Deciduous tree reconstruction algorithm based on cylinder fitting from mobile terrestrial laser scanned point clouds

Valeriano Méndez[a,c,d], Joan R. Rosell-Polo[b], Ricardo Sanz[b], Alexandre Escolà[b], Heliodoro Catalán[a]

[a] Department of Applied Mathematics. Polytechnic University of Madrid. Ciudad Universitaria, s/n, 28040 Madrid, Spain.

[b] Department of Agricultural and Forest Engineering – Research Group on AgroICT & Precision Agriculture. University of Lleida. Av. Rovira Roure, 191, 25198 Lleida, Spain.

[c] Corresponding author. Tel.: +34 917 308 355. E-mail: valeriano.mendez@upm.es

[d] Proofs correspondence. Valeriano Méndez. Department of Applied Mathematics. E.T.S. Ingenieros Agrónomos. Polytechnic University of Madrid. Ciudad Universitaria, s/n, 28040 Madrid, Spain.

ABSTRACT

Vector reconstruction of objects from an unstructured point cloud obtained with a LiDAR-based system (light detection and ranging) is one of the most promising methods to build three dimensional models of orchards. The cylinder fitting method for woody structure reconstruction of leafless trees from point clouds obtained with a mobile terrestrial laser scanner (MTLS) has been analysed. The advantage of this method is that it performs reconstruction in a single step. The most time consuming part of the algorithm is generation of the cylinder direction, which must be recalculated at the inclusion of each point in the cylinder. The tree skeleton is obtained at the same time as the cluster of cylinders is formed. The method does not guarantee a unique convergence and the reconstruction parameter values must be carefully chosen. A balanced processing of clusters has also been defined which has proven to be very efficient in terms of processing time by following the hierarchy of branches, predecessors and successors. The algorithm was applied to simulated MTLS of virtual orchard models and to MTLS data of real orchards. The constraints applied in the method have been reviewed to ensure better convergence and simpler use of parameters. The results obtained show a correct reconstruction of the woody structure of the trees and the algorithm runs in linear logarithmic time.

KEYWORDS

Tree reconstruction; cylinder fitting; LiDAR; mobile terrestrial laser scanning; point cloud.

Variable	*Description*
A	Covariance matrix
α	Polar angle used in the iterative method to obtain $\vec{d}$
B	A branch object
B^*	Temporal branch built when a new point is included in the process

BN	A new branch built by the branching process
c	Centroid of a branch
$\vec{d}$	Cylinder direction of a branch
$\overrightarrow{d^*}$	Cylinder direction of a branch estimated by a numerical method
$\Delta\alpha$	Polar angle resolution used in iterative method to obtain $\vec{d}$
$\Delta\varphi$	Azimuthal angle resolution used in iterative method to obtain $\vec{d}$
$\Delta\theta$	Angular resolution of laser
Δy	MTLS longitudinal resolution (distance between vertical scans)
φ	Azimuthal angle used in iterative method to obtain $\vec{d}$
HMT	Hidden Markov tree
k_r	Factor of radius r to determine whether P is aligned in current branch B or allows a new branch BN
l	Distance from the laser sensor to a tree object
M	Directions to the centroid matrix
N	Number of points in the point cloud
n	Number of points in a branch or cylinder
n_b	Number of branches
$n_{\min}$	Minimum number of points used to determine the significant parent or predecessor branch
n_p	Number of points of the considered parent or predecessor branch
n_s	Number of points that freely seed a cylinder when the building

	of a new branch starts
O	An upper limit of growth of the algorithm response time
ord	Branching order according to the terminology proposed by De Reffye et al. (1988)
ord_C	Order of the checked parent or predecessor branch used to determine the significant parent or predecessor branch
$ord_{\min 1}$, $ord_{\min 2}$	Rank of order used to determine the significant parent or predecessor branch
P	An individual point of the point cloud
P_1	Initial point of the cylinder axis that models a branch
P_2	Final point of the cylinder axis that models a branch
P_d	Projection of P over the cylinder axis in a branch
P_r	Initial point, placed at the base of the trunk, taken as origin of the tree model reconstruction.
θ	Angular position of laser beam
r	Radius of the cylinder that models a branch
t_2	Value of parameter t for P_2 in a vector straight equation defined by P_1 and $\vec{d}$
t_d	Value of parameter t for P_d in a vector equation of a line defined by P_1 and $\vec{d}$
y	MTLS longitudinal position
z_0	Height of the laser sensor

1.0 INTRODUCTION

Geometric reconstruction can be used to obtain a detailed structural analysis of trees. The aim is to derive vegetative parameters such as leaf area, canopy volume or woody volume from massive data point clouds. Direct use of raster information, e.g. a photograph, can be used to obtain any of these parameters (Phattaralerphong & Sinoquet, 2007). Reconstruction of tree geometry supports the implementation of virtual tree models, such as use of the statistical framework of the hidden Markov tree (HMT) model introduced by Crouse et al. (1998) and used for constructing realistic apple trees by Durand et al. (2005) and Fee et al. (2008).

In parallel with the use of massive data from photogrammetry or aerial scanning for the detection of trees and estimation of their general parameters, two main approaches are used to study their geometry at individual tree level. The first is based on digital photographs (Shlyakhter et al. 2001; Mizoue & Masutani, 2003; Phattaralerphong & Sinoquet, 2005 and 2007, Tan et al., 2008;): graphic data are processed to determine the existence of vegetation and sensor parameters (camera height and its horizontal distance to the tree) allow a projection to be obtained on a voxel space, with which the tree-top and leaf area can be estimated (Phattaralerphong & Sinoquet, 2007). The use of a reduced voxel size to improve accuracy dramatically increases the processing time.

