AUTOMATED LARGE SCALE PARAMETER EXTRACTION OF ROAD-SIDE TREES SAMPLED BY A LASER MOBILE MAPPING SYSTEM

R.C. Lindenbergh[a,*], D. Berthold [b], B. Sirmacek[a], M. Herrero-Huerta [a], J. Wang [a], D. Ebersbach [b]

[a] Dept. of Geoscience & Remote Sensing, Delft University of Technology, Delft, The Netherlands
(r.c.lindenbergh, b.sirmacek, m.herrerohuerta, jinhu.wang)@tudelft.nl
[b] LEHMANN + PARTNER GmbH, Schwerborner Straße 1, D-99086 Erfurt - (Berthold,Ebersbach)@lehmann-partner.de

Commission III, WG III/5

KEY WORDS: big data, laser mobile mapping, tree parameters, urban, voxels

ABSTRACT:

In urbanized Western Europe trees are considered an important component of the built-up environment. This also means that there is an increasing demand for tree inventories. Laser mobile mapping systems provide an efficient and accurate way to sample the 3D road surrounding including notable roadside trees. Indeed, at, say, 50 km/h such systems collect point clouds consisting of half a million points per 100m. Method exists that extract tree parameters from relatively small patches of such data, but a remaining challenge is to operationally extract roadside tree parameters at regional level. For this purpose a workflow is presented as follows: The input point clouds are consecutively downsampled, retiled, classified, segmented into individual trees and upsampled to enable automated extraction of tree location, tree height, canopy diameter and trunk diameter at breast height (DBH). The workflow is implemented to work on a laser mobile mapping data set sampling 100 km of road in Sachsen, Germany and is tested on a stretch of road of 7km long. Along this road, the method detected 315 trees that were considered well detected and 56 clusters of tree points were no individual trees could be identified. Using voxels, the data volume could be reduced by about 97 % in a default scenario. Processing the results of this scenario took ˜2500 seconds, corresponding to about 10 km/h, which is getting close to but is still below the acquisition rate which is estimated at 50 km/h.

1. INTRODUCTION

Recent years saw a rapid development of sensor systems that efficiently sample our 3D environment at high detail, (Vosselman and Maas, 2010). Mobile mapping systems implemented in helicopters and cars obtain point clouds consisting of millions to billions of points at a daily basis, (Haala et al., 2008, Puente et al., 2013). In addition, methods from e.g. computer vision and computational geometry became available over the last years that are able to extract useful information from such 3D point clouds by estimating locations and sizes of the different objects sampled by the point clouds, (Puttonen et al., 2011, Rutzinger et al., 2010). Such methods are typically demonstrated in the scientific community at case study scale however. Examples are given of the extraction of geometric parameters from one facade, one or a few trees or 300 m of road furniture, (Monnier et al., 2012). So far, the number of publications that specifically addresses the difficulties of processing large urban point clouds is limited, (Weinmann et al., 2015).

Unfortunately it turns out far from trivial to efficiently exploit the possibilities of combining large 3D point clouds and 3D geometry extraction methods, (Krämer and Senner, 2015). Computers have limited memory capacity, which means that input data needs to be (re)divided into manageable chunks. At the same time an automated division strategy will often affect the objects of interest, as data division will take place before object extraction. Another problem is that the results of geometry extraction algorithms often depend on parameters. For an individual case study such parameters can easily be tuned. When automatically processing an unseen, large data set with a variety of different realizations of the same object type, parameter tuning should be either unnecessary or automated. Another challenge is to make algorithms responsible for the compatibility between consecutive steps: the output of step $(k-1)$ is the input of step k. Even powerful geometry extraction methods may fail when sampling is locally hampered by unfavorable acquisition conditions. Therefore it is important that a human operator responsible for quality control is automatically guided to such locations.

For the workflow presented in this paper the strategy to mitigate the negative effects of large point clouds on computational feasibility, is to process as much as possible at voxel level, rather then at the individual point level. That is, the original point cloud is subdivided into small cubic cells, the voxels, and consecutive information extraction takes place by analyzing local voxel configurations. Switching to voxels has some advantages. Data volume and therefore memory usage is reduced, as many points in one voxel are simply replaced by one single voxel value (Either voxel center, or center of gravity of the points in the voxel); The effect of varying point densities in scan data is largely resolved and irregular points are replaced by regular voxels cells, which makes spatial indexing more efficient; Smaller gaps in data coverage caused by occlusions in the scanning play a smaller role because of the lower resolution of the voxels. Voxels are the most simple way to subdivide a 3D domain. More sophisticated and more scalable is to organize a point cloud in a so-called kd-tree, which allows to prune the point cloud to a certain level, (Triebel et al., 2006), or in an octree.

