

# RECONSTRUCTION OF LASER-SCANNED TREES USING FILTER OPERATIONS IN THE 3D RASTER DOMAIN

Ben Gorte  
Photogrammetry and Remote Sensing  
Delft University of Technology  
the Netherlands

Daniel Winterhalder  
Institute for Forest Growth (IWW)  
University of Freiburg  
Germany

**KEY WORDS:** Laser scanning, Reconstruction, Forestry, Segmentation

## ABSTRACT:

Analysis of terrestrial laser point clouds in a 3D voxel space has been applied recently during a cooperation between Delft University of Technology and the Institute for Forest Growth (IWW) in Freiburg. The purpose was 3D model reconstruction of trees in a forest. Forest experts can use these, for example to calculate the wood volume and other geometric parameters, allowing to draw conclusions about the timber quality. The reconstruction of the tree crowns is the basis for investigations about the vitality or the competitive status of forest trees. The ability to create explorable three-dimensional models of the forest stands is another goal of this approach.

A crucial phase in the reconstruction process is segmentation of the laser points according to the different branches of the tree. As a result, to each point a label is assigned that is unique for each branch, whereas points are removed (labeled 0) that do not belong to the stem or a significant branch (leaves, twigs, noise).

The problem resembles that of recognizing linear structures is 2- dimensional imagery, such as roads in aerial photography. Therefore we decided to tackle it in a similar manner, i.e. by transferring a variety of image processing operators to the 3D domain, such as erosion, dilation, overlay, skeletonization, distance transform and connected component labeling. Also Dijkstra's shortest-route algorithm plays an important role during the reconstruction.

## 1 INTRODUCTION

The general aim of the project *Development of Algorithms for the Separation of Tree Crowns from Terrestrial Laser Scanner Data*, which is a subproject within NATSCAN — *Laser Scanner Systems for Landscape Inventories* is the reconstruction of leaf trees and coniferous trees from terrestrially captured laser scanner data. Reconstruction of trees means that the diameter of branches and their location (i.e. start and end point) have to be determined. This shall be performed automatically.

A such detailed 3-dimensional model of a single tree could be used e.g. for examinations on correlations of environmental influences and tree morphology. A model consisting of all reconstructed trees of a scanned area can serve as a base for visualizations and virtual reality applications. It is also applicable for analysis of forestal structures and habitats.

Programs for the abovementioned tasks were developed by the Section of Photogrammetry and Remote Sensing, Delft University of Technology. Datasets from trees are provided by the Institute for Forest Growth (IWW), University of Freiburg. One plot (i.e. one location in a forest) is captured by one or more positionings of the laser scanner. The datasets are oriented relative to each other (i.e. registered). The data was captured with Zoller+Fröhlich laser scanners and provided in the form of point clouds (sequence of  $(x, y, z)$ -triples).

In order to acquire complete tree models and to reduce occlusions, the plot is scanned from multiple positionings so that the interesting trees are covered from all around with scan points. Targets are located in the scan area so that the single scan point clouds can be matched together (registered) afterwards. The measurement setup is described in detail in [Simonse et al, 2003] and [Thies et al, 2002].

At the final stage of the reconstruction cylinder fitting is carried out for each of the significant branches: a cylinder is fitted through those points that are at the surface of a particular branch [Pfeifer and Gorte, 2004]. Before this is possible, however, it is necessary to have a rough idea which points belong to that particular branch, in other words to know the initial subset of points

that should be involved in the fitting process. This process itself will be able to identify erroneous points (that do not fit) within the initial subset, and possibly to add points to the subset that are in the vicinity and fit on the same cylinder.

The crucial point in the above is to detect structures in the point cloud which represent a main branch (those that have a significant amount of wood). Besides, points that do not seem to correspond to any main branch must be identified as noise. This is the problem addressed in this paper. We are presenting an algorithm that subdivides (segments) a laser point cloud of a tree into subsets that correspond to the main branches.

The algorithm segments a point cloud according to branches and at the same time identifies how the branches are connected to each other. As a result, a "tree" data structure is constructed, which represents the topology of the branches within a tree.