The second approach uses mobile terrestrial laser scanning (MTLS) to obtain a dense point cloud from which a detailed geometrical description can be extracted (Rosell et al. 2009 and Sanz-Cortiella et al. 2011). Simonse et al. (2003) detected woody geometry from MTLS data using the Hough transform and Gorte and Winterhalder (2004) as well as Pfeifer et al. (2004) created a topology skeleton from a voxel space. The use of TIN (triangulated irregular network) to obtain geometric information about woody tree structure is limited by stem capillarity (Fig. 1) and usually supports extraction of neighbourhood graphs (adjacency relations between all the points). Pfeifer et al. (2004) obtained a model of major branches and stems with cylinder fitting. Other methods, which combine scanning data with texture information from high resolution photographs, have been proposed by Reulke and Haala (2005). Iterative closest point (ICP) algorithms have also been used to fit the guide lines obtained in different scans (Besl & McKay, 1992; Henning & Radtke, 2006). The algorithm iteratively revises the geometric transformation needed to minimise the distance between the points of the different raw scans.

It is easy to determine whether a point of the MTLS point cloud belongs to the trunk and main branches. However in the lowest branches, particularly the stems, it becomes more difficult to determine whether a point of the cloud belongs to one stem or another. Neighbourhood graphs, geodesic graphs and several clustering algorithms can be used to obtain the skeleton of the tree and the radius of each branch. The search of points to build neighbourhood graphs is based on kd-tree, a k-dimensional binary tree generated by hyperplane splitting that divides the space in two half-spaces. Verroust and Lazarus (2000) generated the skeleton of a tree from a set of neighbour graphs, geodesic graphs (selecting an initial point at the base of the trunk, P_r, and the shortest path from each point to P_r) and k-levels (defined by Lloyd (1982) which divide the graph into clusters of close points). From a kd-tree, Yan et al. (2009) applied the Lloyd iteration (1982) to obtain a segmentation of the cloud in clusters based on cylinders. Delagrange and Rochon (2011) used the model of Verroust and Lazarus (2000) to obtain the skeleton and select centroids within it. They then applied a clustering process to connect each point to their respective branch. The vector reconstruction method proposed by Verroust and Lazarus (2000) or Delagrange and Rochon (2011) requires executing the process in stages: neighbourhood graph, geodesic graph, skeleton extraction, skeleton population with adjacent points clusters and, finally, fitting each cluster with a surface. Preuksakarn et al. (2010) use a space colonisation algorithm (SCA) as a function of clustering. De Aguiar et al., 2008a and 2008b, use clustering processes to capture shapes from video data.

In this work, the approach proposed by Pfeifer et al. (2004) is used as a direct algorithm for woody structure reconstruction. One of the objectives was to minimize the number of parameters that control the operation of the algorithm. The existence of a large number of empirical parameters controlling the process can distort the method and make it more difficult to attain the desired unique solution. The developed algorithm was applied to point clouds obtained from MTLS measurements of real orchards and point clouds obtained from simulated MTLS measurements of virtual orchards built with SIMLIDAR software (Mendez et al. 2012 and 2013), respectively. Models of woody trees with a high degree of branching, applicable to deciduous leaf species, were used. Simulations with varying degrees of scanning density were also tested.

The information provided by this algorithm could be useful for the modelling of orchards and their evolution from both a scientific and commercial perspective. Using MTLS of trees and subsequently obtaining and quantifying the woody structure with the proposed algorithm at the beginning of the season can help growers and/or advisors to:

- improve the determination of seasonal foliage evolution by subtracting the woody model from the MTLS point clouds obtained during the season.

Knowing the leaf area is very useful in terms of plant protection products dosage and canopy management in general.

- decide on pruning intensity by comparing the woody model obtained at the end of the season with the one obtained at the end of the previous season. Additionally, scanning the trees before and after pruning can help growers see the potential effect of pruning intensity on the next season's production.
- check whether tree growth is correct in terms of its evolution over the seasons and in terms of its training system.
- estimate total volume of the ligneous fraction of the tree orchard and its evolution over the years, constituting a novel approach for other agricultural research purposes.

2.0 MATERIALS AND METHODS

2.1 Data

The proposed algorithm was applied to real MTLS data from a pear orchard and to simulated MTLS data of an apple orchard and a vineyard virtually obtained with SIMLIDAR software (Fig. 2-a and Fig. 2-b).

The real MTLS operation was performed on a cv. Blanquilla pear orchard (*Pyrus communis* L. 'Blanquilla') after leaf-fall (see Fig. 2-c). A Fiatagri 80-76 DT tractor model was used at a forward speed of 1 km h^{-1}. The sensor was placed at a height of 2.10 m, angular resolution ($\Delta\theta$) was set to 1° and longitudinal resolution was 15 mm (distance between vertical scans).

The simulated MTLS operation was applied to a virtual apple orchard obtained with SIMLIDAR software (Méndez et al. 2013), based on a HMT modelling process (Durand et al. 2005) and to a virtual vineyard based on A SIMLIDAR generated growth pattern. A simulated monolateral MTLS using SIMLIDAR (Méndez et al. 2012, 2013) was applied to both virtualisations with an angular resolution of 0.5º and a longitudinal resolution of 10 mm.

2.2 Algorithm

The algorithm was developed in Microsoft ® Visual C++ and run on a PC (HP ® Compaq dc 7700p, Intel(R) Core(TM)2 CPU 6600, 2.40GHz, 3.49GB RAM with a Windows ® XP Professional operating system).

