v_c$ are identical;
2. $\dim(v_i \oplus v_j) \leq \max(\dim(v_i), \dim(v_j))$.

The two vertices (Fig.4) v_i and v_j form a V-pair, because 1) The edge labels $v_i v_c$ and $v_j v_c$ are identical, and 2) $dim(v_i \oplus v_j) = 3$ is smaller than $max(dim(v_i), dim(v_j)) = 4$.

Definition 4.5 A vertex v_i forms an E-Pair with a vertex v_j if:

1. $dim(v_i) \leq dim(v_j)$;
2. $dim(v_i \oplus v_j) \leq \max(dim(v_i), dim(v_j))$;
3. $vdir(v_i) \neq (0, 0, 0)$ and v_i and v_j are connected in direction of one of the non-zero entries of $vdir(v_i)$;
4. v_i and v_j are not a part of an $G(1, 2, 0)$ subgraph.

The definition of E-pairs and V-Pairs find their reasoning in the grid graph property of the octree graph. Because of the underlying octree organization the octree graph is a collection of various connected grid graphs, as introduced in Def. 3.2. The primary goal is to remove the overrepresented graph parts from the graph by merging vertices. These graph parts are mostly G(2,2,0) and G(2,2,2) grid graphs, forming loops.

Three different grid graphs occur locally in the octree graph and in derived, partially collapsed graph's. A G(2,2,2) forming a cubic structure consisting of 12 edges, resulting from eight connected octree cells in cubical order. Note, that a G(2,2,2) subgraph contains 8 possible E-Pair configurations. It requires two of these E-Pairs to reduce the cube to a skeleton. Secondly, a G(2,2,0) forms a loop consisting of 4 edges originating from four connected octree cells, ordered in a square and is not contained in a G(2,2,2). The third possible grid graph, G(1,2,0) contains two vertices connected by an edge belonging to the skeleton, but is not belonging to a G(2,2,0) or G(2,2,2) loop.

4.2.2. Skeleton retraction

The octree graph is retracted by merging vertices forming a V-Pair. If no V-Pair is present in the graph to be retracted, a V-pair is created by an E-Pair. Below it will be shown that each E-Pair results in at least one V-Pair.

Lemma 4.6 The merging of an E-pair results in at least one V-Pair.

Proof Let an octree graph be formed of G(1,2,0), G(2,2,0) and G(2,2,2) subgraphs. A G(1,2,0) is the trivial case belonging to the skeleton. A G(2,2,0) contains 4 valid E-Pair configurations. Consider one arbitrary E-Pair configuration

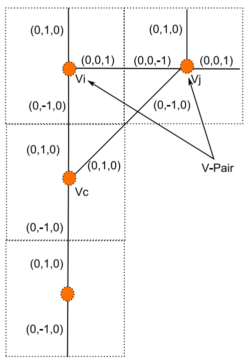

Figure 4: *Example of a V-Pair configuration. The vertices are in orange, the dotted lines denote the cell sides and the labels are shown along the black edges. Note that the edges loose their rectangular configuration during the merging process, while their labels remain.*

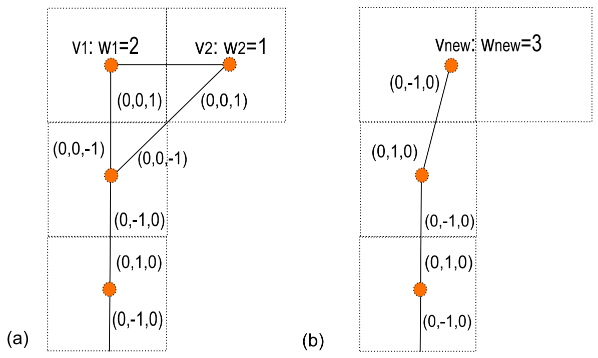

Figure 5: *Example of an embedding. (a) shows a graph before merging is applied to v_1 and v_2 and (b) the graph with the new vertex v_{new}*

of a G(2,2,0). Merging the two vertices involved in the E-Pair, reduces the squared structure to a triangle satisfying the definition of a V-Pair. Every G(2,2,2) subgraph is the union of the vertices and edges of 6 G(2,2,0) subgraphs, each inducing four valid E-Pair configurations. □