Basically, the task of the algorithm boils down to detecting elongated, cylindrical structures in a point cloud. However, point cloud segmentation can be considered a difficult subject, especially in presence of the before-mentioned noise, gaps within branches (missing points, e.g. caused by occlusions by other branches) and varying point densities (as a result of varying distance from the scanner). To establish the topology of branches on the basis of point clouds is an unsolved problem as well.

Therefore we decided to perform this part of the analysis in the three-dimensional raster domain.

## 2 RASTER DATA STRUCTURES

It is not uncommon to perform certain steps in processing airborne laser (LIDAR) point clouds in the 2-dimensional grid (image) domain. Quite naturally, a LIDAR point cloud represents a 2.5D surface. When converted to a 2D grid, grid positions are defined by the  $(x, y)$  coordinates of the points, and their  $z$ -coordinates determine the pixel values. Depending on the chosen spatial resolution, i.e. the size of each pixel in the terrain, some accuracy is lost during the conversion, since planimetric  $(x, y)$  positions are not represented exactly anymore. Two additional prob-

lems may occur. At one hand, several points may end up in the same pixel, and the pixel value will have to be a function of the different  $z$ -values (average, median, minimum, maximum, etc.), which implies that some information is lost. At the other hand, there may be grid positions without any laser points. Under the assumption that the surface does not have holes, interpolation is required to fill the empty grid positions.

To the resulting regular-grid DSM, image processing operations can be applied to perform certain analysis functions. Examples are thresholding to distinguish between terrain and e.g. buildings in flat areas, mathematical morphology to filter out vegetation and buildings in hilly terrain, textural feature extraction to distinguish between trees and buildings [Oude Elberink and Maas, 2000], and region growing to identify planar surfaces [Geibel and Stilla, 2000].

## 2.1 From Images to Voxel Spaces

As opposed to the 2.5D airborne case, point clouds obtained by terrestrial laser scanning are truly 3D. It may be argued that a terrestrial laser scanner located at a certain position records distances as a function of 2 angles, which still could be considered a 2.5D result (Fig. 1), but especially when recordings from several positions are combined, there is not a single surface that can be modeled by  $z = f(x, y)$  or by  $d = f(\phi, \theta)$  as there is in the 2.5D case (Fig. 2).

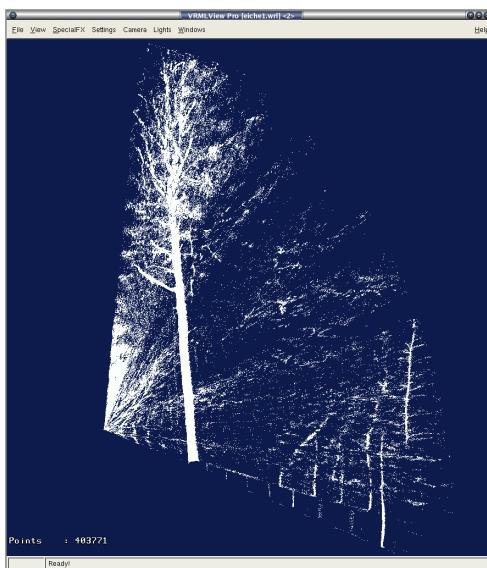


Figure 1: Terrestrial laser scan of a tree recorded from one scan position.

Converting such point clouds to a 2D grid would cause a great loss of information. In this paper, we adopt the alternative approach of converting a 3D point cloud into the 3-dimensional grid domain. The cells in a 3D grid are small cubes called voxels (volume elements, as opposed to pixels or picture elements in 2D). The size of the grid cells determines the resolution of the 3D grid. The 3D grid space can be thought of as a number of images stacked as planes on top of each other, so the position of a voxel is identified by a (plane, line, column) triplet (Fig. 3). During the conversion of a point cloud to a 3D grid, the  $(x, y, z)$  position of a point, together with the chosen resolution, determines the (plane, line, column) position of the corresponding voxel.