MTLS provides distances (l) from the sensor to each tree object, at a given vehicle longitudinal advance position (y) and at an angular value of the sensor's emitted beam direction (θ). For each scan, the acquisition system stores the triplet $\left(y_i \quad \theta_i \quad l_i\right)$ with $i = 1 \cdots N$ (where N is the total number of measurements). From a set of $\left(y_i \; \theta_i \; l_i\right)$ and knowing the longitudinal advance increment (Δy), the angular resolution ($\Delta\theta$) and the height of the sensor (z_0), it is possible to obtain the 3D coordinates $\left(x_i \quad y_i \quad z_i\right)$ of each intercepted point of the tree. By using a global navigation satellite system (GNSS) to determine the sensor position for each scan, it is possible to obtain the absolute coordinates for each point in the point cloud.

Although a lateral MTLS intercepts all the geometrical data of an orchard, its operation is optimum in a sparsely populated structure, as is the case with agricultural deciduous species. When using a bilateral or multilateral scanner, the problem of measurement errors increases significantly, with a dead-reckoning system for the accumulated errors not being possible (Nebot and Durrant-Whyte, 1999; Guivant et al. 2002; Neira et al. 2003). In this case it is essential to use reference points, or guidance systems based on a SLAM algorithm (simultaneous localisation and mapping, Iagnemma et al. 2004, Auat Cheein & Guivant 2014) to statistically estimate the dragged errors.

The work starts with an unstructured point cloud, with all the inner points consistent after a debugging process. The "cylinder following" method proposed by Pfeifer et al. (2004) aims to build the skeleton, simultaneously populating the cylinders with adjacent points, without using a prior neighbourhood or geodesic graph. It is based on constructing a cylinder that fits the trunk of the tree and a cylinder vector structure, which extends upwards and outwards, that is fitted through all the points of the cloud to obtain a populated skeleton that is the woody structure.

2.3 Setting cylinder direction

Setting the direction of the cylinder requires determining the cylinder which best fits a set of points. Given a set of points $S = \{P_i = (x_i \quad y_i \quad z_i)\}$ with $i = 1, \cdots, n$, with n being the number of points of the cylinder, the cylinder trunk that best fits *S* will have an axis that goes through the centroid c of *S*, with $c = (\bar{x} \quad \bar{y} \quad \bar{z}) = \frac{1}{n}\sum_{i=1}^{n}(x_i \quad y_i \quad z_i)$. If the cylinder axial direction $\vec{d} = (d_x \quad d_y \quad d_z)$ is the direction that minimises the maximum of orthogonal distances $(P_i, \quad \vec{d})$, it is possible to obtain $\vec{d}$ with an iterative method (Rabbani & Heuvel, 2005), taking directions with angles $(\alpha \quad \varphi)$ with

$0 \le \alpha \le \pi$, $0 \le \varphi \le 2\pi$ and successively changing $\Delta\alpha$ and $\Delta\varphi$ until finding where the orthogonal distances are minimum.

It is also possible to obtain $\vec{d}$ as a non-linear least-squares estimate (Lukács et al., 1998, Marshall et al., 2001) as an eigenvector of a covariance matrix $A = M^t M$, where the i^{th} row of M is $p_i - c$, that is:

$$M = \begin{pmatrix} x_1 - \bar{x} & y_1 - \bar{y} & z_1 - \bar{z} \\ x_2 - \bar{x} & y_2 - \bar{y} & z_2 - \bar{z} \\ \vdots & \vdots & \vdots \\ x_N - \bar{x} & y_N - \bar{y} & z_N - \bar{z} \end{pmatrix}$$

The matrix A has a maximum of three eigenvectors that fit three cylindrical adjustments to the point cloud, taking the best direction as the one related to the lowest eigenvalue. The eigenvalues and eigenvectors can be calculated using the Rayleigh-Ritz ratio.

2.4 Branching criterion

The algorithm, shown in Table 1, starts by selecting an initial point at the base of the trunk (P_r), with the condition that P_r has a minimum value in *z*. The method continues in Table 2 to search for points close to P_r setting a cylinder that fits the trunk, usually with the direction $\vec{d} \approx (0 \quad 0 \quad 1)$. Optionally, the points search can be supported in a kd-tree to improve processing time. Those points, close to the initial cylinder and aligned with their current direction, can be considered as a continuation of the trunk, otherwise they will be considered the origin of a new branch. The setting of the direction $\vec{d}$ in the starting stage of a new branch is the main weakness of the algorithm. The direction of the trunk, once P_r has been selected, does not emerge immediately from the first clustering of points close to P_r. It is necessary to seed the cylinder with a number of close points (n_s), without checking the alignment ratio of each one with respect to the parameters of the cylinder ($\vec{d}, P_1, P_2, r$). The parameter n_s is applicable to the initial trunk and to all new branches to be reconstructed in the model. The parameter n_s must be selected considering the scanning density used to obtain the point cloud and the branching order following the biological terminology of De Reffye et al. (1988). The density of points in the cloud depends on the values of Δy and $\Delta\theta$ adopted in the MTLS operation; the greater the density, the greater n_s. The value of n_s decreases as branch order increases in the model, which implies a decrease in the radius and the density of scanned points.

In a ligneous structure, the radii of successor branches are smaller than that of their parent. This property is used as a constraint in the model. This restriction has the advantage of reducing the need to find a value of n_s only for the formation of the main trunk, but the behaviour is correct only in the major branches, where the order is low. For higher orders, reconstruction becomes an unrealistic capillary-like structure as all dependent cylinders are forced to have a smaller radius. As an intermediate alternative, in Table 2, a restriction has been used so that the branches have a radius smaller than a predecessor branch which can be considered significant. A branch is considered significant if it meets one of the following two conditions:

$$\begin{array}{l} ord_C < ord_{\min 1} \\ ord_C < ord_{\min 2} \bigcup \ n_P > n_{\min} \end{array}$$

where ord_C is the order of the verified parent branch, $ord_{\min 1}$ and $ord_{\min 2}$ are parameters with values of the order of the parent branch, n_P is the number of points of one of the predecessor branches of the branch under construction and $n_{\min}$ is the minimum number of points that the branch should have to consider it significant. The data model of the branch class used (CBranch) has the properties shown in Fig. 3.