Even when voxels are processed instead of the original points, files consisting of these voxels can grow arbitrary large if simply merged without strategy. Therefore some additional tiling and stitching strategy is needed that chops data in manageable chunks, either for single node processing or for parallel execution, (Yang and Zhang, 2015). If one such chunk can be processed in a reasonable time, the method would not change if you process 1km, 10km, or 10 000 km of data along urban streets, which is

[*] Corresponding author

exactly what is required in practical applications. Here 'reasonable time' is often defined such that processing the data does not take more time then acquiring the data. So, the only remaining requirement is that it should be demonstrated that processing a chunk doesn't take more time then acquiring it.

The above mentioned advantages of using voxels, kd-trees and octrees and octrees are reasons that several voxel based methods have been designed for natural tree related processing of point clouds. (Popescu and Zhao, 2008) divide their airborne point clouds of trees in voxels to determine the transition between trunk and canopy. (Bucksch et al., 2010) organize notably panoramic scanner point clouds sampling trees in voxels to reconstruct the branch structure of trees; (Wu et al., 2013) organize laser mobile mapping data in voxels to divide tree points over individual trees. (Cabo et al., 2014) detect pole like objects in mobile mapping scan points using voxels, which allowed to work on 20-30 % of the original data volume. (Lim and Suter, 2009) use initial individual point features to group points in so-called super voxels, which are used in consecutively obtaining point cloud classification results of a panoramic scanner point cloud.

This short literature review indicates that in principle methods exist that satisfy the needs of the workflow presented in this paper. Still it was chosen to use methods that were implemented from scratch for the following reasons. Available methods typically solve a problem at hand in an approximate way, in the sense that they may estimate related parameters, but not exactly the parameters required by a particular application. In addition, available methods were often developed while solving a problem within the particular context of a specific data set. Applying the method on different data often gives incorrect results. Note that this problem is partly solved by data processing contests, e.g. (Vallet et al., 2015), which compare methods in a more uniform way. Another issue is that within a workflow back-to-back processing is required: the output of the previous module is the input of the next module. A condition which is typically not fulfilled without extra work when relying on available methods. Finally that a method is published doesn't mean that an implementation is available, nor that input and output formats, programming language and targeted operating system match. For the combination of these reasons it was decided to implement the workflow from scratch using relatively light but flexible methodology.

2. METHODOLOGY

In this work we will present a processing chain aiming at the robust and efficient large scale extraction of tree sizes and locations.

2.1 Data description

The processing chain is initially tested on 7 km of laser mobile mapping data, sampling a test route in Saxony, Germany. The laser mobile mapping system contains a clearance profile scanner from Fraunhofer IPM, collecting up to 2 million points per second. The relative precision of each single laser point is 4 mm. The absolute geographic precision of the point cloud is about 15 cm, ensured by a 2 antenna GPS/GNSS positioning system supported by a 200 Hz IMU from APPLANIX and differential GPS corrections applied in postprocessing.

The mobile mapping system originally records the laser scanner data on the road in a binary format until 1 GByte is reached per file (and then the next 1 GB container will be filled etc.). These binary data containers were converted to xyzi-format and separated into 10 meter chunks. These 10 meter chunks are the starting point of the processing chain described in this work. The test road is represented by 830 xyzi files (the 10 meter chunks) of in total 17 GByte. In single cases, the car had to stop because of traffic which lead to larger files. Together, these 830 files contain 427 186 054 points. The processing chain below doesn't use the intensity data, only the xyz information. In its final form the processing chain will be used to extract tree locations and sizes for 100 km of laser mobile mapping data. For this area, ground truth tree locations obtained by a human operator are available.

2.2 Processing chain

The processing chain consist of the steps listed below. Each step corresponds to an algorithm which allows the user to specify up to two parameters as indicated. Some more details are given below.

Downsampling and retiling, with two parameters: voxel size, and maximum number of points per tile.

Tree point classification, with two parameters: grid size and minimum tree height:

Tree segmentation, with two parameters, minimum canopy diameter and voxel size

Tree parameters, rough, no parameters

Upsampling per tree, no parameters

Tree parameters, fine, no parameters

What follows is a short description of the described methodology.