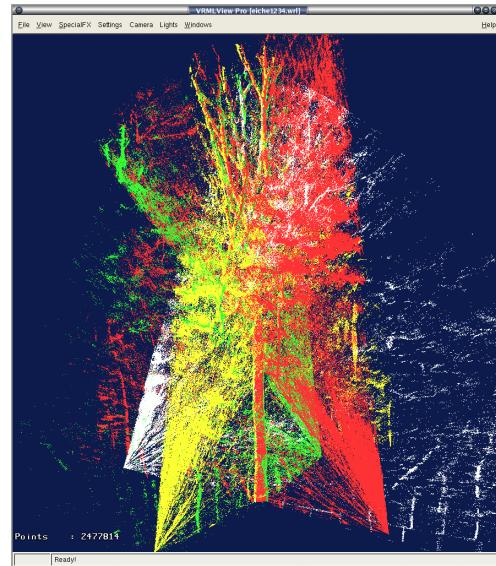


Figure 2: Terrestrial laser scan from 4 scan position combined into a single point cloud.

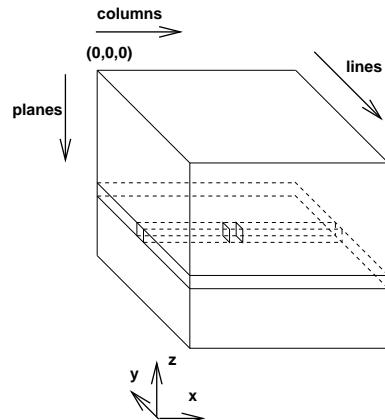


Figure 3: Voxel space with  $(x, y, z)$  and  $(p, l, c)$  coordinate systems.

## 2.2 Voxel Values

The value of the voxel can be defined in different ways. The vast majority of voxels (grid positions) will contain no laser points. One possibility is therefore to assign to those "empty" voxels value 0 and to the other ones value 1. The result is a binary 3D grid, with object and background voxels. Note that in case of laser scanning the object voxels will be the surfaces of the real-world objects. A slightly more advanced scheme is to count the number of points that falls into each grid cell and to assign this number as a voxel value. Semantically, voxel values now denote laser point densities and quite naturally the empty voxels remain 0. If the laser scanner records the intensity of the returned laser beam, then this can also be used as a voxel value. This results in a 3D grey scale image. The value 0 can still be used to represent empty voxels, taking into consideration that surfaces with a reflected intensity of 0 do not exist, or at least will not be recorded. Semantically, however, such a 0-value is different from the other ones, since black is not the same as empty.

### 3 3D RASTER PROCESSING

Similar operations as in 2D image processing can be applied to the 3D voxel spaces. For example thresholding point densities may help to distinguish between "significant" object voxels and those containing noise (Fig. 4).



Figure 4: Tree after rasterization and thresholding on number of points per voxel.

#### 3.1 Neighborhood Operators

Neighborhood operators constitute an important class of 2D image processing operators, including filters (discrete convolutions), morphologic operations (dilation, erosion, opening, closing) and distance transforms. The purpose of this paper is to investigate the use of 3D neighborhood operators in the 3D domain for modeling trees using terrestrial laser scans recorded in a forest. 3D neighborhoods have to be taken into account, which means that filter templates (convolution kernels, morphological structuring elements) become 3 dimensional as well, i.e. consist of planes, lines and columns (Fig. 5).

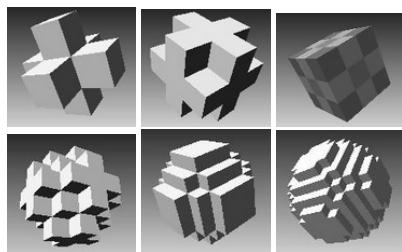


Figure 5: 3D Filter kernels and structuring elements.

Mathematical morphology [Serra, 1985] is a set theory based approach to processing digital raster data. It fits quite naturally to binary images and raster maps, where objects (sets of object pixels) are distinguished from a background. The operations serve to shrink and enlarge objects, suppress 'binary' noise, remove small objects, fill holes and gaps in/between larger objects, count and label objects, recognise object structures, etc. etc. We have applied the 3D extension of morphological *closing* to "fill" the stem and the main branches of the tree with object voxels — after rasterization and thresholding on density these are hollow, since

object voxels are found only at the surfaces (Fig. 6). It should be noted, that gray scale extensions of mathematical morphology have been developed as well.



Figure 6: Cross section through tree stem before and after morphological closing.