Each branch, except the trunk, has a pointer to the predecessor or parent branch and from zero to N successor branches. In the data structure, a pointer to the predecessor is provided, the value of which should be null for the main trunk which is the branch of order 1. The successor branches, if any, are stored in an array of pointers. A significant branch is selected by moving back recursively in the predecessor hierarchy of a given branch, through the pointers of parents of following branches, searching for the predecessor that fulfils the minimum order ($ord_C < ord_{\min 2}$) and a minimum number of points ($n_P > n_{\min}$). If this condition is not met in the hierarchy of predecessors, then the first branch that meets the condition $ord_C < ord_{\min 1}$, with $ord_{\min 1} < ord_{\min 2}$, is considered significant.

Determining if a point P is aligned with the current branch and may be incorporated to a branch B or whether it is necessary to start the building of a new branch (BN) is a process that depends on the characteristics of the cylinder B ($\vec{d}, P_1, P_2, r$) and on the characteristics of B^*, with $B^* = B \cup P$. The cylinder generated by B^* is characterised by $\overrightarrow{d^*}$, P_1^*, P_2^*, r^*. If $r^* > k_r * r$ with $k_r > 1$, then it is considered that the point does not align and a new branch BN is started. The value of k_r depends on the position of the point P when it is projected on the branch. If P_d is the projection on the straight line defined by P_1 and $\vec{d}$, then it will be true that $P_d = P_1 + t_d * \vec{d}$. Furthermore, as P_2 is selected so that $P_2 = P_1 + t_2 * \vec{d}$, where $t_2 > 0$, it results that $P_1 < P_2$. Therefore, depending on the position of P_d (or the value of t_d), different values of k_r may be taken.

2.5 Clustering

The algorithm can make the mistake of considering that P generates a new branch BN when it is actually a mere bulge of B. In addition, from this mistaken new branch BN, a thread is reconstructed that actually belongs to the predecessor branch. The multithreading problem is solved with two alternative clustering processes. The first process, shown in Table 3, detects successor branches of one predecessor with a similar direction $\vec{d}$ between them and merges them all. The second process, shown in Table 4, detects a predecessor branch and one successor branch that must also be a continuation of each other and forms a single cylinder.

Finally a balanced clustering process, also following the hierarchy between each branch and its successors, is adopted as shown in Table 5. It is considered that the tree structure must be optimal, in other words that its main geometric parameters must be minimum. Then the points between a predecessor (B) and successor (BN) branch must be distributed minimising their volume. Calculating the volume of a current branch, knowing $\vec{d}$, P_1, P_2, r, is a direct operation without additional processing time cost. The clustering process is done by comparing each branch with its successor, which requires less time than comparing each branch with all the rest.

3 RESULTS AND DISCUSSION

Both methods, iterative and least-squared estimate, were compared in a test by generating 100 random directions $\vec{d}$ and, from each direction, an unstructured point cloud. The results, showing both processing time and accuracy, are shown in Table 6.

The reconstruction of the pear tree model required the lowest values of k_r and n_s since the generated point cloud was less dense. In the case of the vine, values of k_r and n_s were smaller than those required for the apple tree since the virtual model had higher ligneous shoot density (Table 7). The number of reconstructed branches and the processing time are shown in Table 8. The reconstruction process, by steps, is shown in Fig. 2.

In the virtual apple tree, the process starts with a sapling which gives rise to the trunk of order 1 and, in the subsequent growth iterations when a branch occurs an order is added to it. The reconstruction process (superposition of branches in the virtual model together with the operation of the MTLS) resulted in over branching of the tree pattern when compared with the original virtual model. Total branch volume was over-estimated, especially in the apple tree reconstruction. As the volume is $h\pi r^2$, the error in the radius must be the square root of the error in the volume. In other words, in the initial point cloud, the belonging of a point to a cluster and the cluster hierarchy may have a higher probability than indicated by the initial model. There are also model limitations with respect to the adopted parameters (Table 7). Parameter t_d has a stable value, the value of n_s is more dependent on the density of scan process. It is required an easy try to verify that the trunk is generated in one cylinder. The algorithm has the advantage that the control of radius with the parent branches is a self-tuning approach.

The lack of accuracy, the reconstructed model is not equal to the SIMLIDAR virtual model, is due to the lack of convergence of the method, defects in the virtual model and the effects derived from the scanning operation. The simulated MTLS operation can generate shadow effects which are aggravated if two branches in the virtual model are superimposed. These shadow effects may cause the reconstruction of a branch to bifurcate to a branch that is, in reality, a continuation of a different branch of the model.

A wrong choice of input parameters can result in an unrealistic reconstruction. Figure 4 shows three examples where incorrect parameter selection led to a poor reconstruction. If k_r is given a large value (Fig. 4-a, with $k_r = 1.22$ and $t_d < 0.9$) branches thinner than normal are obtained, despite the limitations imposed by the constraint that the radius of a branch cannot be greater than its predecessor branch. By taking a value that prevents trunk branching (Fig. 4-b, with $k_r \geq 1.23$ and $t_d < 0.9$), a single unrealistic cylinder is obtained which contains all the points in the point cloud. Choosing a low value of n_s (Fig 4-c, with $n_s = 4$) also results in a poor reconstruction with excessive branching. Based on the De Reffye et al. (1988) branching order, chains of small branches are created resulting in a maximum order in the model much higher than actually exists (in Fig 4-c the maximum order is about 35).