The input data is provided in chunks that in principal correspond to 10m of road. It is assumed that these chunks are ordered according to the order in which they were sampled A main step to make the processing feasible is actually the first step, the downsampling and retiling step. In this step, input data chunks are read and uniformly downsampled. This means that a fixed voxel size is set by the user, e.g. 30 cm, which defines a 3D grid over the scene sampled by the chunk. Consecutively all original points in a given voxel cell are replaced by the center of gravity of those points. Downsampled points are collected in order until a preset number of points is reached. These, for example 200 000 points are written to a file. Therefore this step will result in files containing at most 200 000 points at a resolution of 30 cm.

These retiled files are the input for the classification step, (Sirmacek and Lindenbergh, 2015). The aim of this step is to divide the 3D points in two classes, tree and non-tree points. In this step a 2D grid is defined over the data of a given grid size, e.g. 20 cm. For each grid cell, the number of 3D points are counted whose xy location belong to the grid cell. Local maxima in the resulting counts are assumed to correspond to tree locations. 3D points above the ground and close to these tree locations will be assigned to the tree class, the other points to the non-tree class. To distinguish trees from shrubs, a minimal tree height can be defined by the user, for example 2 m. A disadvantage of the current algorithm is that it sometimes generates false positives at street poles. This step is also illustrated in Figure 1. In the left image, points colored gray are non-tree points, while the dark green points are classified as trees.

The classified points proceed to the tree segmentation step. Goal of this step is to segment the tree points into individual trees. To do so, tree points are again distributed over a 3D voxel structure. First local maxima are detected. If a local maximum is sufficiently prominent, according to the minimum canopy width

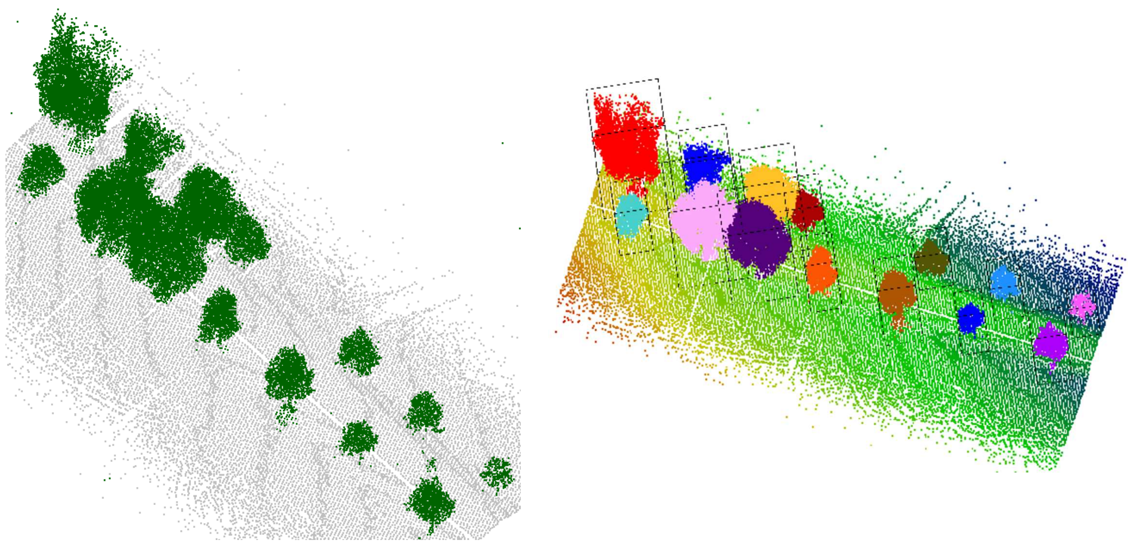

Figure 1: **Left** Points classified as trees (green points) and other (gray points). **Right** Identified individual trees (solid colors) in their bounding box.