The convolution type of filters finds its origin in 1-dimensional signal processing and was extended to the 2D image domain. Whereas a signal can be regarded as a continuous function (such as a voltage as function of time), and an image as a continuous 2D function (gray level as function of (x,y)-position), digital image processing requires some discretization, regarding domain of the function (into pixel positions), as well as its range (into discrete gray levels). The operations, therefore, are suitable for digitally processing gray level imagery.

At a higher level, 2D operations like connected component labeling, distance transform and skeletonization can be extended to 3D. In these cases, the benefit of voxel spaces, compared to the original point cloud, lies in the implicit notion of adjacency in the former.

#### 3.2 Connected Component Labelling

Above, we mentioned the notion of object and background pixels in binary images and their 3D equivalents in voxel spaces. It is common to consider groups of adjacent object pixels as single objects. In other words, any two object pixels belong to the same object if and only if a sequence of object pixels exists that contains these two pixels, such that any two consecutive pixels in the sequence are adjacent. An algorithm that identifies objects in the image and assign labels all to objects pixels that are unique for each object, is commonly referred to as *connected component labelling*.

When determining pixel adjacency in 2D it has to be decided whether only horizontal and vertical neighbors are considered adjacent (4-adjacency), or also diagonal neighbors (8-adjacency). Note that when 4 adjacency is assumed for objects, the background is best considered 8-adjacent, vice versa. The first view regards a chess board as a white background with a lot of small black objects; in the second it is seen as a single black object with a lot of white holes.

In the 3D case, a voxel has 26 neighbors; looking at voxels as cubes that together fill the space, 6 of the neighbors share a face, 12 share an edge and 8 share a corner with the voxel under consideration. This leads to a distinction between 6-adjacency, 18-adjacency (6+12) and 26 adjacency (6+12+8). Which of these views is most appropriate, depends mostly on the resolution with respect to the (expected) size of objects considered. When in a 'poor' resolution the thickness of linear objects is small compared to the size of a voxel, 26-adjacency will be required. Otherwise it may happen that voxels belonging to the same 'real world' object do not belong to the same group. At the same time, 6-adjacency will be assumed for the background. With high resolutions, such that any cross section of objects contains multiple voxels, 6-adjacency for objects (and 26-adjacency for the background) may be considered a 'cleaner' solution, with less probability to create false connections between nearby objects. 18-adjacency is not a good compromise, since it leads to object/background anomalies.

In forest laser scans, we assume that the stem and main branches of a tree together form one object, and that different trees are far enough apart from each other as to belong to different objects, even under 26-adjacency for objects. Although our scans are intended to capture single trees, it cannot be avoided that parts of other trees are recorded at the same time. 3D connected component labelling is therefore applied to distinguish between the trees, and the largest component (with most voxels) is assumed to be the tree under consideration.

### 3.3 Skeletonization

Skeletonization (thinning) in 2D binary images is the reduction of elongated objects to single pixel width. When the objects form a network (such as a road network that has been detected in an aerial photograph), the connections between the objects (*i.e.* the object topology) is preserved. The number of objects is not changed by skeletonization. When both connected component labelling and skeletonization are to be applied, the order in which these operations are performed does not matter.

Different versions of skeletonization exist, according to whether ‘loose’ ends of lines become shorter or not, or whether certain kinds of extremities are preserved or removed. In all cases, 8-adjacency is assumed for objects, and therefore the remaining number of object pixels is minimal under the constraint of topology preservation.

Skeletonization has been a difficult problem in 2D and even more so in 3D, but nowadays various solutions can be found in literature [Borgefors *et al.*, 1999], [Lohou and Bertrand, 2001], [Palágyi *et al.*, 2001], [Telea and Vilanova, 2003].

In 3D, a distinction has to be made between surface and line skeletonization. Both variants iteratively ‘peel’ layers of voxels from the outside of objects until the minimum is reached. Removing one more voxel would create a hole. Surface skeletonization stops when surfaces are formed that cannot be made thinner anymore (when thick walls have become thin foils). Line skeletonization continues until only linear structures remain. In all cases, 26-adjacency is assumed.

Some approaches can be subdivided according to the number of subiterations within each iteration. This number can be 6, 8 or 12, depending on whether a cubic object to be skeletonized would be ‘attacked’ from the faces, corners or edges, respectively. For this project we implemented an 8-subiteration method, described by [Palágyi and Kuba, 1999], which shows excellent performance.