Condition 3 in Def. 4.5 ensures convergence toward the skeleton. Vertex dimension 6 results in vertex direction 0 in all cases, therefore, the octree graph is reduced successively from vertices of dimension 5 to 2. This guarantees that first the $G(2,2,2)$ graph parts are reduced, before $G(2,2,0)$ subgraph regions are processed. Another example to depict the algorithm is to characterize vertices of a certain dimension. For example, $dim(v) = 3$ can be either a corner of a $G(2,2,2)$ with $vdir(v) = (\pm1,\pm1,\pm1)$ or a vertex on the boundary of $G(2,2,0)$ subgraph with one entry $\neq 0$ in $vdir(v)$. Therefore, boundary vertices of a $G(2,2,2)$, which have dimension 4, are processed before the 'corner'-vertices of a $G(2,2,2)$ area, and the boundary of a $G(2,2,0)$ region is processed before its 'corners' of $dim(v) = 2$. This assures that the elongation of the object is represented by the final skeleton, and demonstrates the necessity to take the vertex direction into account.

By systematically merging higher dimensional vertices both the amount and the dimension of remaining vertices is reduced. This ensures fast convergence of the algorithm. As vertices, which "over-represent" the point cloud, are merged systematically it is ensured that the final product is a skeleton.

4.3. Graph embedding

The desired centeredness of the skeleton is achieved via a graph embedding. Therefore, the octree graph is embedded into the point cloud by averaging the points Σ_i belonging to an octree-cell. The embedding explained in [PSBM07] was adapted to points clouds, because their embedding can be applied during the computation of the SkelTre Skeleton. Every vertex in the octree graph has a initial weight w which is equal to the number of points belonging to the corresponding octree cell. During the merging process the weighted average of the 3D positions of two merged vertices v_1 and v_2 is taken for the position of the newly created vertex $vnew$. The weight $wnew$ of $vnew$ is then the sum of the previous weights, Fig.5. $wnew = w_1 + w_2$. These weights are used to compute the coordinates of v_{new} into the point cloud. The position of $vnew$ is calculated as: $vnew = \frac{w_1 \cdot v_1 + w_2 \cdot v_2}{w_1 + w_2}$ In case of different levels of subdivision, the subdivision level is multiplied to the weight to obtain a centered skeleton as a result. Embedding can be problematic, if the hull of Σ_i is concave, because then the centroid is not necessarily equal to the weighted average described above. In case of trees this occurs on vertices, where the skeleton branches. Therefore a post-processing step is needed for such cases. The post-processing treats the 3 or more connected vertices of the skeleton, by investigating the distance of a point subset to its bounding box. Let the 6 sides of a bounding box of some Σ_i be the 6 Cartesian directions. And let every point p_i be closest to one of these sides, then the $\{x,y,z\}$-coordinate component closest to a side of the bounding box is selected. The corrected vertex coordinate is computed by averaging selected $\{x,y,z\}$ values for every coordinate component.

5. Implementation

At this stage an implementation of the algorithm is discussed. Successively for the vertex dimensions counting from $n = 5$ to $n = 2$, the following steps have to be implemented.

1. Initialize a vertex-list *VPairList* containing all vertices of dimension n. For each vertex v_i in *VPairList* test if new V-Pairs can be formed with its direct neighbors v_j, until

either a V-Pair is found, or no direct neighbors to test are left. All V-Pairs found are stored in a list.

2. If during the loop through the vertex list *VPairList* no V-Pair was found, go through the vertex list *VPairList* again. Whenever possible, an E-Pair is merged to one or more V-Pairs with the direct neighbors v_j of v_i, until either an E-pair is found, or no direct neighbors to test are left. Each E-pair is merged to one or more V-Pairs which are added to the list of V-Pairs.
3. All vertex pairs in the *VPairList* are merged. Directly after merging it is tested whether the merging resulted in the creation of new V-Pairs. If so, these are added at the end of the V-Pairs list.

```
Input: An Octree graph
Output: SkelTre Skeleton
contains the vertices of the of the processed dimension;
dimList[];
contains found VPairs;
VPairList[];
for dim=5 to 2 do
    $dimList := \{v_i | dim(v_i) \geq dim, i = 0...n\}$;
    forall $v_i \in dimList$ do
        if $Def.4.4 = true$ for some $v_k$ of $v_i$ then
            add found pair to the end of VPairList;
        end
    end
    while $VPairList \neq \emptyset$ and $Def.4.5 = true$ do
        $\{v_i, v_j\} =$ first unprocessed entry in VPairList;
        $v_{new} = v_i \otimes v_j$;
        if ($Def.4.4 = true$ for some $v_k$ of $v_{new}$) then
            add found pair to the end of VPairList;
        end
        if $dim(v_{new}) \geq max(dim(v_i, v_j))$ then
            add $v_{new}$ to the end of dimList;
        end
        remove $\{v_i, v_j\}$ from VPairList;
    end
end
```