It has been estimated that the cost of the algorithm is $O(N \cdot \log(N) \cdot \log(n_b))$, being O an upper limit of growth of the algorithm response time with the increase of N, the total number of points in the point cloud, and n_b the total number of branches. The main cost of the algorithm is located in the main process (Table 1, lines 4-16), where the iteration is executed N times. Moreover, the FindTheClosestPoint function (Table 2, lines 3-13) function has a cost of $O(\log(N) \cdot \log(n_b))$. For nearby points in kd-tree it has a cost of $O(\log(N))$ (Cormen et al. 2009). Together with the estimation of $O(\log(n_b))$ to check that the point is not closer to the other branches of the model (Table 2, line 7; costing $O(n_b)$, but underestimated as a result of line 6). Additionally, the cost to build a kd-tree (Table 1, line 1) is also $O(N \cdot \log(N))$ (Cormen et al. 2009). The AlignedChildrenBranches procedure, which is called in line 17 (Table 1), has a cost of $O(n_b)$, with n_b being the total number of branches; the main cost is in iteration I (Table 3, line 1) because the cost of the rest of iterations (depending on the number of children of the branch) is small and does not increase with n_b. In line 18 (Table 1) ConnectAlignedBranches is called, with a cost of $O(n_b)$ located in iteration I (Table 4, line 1); the times this function is called is reduced, having an estimated cost of $O(n_b \cdot \log(n_b))$. Finally, in the Clustering function (Table 5) iteration I (line 1) is performed n_b times, while for iteration K (line 10) the average number of points in a branch can be estimated as $\frac{N}{n_b}$, resulting in a cost of $O\left(n_b \cdot 2 \cdot \frac{N}{n_b}\right) = O(N)$ which includes, as before, the cost to call it in the main function, $O(N \cdot \log(n_b))$. To summarize, by adding all the above results (Table 9, lines 1-6) and considering that the order of magnitude of n_b is lower than N, the proposed algorithm is

$O(N \cdot \log(N) \cdot \log(n_b))$. That is, in the worst case, the computational cost increases in a linear logarithmic order according to the number of points in the cloud.

CONCLUSIONS

Individual tree reconstruction is feasible with a short processing time cost using the proposed algorithm. The disadvantage of the algorithm is the absence of a unique convergence. It is important to correctly adjust the values of the input parameters, in general depending on the MTLS point cloud density. The main parameters are the number of free seed points (n_s) and the radius factor (k_r), which are used to determine whether or not a point is aligned with a branch. The reconstructions obtained correctly matched with the real woody structure of the trees although they are not completely accurate.

The combination of constraints used (n_s, k_r and significant branch radius criterion) avoids divergence of the algorithm and makes the values of the parameters easier to find and less dependent on the type of tree to be reconstructed.

One major advantage of the model is that it only requires a short processing time, and it could therefore be suitable for use in whole orchard reconstruction with several trees trained with common agricultural systems. Orchard reconstruction could be approached by selecting N tree feet or root points and applying the algorithm to all of them simultaneously. In this case, a kd-tree structure will be required to improve the point-searching operations. Finally, a clustering process to separate branches that intermingle with each other in different trees would need to be introduced.

Table Captions

- **Table 1**. Function of the **main process** of reconstruction.
- **Table 2**. Function that searches for the nearest point to a branch (top) and the auxiliary function that gets the significant parent of a current branch (bottom).
- **Table 3**. Function that joints a set of children branches that get a single aligned branch.
- **Table 4**. Function that joints a branch with its parent branch when both are aligned.
- **Table 5**. Function that balances every branch with its parents to minimise both volumes.
- **Table 6**. Performance of the iterative and least-squares methods to estimate cylinder direction.
- **Table 7**. Main parameters used in the analysed reconstructions. k_r is the radius factor used to consider if a new point is aligned in a current branch or allows a new branch; Δy is the distance between vertical scans; $\Delta\theta$ is the angular resolution of the LiDAR sensor; t_d is the parameter of the projection of a point over cylinder axis $\overline{P_1P_2}$ ($t_d = 0$ when it is projected over P_1 and 1 if it is projected over P_2); n_s is the number of points that freely seed a cylinder when the building of a new branch starts (this parameter changes depending on the branch order (ord)).
- **Table 8**. **Number of points in the point cloud, n**umber of branches, processing time and volume simulated and reconstructed by the process.

- **Table 9**. Cost of the developed functions, being N the total number of points of the cloud, n_b the total number of branches and O the worst case of computing time by dimension of input data..

Figure Captions

- **Fig. 1**. MTLS unstructured point cloud simulated with SIMLIDAR (a), where a triangulated irregular network (TIN) has been calculated. The broad capillarity prevents reconstruction through filtering of initial tetrahedrons (b) by size (c).
- **Fig. 2**. Reconstructions of a virtual apple-tree (a) and vineyard (b) from their simulated MTLS. Reconstruction of a real pear-tree (c) from their MTLS. The order number is represented as cycles of red, green and blue colours.
- **Fig. 3**. Data model of CBranch class.
- **Fig. 4**. Effect of input parameters on tree model reconstruction: branches of the model wider than those of the measured tree (a); one unrealistic large trunk containing all the points (b); excessive branching (c).

ACKNOWLEDGMENTS

This research was partially funded by the Spanish Ministry of Science and Innovation (SAFESPRAY Project; Agreement No. AGL2010-22304-C04-03)

REFERENCES

De Aguiar, E., Stoll, C., Theobalt, C., Ahmed, N., Seidel, H.-P., & Thrun, S. (2008a). Performance capture from sparse multi-view video. *ACM Transactions on Graphics*. 27(3), Article No. 98.