In our application, we apply line skeletonization to reduce the thickness of tree trunks and branches to one voxel. After that, it will be possible to identify different branches and reveal topological relations between them (Fig 7). Note that skeletonization only removes object voxels and never introduces new ones. The skeleton only contains existing object voxels. This explains the importance of filling the ‘hollow’ stem and thick branches (Fig. 6 above). As a result, the skeleton will be located centrally in the interior of these, whereas otherwise it would have to spiral somehow over the surface.

### 3.4 Skeleton Segmentation using Dijkstra’s Algorithm

After skeletonization, object voxels situated in the ‘middle’ of a branch will have only two neighbors within the same branch. Only at junctions, where branches split or smaller branches appear from bigger ones, object voxels may have three or more neighbors. Unfortunately, when two branches are near to each other (compared to the voxel size), ambiguities may arise. Also it is not always possible to exactly distinguish between junction voxels and other voxels by just counting the number of neighbors

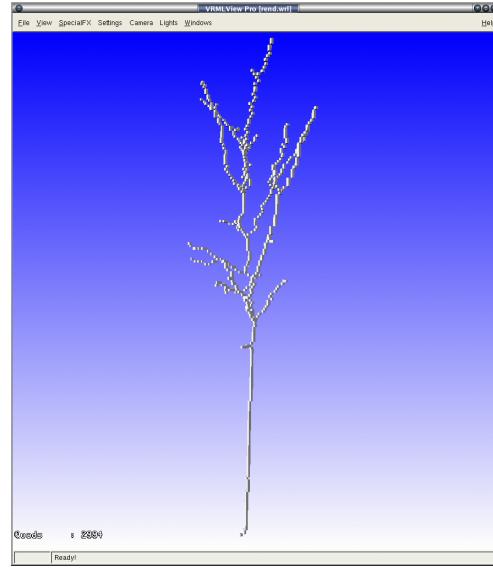


Figure 7: Skeletonization applied to the largest connected component, *i.e.* the main tree in the dataset.

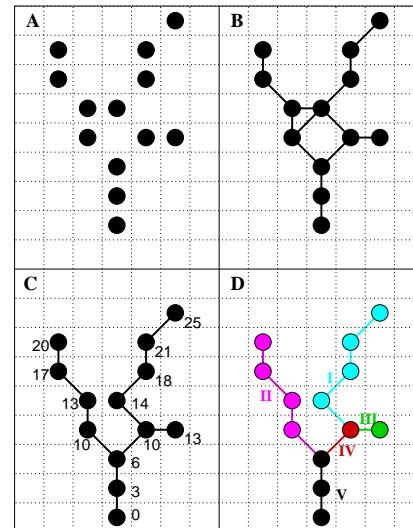


Figure 8: 2D illustration of skeleton segmentation. A. input skeleton; B. graph nodes and edges; C. spanning tree with shortest distances; D. segments and topology.

(Fig. 8.A). Therefore, establishing branch topography requires some extra attention.

As suggested by [Palágyi, 2003] this step is based on Dijkstra’s algorithm, a well-known method for finding shortest routes in a weighted directed graph (representing, for example, a road network).

First, a graph is created. Its nodes are the voxels that participate in the skeleton, and any pair of adjacent voxels generates two arcs between the nodes, one in either direction (see Fig. 8.B, but there each pair of arcs is represented by only one line). Arcs are assigned weights of either 3, 4 or 5, depending on whether the two voxels share a face, an edge or a point respectively [Verwer, 1991].

In this graph, Dijkstra’s algorithms finds for each node the short-

est route to a destination node (the root!). The collection of shortest routes leading from anywhere to the root forms a spanning tree of the graph, which provides a logical model for a tree in a rasterized laser point cloud. Any ambiguities, such as loops caused by falsely-connected branches, are resolved now (Fig. 8.C).

After the algorithm finishes, each node (except the root) knows its parent, the next node along the shortest route to the root. By traversing the tree once more it is easy to find branches and junctions, the latter being nodes (voxels) that appear as a parent more than once. During this traversal each voxel is assigned a unique branch identification number, and the result is a segmented skeleton. Each time a junction is found, a new identification number is given to this voxel and to the other voxels in the same branch closer to the root.(Fig. 8.D, Fig. 9).