Procedure 1 `computeSkeleton`

6. Computational Complexity

This section discusses the computational complexity of the algorithm. In practice, the computation of the skeletons operates on the far smaller set of vertices than the point cloud consists of. Therefore, we focus on the explanation, that the graph-reduction of the SkelTre algorithm is linear in time. A pseudo code to implement the algorithm,procedure *computeSkeleton()*, is shown in Procedure 1. Let v_i be a vertex of the set of vertices V of the processed graph. The dimension of v_i is denoted as $dim(v_i)$ and the number of incident edges as $k(v_i)$. v_j denotes a direct neighboring vertex of v_i and $c_1 \leq 6$ a constant. The procedure *computeSkeleton* contains an outer for-loop, which is bounded by *O(1)*. As can be noticed in Procedure 1, *dimList* is always initialized with *O(V)*. Note that V is decreasing after every dimension. The inner while-loop is operating on a subset of V with at most $\frac{V}{2}$ operations, which results in $O(\frac{V}{2})$ as an upper bound.

The condition $Def.4.5 = true$, selects the first unprocessed entry v_i in *dimList*, that fulfills Def.4.5, and therefore loops through all elements of *dimList*. We show the influence of this condition on the inner while-loop for two boundary cases:

- **Case 1:** The input graph is a skeleton, e.g. a combination of G(2,1,0) subgraphs, all connected on only 2 vertices. This combination will lead to no merge at all; therefore, the whole inner while-loop of *computeSkeleton()* stays *O(V)* by checking one time $Def.4.5 = true$.
- **Case 2:** All $dim(v_i)$ are equal within V, e.g. a graph formed by G(2,1,0) subgraphs, which all are connected on three vertices. This will lead to exactly one check of condition $Def.4.5 = true$ in the given example, and c_2 calls in general. c_2 is bounded by the minimal number of aligned G(2,2,0) subgraphs in one of the principal Cartesian directions.

Now that it is shown that the influence of $Def.4.5 = true$ on the inner while loop is $O(c_2)$ or $O(V)$, the algorithms overall complexity can be calculated as follows from the upper bound. $O(1) \cdot (O(V) + O(\frac{V}{2}) + O(V)) = O(V)$.

7. Results and Practical Validation

The evaluation of the extracted skeletons considers several examples and validation parameters. Fig.6 shows point clouds of 3 trees with different characteristics, all obtained with different terrestrial laser scanners. Tree 1 (4,09m height) has many small gaps and a huge varying sampling density, but less points. It was scanned with a pulse based laser scanner (Leica Scan Station). Tree 2 (2,91m height) and Tree 3 (3,20m height) were scanned with a phase based laser scanner (Z+F Imager 5003). Tree 2 suffers from many occlusions and lots of noise. The inner crown part of the tree is the previously discussed noise example 1. Tree 3 contains extreme curvatures and very small spacing between the branches.

7.1. Skeleton Quality

A minimum octree cell size of 5cm is chosen for evaluating the skeleton quality. No post processing is applied to the skeleton. Centeredness is analyzed by considering the (average) Euclidean distance of every point cloud point to the skeleton. As an indicator of the topology we give the detected number of branches.

7.2. Results

The example of Tree1 shows good connectedness in the 18 detected branches. Even under bad sampling conditions (upper red box in Fig.6). The distances to the skeleton show