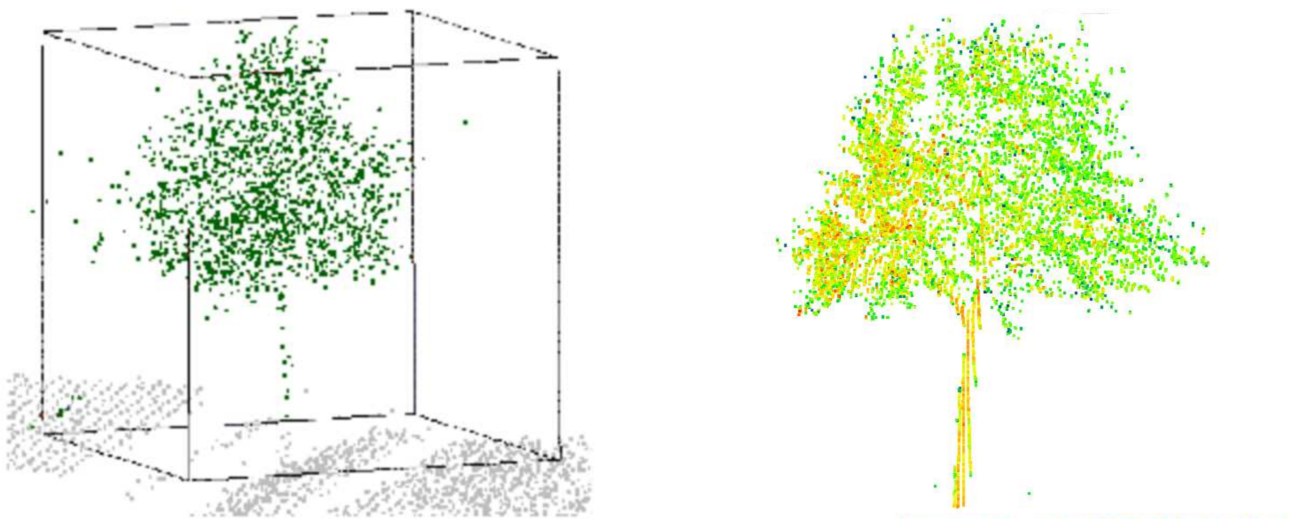

Figure 2: Restoring of the original tree points for one individual tree. **Left**. Voxelized points belonging to a tree (in dark green) and not belonging to this tree (in gray). **Right**. Using the bounding box in the left image, the original points are restored in the right image. These original points are used to determine final tree parameters like Diameter at Breast Height (DBH).

parameter, it is selected as a seed point for growing a tree by traversing the layers of the adjacent octree cells from top to bottom. Voxel cells at a certain layer in between two seed points, are assigned to one of the two seed points according to a proximity criterion. The success of this tree segmentation step is indicated by a quality flag which can be used to guide a human operator to situations were automated processing was not successful.

As a result of the tree segmentation step, tree voxels are assigned to individual trees. By determining the bounding box of all voxels belonging to one tree, a first estimation of the canopy width, tree height and tree location is obtained. Indeed, the canopy width and tree height are simply the width and height of the bounding box, while an estimation for the tree location is the intersection of the horizontal diagonals of the bounding box. This step is illustrated on the right in Figure 1. Different trees have different colors, and around each tree a bounding box is visible.

Note that all of the tree extraction steps above were performed on the uniform downsampled point cloud of, say, 30 cm resolution. As by now individual trees have been identified, it is possible and also computationally feasible to go back to the full resolution original point cloud, as the correspondences between downsampled points and original points were maintained.

The final step is therefore performed per individual tree on the original points, compare Figure 2. In the left image, the downsampled points are shown, in the right image, the original points are restored. The number of down-sampled tree points in this example is 1702, while the original number of points inside the bounding box is 22 283. Using a histogram analysis of the vertical distribution of the tree points, compare also (Popescu and Zhao, 2008), the trunk is separated from the canopy. The direction of the trunk is estimated by Principal Component Analysis (PCA) and the diameter at breast height (DBH), which is the diameter of a tree at 1m30, (Bucksch et al., 2014), is estimated by determining the diameter of a circle through tree points, after