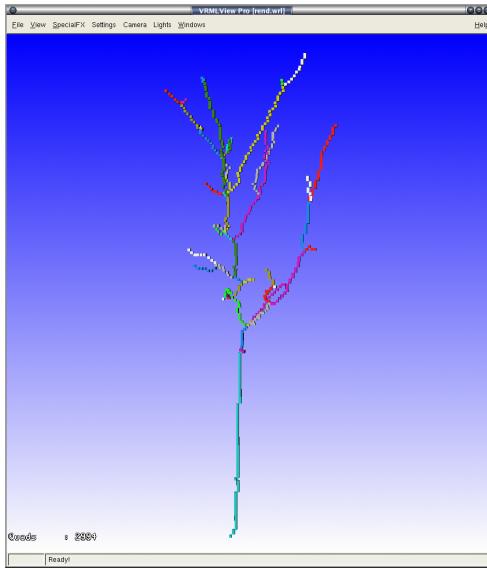


Figure 9: Segmented skeleton: every branch obtains a unique value (branch label).

### 3.5 Distance Transform

After segmenting the skeleton of a tree into different branches, the next step is to segment the remaining voxels of the tree. The method we designed assigns to every non-skeleton voxel the number of the nearest branch in the segmented skeleton. This is implemented by a distance transform, a well-known raster operation that gives to every background pixel (or voxel) the distance to the nearest object. When accepting approximate distances (with less than 2% average error), very efficient algorithms exist [Borgefors, 1996]. A small modification to the algorithm allows to use labeled object pixels and calculate for each background pixel not only the distance to the nearest object pixel, but also record its label (Fig. 10).

### 3.6 Point cloud labelling

As the final segmentation step, branch labels are assigned to the original laser points. This is done by examining for each point which was the corresponding voxel during the initial rasterization. The label that has been assigned to this voxel during the above-described segmentation process, is now assigned to the laser point as well.

Note that a large number of laser points went into voxels that finally did not obtain a branch label: voxels that were removed because of insufficient point density (Section 2.2), but also those



Figure 10: Segmented voxel space with branch labels assigned to all voxels.

belonging to other components (Section 3.2). These voxels are not included in the labeled set (Fig 11).

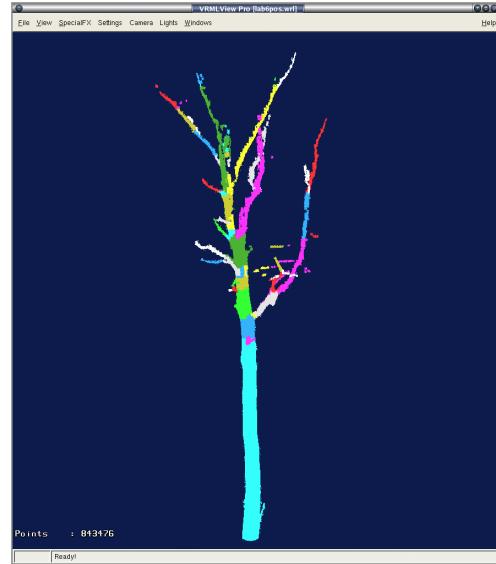
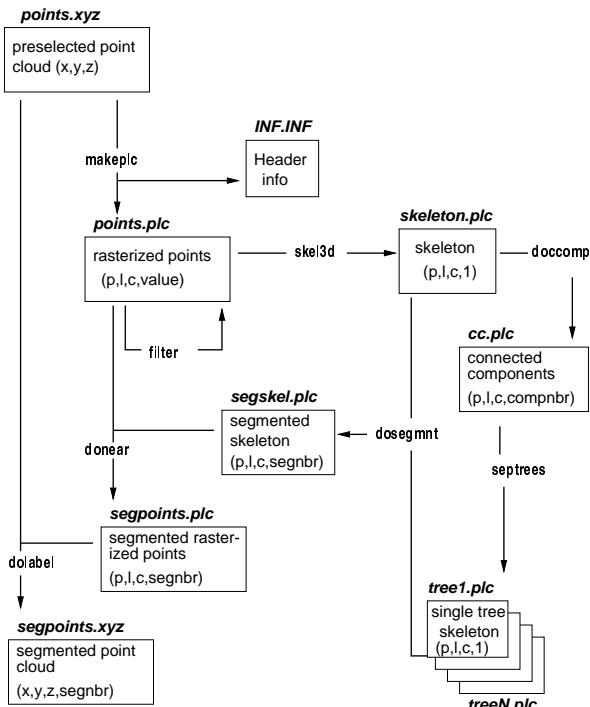


Figure 11: Segmented point cloud with branch labels assigned to laser points and other laser points removed.