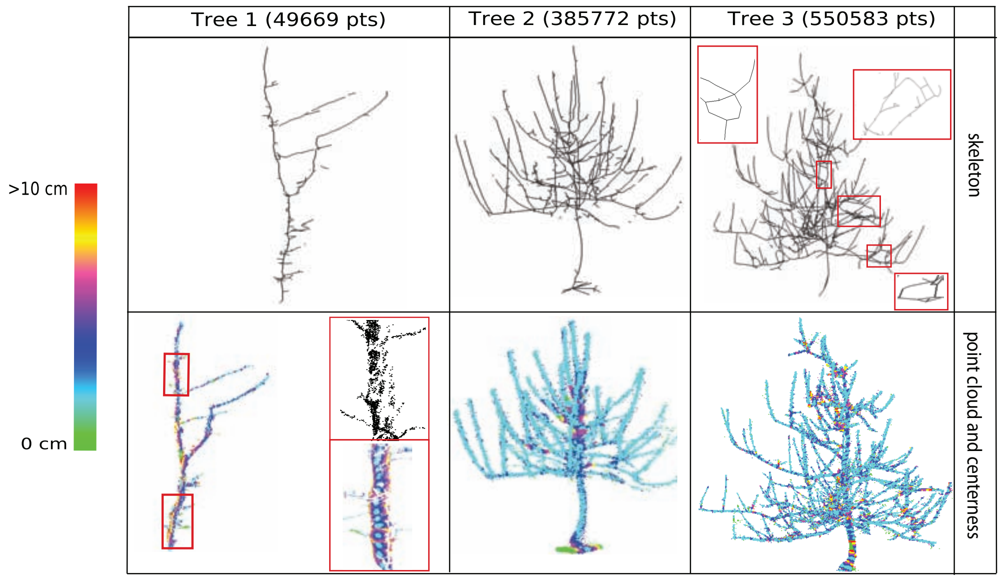

Figure 6: *The 3 test trees. First row shows the computed skeleton. The second row shows the input point cloud colored by the distances of the point cloud points to the skeleton. The black point cloud part belonging to tree one shows strong undersampling due to occlusion.*

a symmetric pattern. Tree1 is a single scan, and therefore not registered from several scans. The skeleton is attracted to one side (light blue color), because the back of the tree was not scanned. The sampling density shows a very noisy pattern already on a single scan. Spurious branches on the boundary indicate the noise on the object boundary. Simple removal of vertices with one incident edge adjacent to a vertex with three ore more incident edges solves this problem in our experience.

The skeleton of Tree2 shows good connectedness on 136 detected branches, but has one erroneous loop caused by crossing branches in the noisy inner part. Further the unidentified branch from Fig.1 was not correctly modeled. The distances to the skeleton around the discussed critical part, where separation of branches was not possible anymore, are around 6cm.

Tree3 (251 detected branches) contains four loops caused by crossing branches and some smaller loops from branches smaller then the predefined minimum cell size. Except for some extreme cases, the resulting distances of point cloud points to the skeleton are in the order of 2-4 cm on the branches. Further a pattern is visible on tree three along the stem and the branches, which we could only explain by the instruments behavior.

7.3. Improvements with respect to a previous method

The SkelTre Skeleton is compared to the CAMPINO skeleton [BL08] on an imperfect point cloud in Fig. 7 and Fig. 8. The example tree contains noise, undersampling, occlusion effects and varying point density in the indicated areas. To give a fair comparison between the two graph reductions the same octree was used as input with an minimum cell size of 0.1m. Fig. 7 shows the improved centeredness as distances to the skeleton between the CAMPINO method on the left and the SkelTre Skeleton on the right. The distances of the SkelTre skeleton never exceed 6cm on the stem, while the CAMPINO methods shows distances up to 13cm. A geometric extraction using the height function would result in a non-centered skeleton, because some branches are parallel to the “ground”. In Fig.8(left) an important characteristic on scanned botanic trees is visible. The points in the strongly undersampled region (a) are in line-like order, which makes it hard to define a suitable inside of the object, as required by some of the algorithms in the related work section. Fig.8(middle) and (right) show two skeletons. The SkelTre Skeleton, Fig.8(middle), is colored with the corresponding edge labels. The skeletons show that the previous algorithm, Fig.8(right), did not maintain the branching structure as good as the SkelTre Skeleton. The example point cloud, Fig.8(left), contains no loop. The SkelTre Skeleton, Fig.8(middle), contains no loops in the resulting graph, while the CAMPINO Skeleton contains 4 loops caused by wrong connections in the octree graph.

Both skeletons in Fig.8 contain two unconnected branch parts because of undersampling and noise, but differ in the number of branches. The CAMPINO Skeleton extracted 33 branches, but only 25 branches are present in the point cloud.

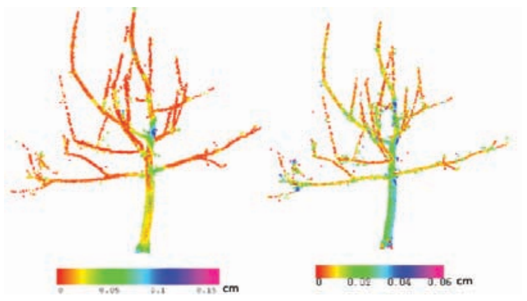

Figure 7: *(Left) Centeredness as distances to the skeleton with CAMPINO (Right) Improved centeredness with the SkelTre Algorithm*