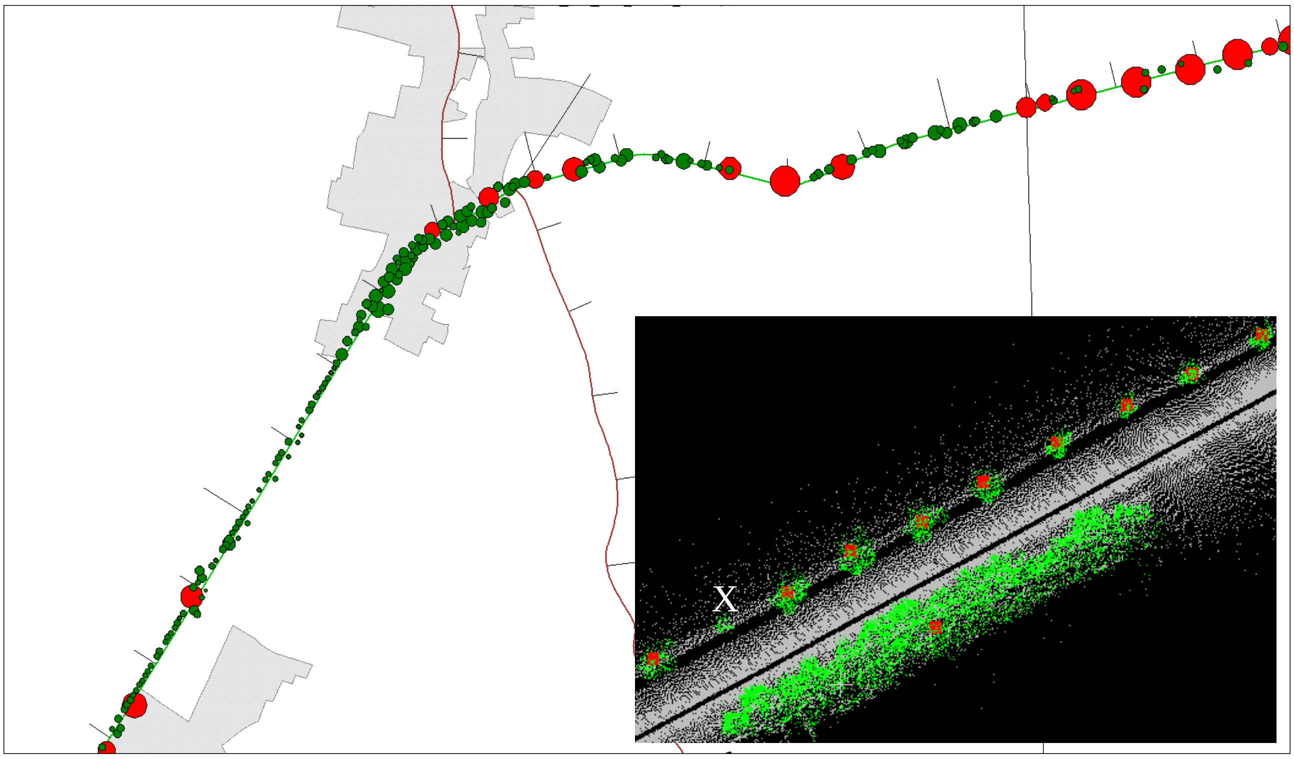

Figure 3: **Large figure:** Identified trees. Green circles correspond to tree locations marked by the method as correct, red circles correspond to tree points where the algorithm was not able to identify individual trees. **Small figure:** some tree classification and segmentation results. Green points were identified as tree points. The red markers indicate single trees (top of the road) or a cluster of trees that couldn't be segmented (bottom of the road).

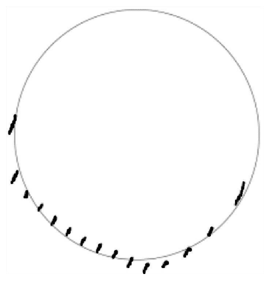

Figure 4: An estimation of the Diameter at Breast Height (DBH) is obtained by fitting a circle to projected trunk points of an appropriate portion of the trunk.

removing outliers using RANSAC, (Fischler and Bolles, 1981), compare also Figure 4. The figure shows only a partially sampled circle, which corresponds to the mobile mapping system seeing only the part of the trunk that is facing the road. The PCA step in combination with restoring the original point cloud can also be used to improve the estimation of the location of the tree by determining the intersection of the principal axis with the terrain.

2.3 Sensitivity analysis

There is an apparent tradeoff between the different parameters. Downsampling with a smaller voxel size in the first step will result in a higher resolution downsampled file, which contains more details, from which consecutive steps could profit. At the same time, if the maximum number of points per tile is kept fixed, the spatial extent of the tile will shrink. This implies that more trees will be cut in two at tile boundaries, which will negatively affect the results.

Increasing the number of points per tile will in turn have a negative effect on the computational performance of the processing chain. In addition, the first three steps, downsampling, classification and tree segmentation, all us a 2D or 3D grid. How the sizes of these grids interact should be further investigated.