De Aguiar, E., Theobalt, C., Thrun, S., & Seidel, H.-P. (2008b). Automatic conversion of mesh animations into skeleton-based animations. *Computer Graphics Forum*. 27(2), 389-397.

Auat Cheein, F., & Guivant J. (2014). SLAM-based incremental convex hull processing approach for treetop volume estimation. *Computers and Electronics in Agriculture* 102, 19–30.

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

Cormen, T.H., Leiserson, C.E., Rivest, R.L., & Stein, C. (2009). *Introduction to Algorithms* (pp. 248-300), (3rd ed.). The MIT Press.

Costes, E., Smith, C., Renton, M., Guédon, Y., Prusinkiewicz, P., & Godin, C. (2008). MAppleT: simulation of apple tree development using mixed stochastic and biomechanical models. *Functional Plant Biology*, 35, 936–950.

Crouse, M.S., Nowak, R.D., & Baraniuk, R.G. (1998). Wavelet-based signal processing using hidden Markov models. *IEEE Transactions on Signal Processing*, 46, 886–902.

Delagrange, S., & Rochon, P. (2011). Reconstruction and analysis of a deciduous sapling using digital photographs or terrestrial-LiDAR technology. *Annals of Botany* 108, 991–1000.

Durand, J.B., Guédon, Y., Caraglio, Y., & Costes, E. (2005). Analysis of the plant architecture via tree-structured statistical models: the hidden Markov tree models. *New Phytologist*, 166, 813–825.

Gorte, B., & Winterhalder, D. (2004). Reconstruction of laser-scanned trees using filter projections in the 3D-raster domain. International Archives of Photogrammetry, *Remote Sensing and Spatial Information Sciences*, 36 (Part 8/W2), 39-44.

Guivant, J., Nebot, E., & Durrant-Whyte, H. F. (2002). Simultaneous Localization and Map Building Using Natural features in Outdoor Environments. *Robotics and Autonomous Systems*, 20(2-3), 79-90

Henning, J., & Radtke, P. (2006). Detailed stem measurements of standing trees from ground-based scanning LIDAR. *Forest Science*, 52(1), 67-80.

Iagnemma, K., Kang, S., Shibly, H., & Dubowsky, S. (2004). Online terrain parameter estimation for wheeled mobile robots with application to planetary rovers. *Robotics, IEEE Transactions* 20 (5), 921-927.

Lloyd, S. (1982). Least square quantization in PCM. *IEEE Transactions on Information Theory*, 28, 129-137.

Lukács, G., Martin, R., & Marshall, D. (1998). Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation. In: ECCV ’98: *Proceedings of the 5th European Conference on Computer Vision-Volume I* (pp. 671-686), Springer-Verlag.

Marshall, A. D., Lukács, G., & Martin, R. (2001). Robust segmentation of primitives from range data in the presence of geometric degeneracy. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(3), 304–314.

Méndez, V., Catalán, H., Rosell, J.R., Arnó, J., Sanz, R., & Tarquis, A. (2012). SIMLIDAR – Simulation of LiDAR performance in artificially simulated orchards. *Biosystems Engineering*, 111(1), 72-82.

Méndez, V., Catalán, H., Rosell, J.R., Arnó, J., & Sanz, R. (2013). LiDAR simulation in modelled orchards to optimise the use of terrestrial laser scanners and derived vegetative measures. *Biosystems Engineering*, 115, 7-19.

Mizoue N., & Masutani T. (2003). Image analysis measure of crown condition, foliage biomass and stem growth relationships of Chamaecyparis obtusa. *Forest Ecology and Management* 172, 79–88.

Nebot E., & Durrant-Whyte H. (1999). Initial Calibration and Alignment of Low Cost Inertial Navigation Units for Land Vehicle Applications. *Journal of Robotics Systems*, 16(2), 81-92.

Neira, J., Tardos, J.D., & Castellanos, J.A. (2003). Linear time vehicle relocation in SLAM. Proceedings. ICRA '03. *IEEE International Conference on Vol 1* (pp. 427-433). Robotics and Automation, 2003.

Phattaralerphong J., & Sinoquet H. (2005). A method for 3D reconstruction of tree crown volume from photographs: assessment with 3D-digitized plants. *Tree Physiology* 25, 1229–1242.

Phattaralerphong J., & Sinoquet H. (2007). *Tree analyser: software to compute tree structure parameters from photographs*. *User manual*. PIAF-INRA. http://www2.clermont.inra.fr/piaf/eng/download/download.php.

Pfeifer, N., Gorte, B., & Winterhalder, D. (2004). Automatic reconstruction of single trees from terrestrial laser scanner data. Proceedings of 20th ISPRS Congress: Geo-Imagery Bridging Continents, 12-23 July, Istanbul, Turkey, pp. 114-119.

Preuksakarn, C., Boudon, F., Ferraro, P., Durand, J.B., Nikinmaa, E., & Godin, C. (2010). Reconstructing Plant Architecture from 3D Laser scanner data. 6th International Workshop on Functional-Structural Plant Models, Davis: USA.

Rabbani, T., & Heuvel, F. (2005). Efficient Hough transform for automatic detection of cylinders in point clouds. ISPRS WG III/3, III/4, V/3 Workshop "Laser scanning 2005", Enschede, the Netherlands, September 12-14

De Reffye, P., Edelin, C., Jaeger, M., & Puech, C. (1988). Plant models faithful to botanical structure and development. *Computer Graphics*, 22, 151–158.

Reulke, R., & Haala, N. (2005). Tree Species Recognition with Fuzzy Texture Parameters. Combinatorial Image Analysis, Springer. *Lecture Notes in Computer Science*, 3322, 607-620.