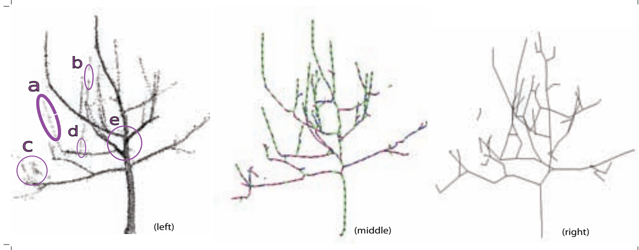

Figure 8: ***(left)*** *The reference tree marked with (a) strongly undersampled region. The branch is already visually not identifiable as a volume. (b) data gap because of occlusion. (c) random noise because of combined effects. (d) combined occlusion and undersampling (e) varying point density* ***(middle)*** *the extracted SkelTre Skeleton with edges colored by their label.* ***(right)*** *a comparable result with [BL08]*

The SkelTre skeleton has 26 branches. One branch too much is caused by the marked region c in Fig.8(left).

The same reference tree Fig.8 was used before in [BL08] and a mean distance to the skeleton of 1,8cm and a maximum distance of 15,6cm was achieved under optimized conditions. The SkelTre Skeleton improved by almost 20% in the mean distance to 1,5 cm. An improvement of almost 33% was obtained in the maximum distance to the skeleton, which reduced to 10,5cm.

8. Conclusion

In this paper the SkelTre skeletonization method has been presented. The method reduces an initial graph, corresponding to the subdivision of a point cloud by a suited octree, to the SkelTre skeleton. It could be shown that the graph reduction is linear in time, which underlines the efficiency of the method that is designed to skeletonise real, large point clouds of botanic trees. Both the correctness and robustness could be demonstrated on several laser scanning point clouds of trees. Further research will first focus on applying the retrieved skeletons on the automatic extraction of structural parameters and sizes of tree parts. The capability of the method to skeletonise point clouds of a much larger class of objects is anticipated, but should be further investigated.

References

[ACK01] AMENTA N., CHOI S., KOLLURI R. K.: The power crust, unions of balls and the medial axis transform. *COMP GEOM-THEOR APPL 19*, 2-3 (2001), 127–153.

[BL08] BUCKSCH A., LINDENBERGH R.: Campino - a skeletonisation method for point cloud processing. *ISPRS J PHOTOGRAMM 63* (2008), 115–127.

[CH88] CHEN H. H., HUANG T. S.: A survey of construction and manipulation of octrees. *COMPUT VISION GRAPH 43*, 3 (1988), 409–431.

[CM07] CORNEA N. D., MIN P.: Curve-skeleton properties, applications, and algorithms. *IEEE T VIS COMPUT GR 13*, 3 (2007), 530–548.

[CMEH*04] COLE-MCLAUGHLIN K., EDELSBRUNNER H., HARER J., NATARAJAN V., PASCUCCI V.: Loops in Reeb graphs of 2-manifolds. *DISCRETE COMPUT GEOM 32*, 2 (2004), 231–244.

[DS06] DEY T. K., SUN J.: Defining and computing curve-skeletons with medial geodesic function. *In Proc.: 4th EG Symp. on Geometry processing* (2006), 143–152.

[Gor06] GORTE B.: Skeletonization of laser-scanned trees in the 3d raster domain. *InProc.: 3DGeoInfo06* (2006).

[GP04] GORTE B., PFEIFER N.: Structuring Laser-Scanned Trees Using 3D Mathematical Morphology. *IAPRS XXXV*, B5 (2004), 929–933.

[PS03] PEMMARAJU S., SKIENA S.: *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Cambridge University Press, 2003.

[PSB*01] PALAGYI K., SORANTIN E., BALOGH E., KUBA A., HALMAI C., ERDOHELYI B., HAUSEGGER K.: A sequential 3D thinning algorithm and its medical applications. *In Proc.: IPMI '01* (2001), 409–415.

[PSBM07] PASCUCCI V., SCORZELLI G., BREMER P.-T., MASCARENHAS A.: Robust on-line computation of Reeb graphs: simplicity and speed. *ACM TOG 26*, 3 (2007), 58.

[Ser82] SERRA J.: *Image analysis and mathematical morphology*. Academic Press, London, 1982.

[VL00] VERROUST A., LAZARUS F.: Extracting skeletal curves from 3d scattered data. *VISUAL COMPUT 16* (2000), 15–25.

[ZKW98] ZHOU Y., KAUFMAN A., W.TOGA A.: Three-dimensional skeleton and centerline generation based on an approximate minimum distance field. *VISUAL COMPUT 14*, 7 (1998), 303–314.