## 4 CONCLUSIONS

The above is a description of a pipeline to segment point clouds of terrestrially laser scanned trees into different branches (Fig. 12). It is based on a carefully selected and implemented set of 2D raster and image processing operators into the 3D raster (voxel space) domain. The operators are taken from a rather wide range, but they fit together well and form a coherent system. The resulting segmented point cloud can be easily split into different branches, of which the starting and ending points are now known. The laser points can now be submitted to the next phase in the reconstruction process, *i.e.* cylinder fitting, one branch at a time.



[Serra, 1985] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982.

[Simonse et al, 2003] Simonse, M., T. Aschoff, H. Spiecker, M. Thies. Automatic determination of forest inventory parameters using terrestrial laserscanning. *Proceedings of the ScandLaser Scientific Workshop on Airborne Laser Scanning of Forests*, Umeå, S: 251- 257.

[Telea and Vilanova, 2003] Alexandru Telea and Anna Vilanova, A robust level-set algorithms for centerline extraction, *Joint EUROGRAPHICS-IEEE Symposium on Visualisation (2003)*

[Thies et al, 2002] Thies, M., B. Koch, H. Spiecker: Einsatzmöglichkeiten von Laserscannern für Wald- und Landschaftsinventuren. *AFZ/Der Wald* 57(8) (2002), 395- 397.

[Verwer, 1991] B.Verwer, Local Distances for Distance Transformations in Two and Three Dimensions *Pattern Recognition Letters* Vol.12, No. 11 (1991), 671-682

Figure 12: Raster processing work flow. The arrows represent processing steps, labelled by the names of the corresponding routines.

## REFERENCES

- [Borgefors, 1996] Gunilla Borgefors, On digital distance transforms in three dimensions, *Computer Vision and Image Understanding*, Vol.64 No.3 pp.368-376.
- [Borgefors et al, 1999] Gunilla Borgefors, Ingela Nyström and Gabriella Sanniti Di Baja, Computing skeletons in three dimensions, *Pattern Recognition* 32 (1999) 1225-1236.
- [Geibel and Stilla, 2000] Geibel R. and U. Stilla, Segmentation of laser-altimeter data for building reconstruction: Comparison of different procedures. *IAPRS*, vol. 33, part B3, Amsterdam, 2000, 326-334.
- [Lohou and Bertrand, 2001] Cristophe Lohou and Gilles Bertrand, A new 3D 12-subiteration thinning algorithm based on P-simple points, *IWCIA*, 2001.
- [Oude Elberink and Maas, 2000] Oude Elberink, S. and H-G. Maas, The use of anisotropic height texture measures for the segmentation of airborne laser scanner data. *IAPRS*, vol. 33, part B3, Amsterdam, 2000, pp. 678- 684.
- [Palágyi, 2003] K. Palágyi, J. Tschirren, M. Sonka: Quantitative analysis of intrathoracic airway trees: methods and validation, *LNCS* 2732, Springer, 2003, 222-233.
- [Palágyi et al, 2001] K. Palágyi, E. Sorantin, E. Balogh, A. Kuba, C. Halmai, B. Erdöhelyi and K.Hausegger, A sequential 3D thinning algorithm and its medical applications, in M. F. Insana and R. M. Leahy (Eds.): *IPMI 2001, LNCS* 2082, Springer 2001, pp.409-415.
- [Palágyi and Kuba, 1999] Kálmán Palágyi and Attila Kuba, Directional 3D thinning using 8 subiterations, *LNCS* 1568, Springer, 1999, 325-336.
- [Pfeifer and Gorte, 2004] Norbert Pfeifer and Ben Gorte, Automatic reconstruction of single trees from terrestrial laser, *IAPRS* 35, Istanbul, 2004.