Rosell, J. R., Llorens, J., Sanz, R., Arnó, J., Ribes-Dasi, M., Masip, J., Escolà, A., Camp, F., Solanelles, F., Gràcia, F., Gil, E., Val, L., Planas, S., & Palacín, J. (2009). Obtaining the three-dimensional structure of tree orchards from remote 2D terrestrial LIDAR scanning. *Agricultural and Forest Meteorology*, 149, 1505–1515.

Sanz-Cortiella, R., Llorens-Calveras, J., Escolà, A., Arnó-Satorra, J., Ribes-Dasi, M., Masip-Vilalta, J., Camp, F., Gràcia-Aguilá, F., Solanelles-Batlle, F., Planas-DeMartí, S., Pallejà-Cabré, T., Palacin-Roca, J., Gregorio-Lopez, E., Del-Moral-Martínez, I., & Rosell-Polo, J. R. (2011). Innovative LIDAR 3D Dynamic Measurement System to Estimate Fruit-Tree Leaf Area. *Sensors*, 11, 5769–5791.

Shlyakhter I., Rozenoer M., Dorsey J., & Teller S. (2001). Reconstructing 3D tree models from instrumented photographs. *IEEE Computer Graphics and Applications*, 21, 53–61.

Simonse, M., Aschoff, T., Spiecker, H., & Thies, M. (2003). Automatic determination of forest inventory parameters using terrestrial laser scanning. Proceedings of ScandLaser Workshop, 3-4 September 2003, Umea, Sweden, 251-257.

Tan P., Fang T., Xiao J., Zhao P., & Quan L. (2008). Single image tree modeling. *ACM Transactions on Graphics* 27: Article 108. doi:10.1145/1409060.1409061.

Verroust, A., & Lazarus, F. (2000). Extracting skeletal curves from 3D scattered data. *The Visual Computer*, February 2000, 16(1), 15-25.

Yan, D.-M., Wintz, J., Mourrain, B., Wang, W., Boudon, F., Godin, & C. (2009). Efficient and robust tree model reconstruction from laser scanned data points. *CAD/Graphics* 2009, 572-575.

Function	MainProcess
Input	Void
Output	Void

```
CreateKTree()
Branch ← GetFootTree()
List_Branches.Insert(Branch)
Iter From I = 1 To length(List_Branches)
        Branch ← List_Branches[I]
        If Branch.Status = 0 Then
                Status ← FindTheClosestPoint(Branch, ClosestPoint)
                If Status = 0 Then
                        Branch.Status ← 1
                Else If Status = 1 Then // No Aligned, new branch
                        List_Branches.Insert(new CBranch(ClosestPoint))
                Else // Aligned, insert point in current branch
                        Branch.AddPoint(ClosestPoint)
                End(If)
        End(If)
End(I)
AlignedChildrenBranches()
While(ConnectAlignedBranches)
While(Clustering)
Return
```

Table 1.- Function with the main process of reconstruction.

Function	FindTheClosestPoint
Input	Branch Object
Output	Status, ClosestPoint

```
storedDist = -1
mTree ← Find_Closed_KDTtree(objBranch)
Iter From I = 1 To lenth(mTree.ListPoint)
        Point = mTree.ListPoint[I]
        Dist = Distance(objBranch, Point)
        If Dist < storedDist  and Dist < Precision Then
                If NoCloserOtherBranch(Point, Branch) Then
                        storedDist ← Dist
                        ClosestPoint ← Point
                        IndexPoint  ← I
                End(If)
        End(If)
End(I)
If storedDist = -1 Then
        Status ← 0
        Return
End(If)
mTree.ListPoint[IndexPoint].RemovePoint()
If Branch.NumPoints < FreeSeed Then
        Status ← 2
        Return
End(If)
Iter From I = 1 To Branch.NumPoints
        Temp.AddPoint(Branch.Point[I])
Temp.AddPoint(ClosestPoint)
ParentSignif ← ParentSignificant(Branch.Parent)
If Temp.Radius > ParentSignif.Radius Then
        Status ← 1
Else
        Status ← 2
End(If)
Return
```

Function	ParentSignificant
Input	currentBranch
Output	signifBranch

```
If currentBranch.order < OrderMin_1  Then
        signifBranch ← currentBranch
Else If currentBranch.order < OrderMin_2  Then
        If currentBranch.NumPoints > MinNumPoints  Then
                signifBranch ← currentBranch
        Else
                signifBranch ← ParentSignificant (currentBranch.Parent)
        End(If)
Else
```

```
        signifBranch ← ParentSignificant (currentBranch.Parent)
End(If)
Return
```

Table 2.- Function that searchs the nereast point to a branch (top) and the auxiliary function that gets the significant parent of a current branch (bottom).

```
Function  AlignedChildrenBranches
Input     void
Output    void
Iter From I = 1 To length(List_Branches)
    Branch ← List_Branches[I]
    If Branch.NumChildren > 1 Then
        Iter From K = 1 To Branch.NumChildren
            Child ← Branch.ListChildren[K]
            Angle[K] ← ArcCos(Branch.direction, Child.direction)
        End(K)
        Iter From K = 1 To Branch.NumChildren
            Iter From J = 1 To Branch.NumChildren
                If K≠J and abs(Angle[K]-Angle[J])<4º  Then
                    Child1 ← Branch.ListChildren[K]
                    Child2 ← Branch.ListChildren[J]
                    Iter From T = 1 To Child2.NumPoints
                        Child1.AddPoint(Child2.Point[T])
                    Remove(Child2)
                    ChangeParent(Child2, Child2)
                End(If)
            End(J)
        End(K)
    End(If)
End(I)
```

Table 3.- Function that joints a set of children branches that get a one aligned branch.

```
Function  ConnectAlignedBranches
Input     Void
Output    Connected
Iter From I = 1 To length(List_Branches)
    Branch ← List_Branches[I]
    Parent ← Branch.Parent
    Angle ← ArcCos(Branch.direction, Parent.direction)
    If abs(Angle)<11.5º  Then
        Iter From K = 1 To Branch.NumPoints
            Parent.AddPoint(Branch.Point[K])
        Remove(Branch)
        ChangeParent(Branch, Parent)
        Connected ← True
    End(If)
End(I)
```

Table 4.- Function that joints a branch with its parent branch when both are aligned.