Note that in the current workflow the first three steps all use voxels or pixels. In the current implementation the different voxel and pixel sizes are not yet aligned. Therefore it is for example possible that when using a voxel size of 30 cm in the first step, as stated as an example above, and having a grid size of 20 cm in the second step, some grid cells are in fact empty. It would be preferably if these sizes were aligned, as it would further simplify the tuning of the remaining parameters and the interpretation of unexpected results.

3. RESULTS

All above mentioned steps have been implemented, but the final tree parameter estimation step could not yet been tested within the workflow.

As 'default' settings, the values as shown in Table are used. The large image in Figure 3 shows results for part of the 7km of road considered as case study. Green disks indicate trees identified by the processing chain. The diameter of each disk corresponds to the estimated width of the tree canopy. Red disks correspond to tree points where the segmentation method was not able to identify single trees. The algorithm indicates this by a quality flag, which guides an operator to these cases. Again, the diameter of the red disks correspond to the width of the bounding box, in

Step	Parameter	Value
Downsampling	Voxel size	0.3 m
	Max points per tile	20 000
Classification	Grid size	0.2 m
	Min. tree height	2 m
Segmentation	Voxel size	1 m
	Min. canopy diameter 6 m	

Table 1: Default settings workflow

this case of the cluster of tree points that couldn't be segmented. As such cluster in reality typically corresponds to a number of trees, the diameters of the red disks are in general larger than those of the green disks. In total 315 trees were identified using default settings. For an additional 58 tree point clusters the tree segmentation step didn't succeed in identifying individual trees. Note however, that the locations of the identified trees currently have not yet been validated using the ground truth data.

The small image in Figure 3 illustrates some of the intermediate steps. The results of the classification are indicated in green (tree points) and grey (non tree points). Locations of successfully identified trees at the top of the road are indicated by a red marker. Note that a small tree marked ‘X‘ was not identified, probably because it was to small and therefore removed. The tree segmentation method failed on the large cluster of tree points at the bottom of the road. In this case, the location of the cluster is still reported, but with a quality flag indicating the lack of success. In this case also a human operator would probably have difficulties to separate this cluster into individual trees.

The effect of using voxels on the data volume is shown by the following example. Retiling the first 50 tiles using a voxel size of 30 cm reduced the number of points from 23 282 574 to 518 365 points, which corresponds to a reduction in data volume of 97.8 %.

Increasing the tile size could affect memory usage in a negative way but should decrease tiling effects. The effect of changing the voxel size in the first step is less obvious, as also consecutive steps use voxels. Full evaluation of results of different scenarios, including validation against ground truth, will take place after the voxel sizes have been harmonized throughout the workflow

The tests in Table 1 were run on a desktop PC with Intel Xeon 3.6GHz processor and 16GB of RAM. All algorithms where implemented in C++ and compiled and run on the Ubuntu 14.04 64-bit operating system. Running the scenario with the default settings, corresponding to the first row in Table 1 took 2573 seconds in total, which means that 1 km of data takes about 2573:7 km = 368 seconds. This processing speed corresponds to roughly 10 km an hour. This means that the initial goal of processing at the same rate as the acquisition is close but not yet met, as this probably would mean that the processing rate should be improved by a factor five. Probably this improvement could be reached by a further optimization of the current code.

The total computation time of the presented scenario's is divided over the different steps as follows: Retiling takes approximately 65 %, classification takes 32 % and tree separation 2 %. Change of scenario has no large effect on this division, that is effects are not more then a few per cent. It makes sense that computational efforts drop throughout the workflow. Retiling still considers the full input point clouds and file reading and writing is required. Classification only operates on voxels. Finally, tree separation works on tree voxels only. The final tree parameter estimation step, where the full point clouds within the bounding boxes are restored, were not available for testing yet.

CONCLUSIONS

In paper a processing chain aiming at the extraction of tree locations and tree sizes from laser mobile mapping data is presented. All-though further validation is needed, initial results indicate that the workflow is able to extract individual trees at sufficient quality and at a rate that is approximating the data acquisition rate: processing takes place at a rate of about 10 km/h. For efficient processing, the input is downsampled using voxels to a volume that is below 3 % of the original data volume. Besides validation, still harmonization between the parameters in the different steps is required. Such steps, in combination with code optimization are expected to be sufficient to reach the final goal of automatized estimation of features sampled by mobile mapping at a rate that matches the acquisition speed and at a quality that matches the result of a human operator.

ACKNOWLEDGMENTS

This work is initiated on behalf of the LISt-GmbH, kindly supported by the Saxon State Ministry for Economic Affairs, Labour and Transport, Germany. The research is partly funded by the FP7 project IQmulus (FP7-ICT-2011-318787), aiming at the design and implementation of a high volume fusion and analysis platform for geospatial point clouds, coverages and volumetric data sets.

REFERENCES

Bucksch, A., Lindenbergh, R., Abd Rahman, M. Z. and Menenti, M., 2014. Breast height diameter estimation from high-density airborne LiDAR data. IEEE Geoscience and Remote Sensing Letters 11(6), pp. 1056 – 1060.

Bucksch, A., Lindenbergh, R. and Menenti, M., 2010. SkelTre – Robust skeleton extraction from imperfect point clouds. Visual Computer 26(10), pp. 1283 – 1300.

Cabo, C., Ordonez, C., García-Cortés, S. and Martínez, J., 2014. An algorithm for automatic detection of pole-like street furniture objects from mobile laser scanner point clouds. ISPRS Journal of Photogrammetry and Remote Sensing 87, pp. 47 – 56.

Fischler, A. and Bolles, C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24(6), pp. 381 – 395.

Haala, N., Peter, M., Kremer, J. and Hunter, G., 2008. Mobile LiDAR mapping for 3D point cloud collection in urban areas – a performance test. International Archive of Photogrammetry, Remote Sensing and Spatial Information Sciences 37, pp. 1119 – 1127.

Krämer, M. and Senner, I., 2015. A modular software architecture for processing of big geospatial data in the cloud. Computers & Graphics 49, pp. 69 – 81.

Lim, E. and Suter, D., 2009. 3D terrestrial LIDAR classifications with super-voxels and multi-scale conditional random fields. Computer-Aided Design 41(10), pp. 701–710.

Monnier, F., Vallet, B. and Soheilian, B., 2012. Trees detection from laser point clouds acquired in dense urban areas by a mobile mapping system,. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences 3(3), pp. 245 – 250.

Popescu, S. C. and Zhao, K., 2008. A voxel-based lidar method for estimating crown base height for deciduous and pine trees. Remote Sensing of Environment 112(3), pp. 767 – 781.

Puente, I., Jorge, H. G. ., Martínez-Sánchez, J. and Arias, P., 2013. Review of mobile mapping and surveying technologies. Measurement 46(7), pp. 2127 – 2145.

Puttonen, E., Jaakkola, A., Litkey, P. and Hyyppä, J., 2011. Tree classification with fused mobile laser scanning and hyperspectral data. *Sensors*, 11(5), pp. 51585182. Sensors 11(5), pp. 5158 – 5182.

Rutzinger, M., Pratihast, A., Oude Elberink, S. and Vosselman, G., 2010. Detection and modelling of 3d trees from mobile laser scanning data. ISPRS Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 38(5), pp. 521 – 525.

Sirmacek, B. and Lindenbergh, R., 2015. Automatic classification of trees from laser scanning point clouds. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. II-3/W5.

Triebel, R., Kersting, K. and Burgard, W., 2006. Robust 3D scan point classification using associative markov networks. In: Proceedings of the 2006 IEEE International Conference on Robotics and Automation, Orlando, United States.

Vallet, B., Brédif, M., Serna, A., Marcotegui, B. and Paparoditis, N., 2015. Terramobilita/iqmulus urban point cloud analysis benchmark. Computers & Graphics 49, pp. 126 – 133.

Vosselman, G. and Maas, H. (eds), 2010. Airborne and Terrestrial Laser Scanning. Whittles Publishing.

Weinmann, M., Urban, S., Hinz, S., Jutzi, B. and Mallet, C., 2015. Distinctive 2D and 3D features for automated large-scale scene analysis in urban areas,. Computers & Graphics 49, pp. 47–57.

Wu, B., Yu, B., Yue, W., Shu, S., Tan, W., Hu, C., Huang, Y., Wu, J. and Liu, H., 2013. A voxel-based method for automated identification and morphological parameters estimation of individual street trees from mobile laser scanning data. Remote Sensing 5(2), pp. 584–611.

Yang, J. and Zhang, J., 2015. Parallel performance of typical algorithms in remote sensing-based mapping on a multi-core computer. Photogrammetric Engineering & Remote Sensing 81, pp. 373 – 385.