Function	Clustering
Input	Void
Output	ChangedPoint

```
Iter From I = 1 To length(List_Branches)
    Iter From Side = 1 To 2
        If Side = 1 Then
            Branch ← List_Branches[I]
            Parent ← Branch.Parent
        Else
            Branch ← Branch.Parent
            Parent ← List_Branches[I]
        End(If)
        Iter From K = 1 To Branch.NumPoints
            Iter From J = 1 To Branch.NumPoints
                If J ≠ K Then
                    Tmp1.AddPoint(Branch.Point[J])
            End(J)
            Iter From J = 1 To Parent.NumPoints
                Tmp2.AddPoint(Parent.Point[J])
            Tmp2.AddPoint(Parent.Branch[K])
            DiffBranch ← Tmp1.Volume() – Branch.Volume()
            DiffParent ← Tmp2.Volume() – Parent.Volume()
            If DiffBranch + DiffParent < 0 and Tmp1.radio < Tmp2.radio Then
                Parent.AddPoint(Branch.Point[K])
                Branch.DeletePoint[K]
                ChangedPoint ← True
            End(If)
        End(K)
    End(Side)
End(I)
```

Table 5.- Function that balance every branch with its parents to minimize the volume of both.

Method	Runinng time (ms)	Average Angle($\vec{d}$, $\overrightarrow{d^*}$)	Standard Deviation Angle($\vec{d}$, $\overrightarrow{d^*}$)
Iterative	9.37	0.45 º	0.23 º
Least-squared	0.31	0.13 º	0.06 º
	$\vec{d}$ real direction, $\overrightarrow{d^*}$ estimated direction		

Table 6.- Performance of the Iterative and Lest-squared methods to estimate the cylinders direction**.**

			k_r			
	Δy (cm)	$\Delta\theta$ (º)	$t_d < 0.9$	$t_d \geq 0.9$	$ord(n_s)$	n_s
Apple tree	1	0.5	1.05	1.10	1;3;7;10;9999	80;60;40;30;20
Vine	1	0.5	1.05	1.10	1;3;9999	80;50;20
Pear tree	1.5	1	1.05	1.05	1;7;9999	20;15;10

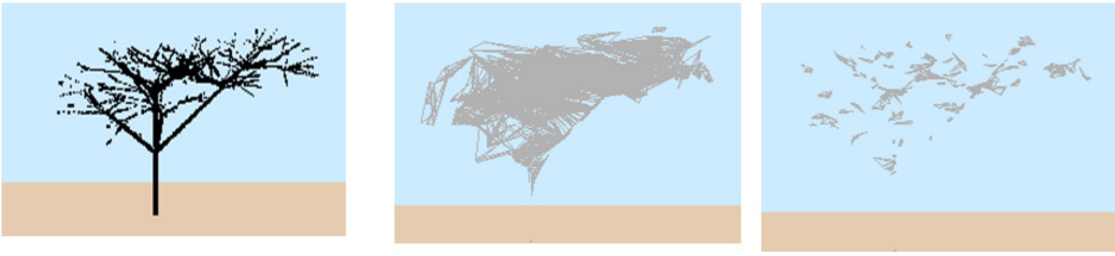
Table 7.- Main parameters used in the analysed rebuildings. Being k_r the factor of radium to consider if a new point is aligned in a current branch or allow a new branch; Δy the distance between vertical scans; $\Delta\theta$ the angular resolution of laser; t_d parameter of projection of a point over cylinder axis $\overline{P_1P_2}$ ($t_d = 0$ when is projected over P_1 and 1 if is projected over P_2); n_s the number of points that seed freely a cylinder when starts the building of a new branch, parameter that changes depending of order of branch (ord).

	#Points	# Branches	Processing time (min)	Model order	Rebuilding order	Vol. model (dm^3)	Vol. rebuilt (dm^3)	% Vol. Error
Apple tree	2,350	164	1	7	11	2.80	3.63	29%
Vine	4,941	271	2	10	12	6.83	7.41	8%
Pear tree	2,741	278	0.5		20			

Table 8.- Number of points in the point cloud, number of branches, processing time and volume simulated and rebuilt by the process.

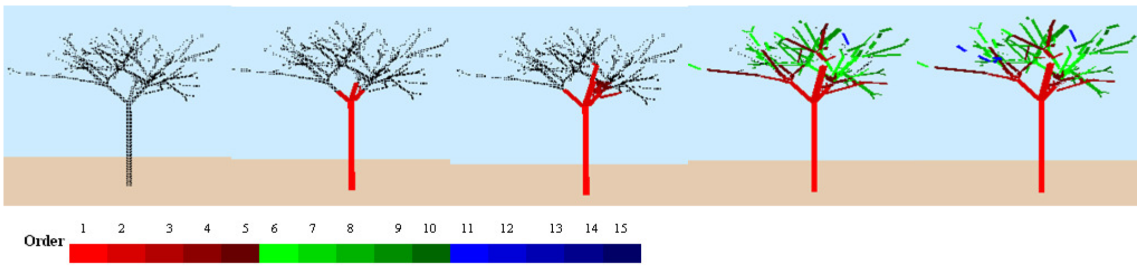
Function	***Cost***
CreateKTree	$O(N \cdot \log(N))$
FindTheClosestPoint	$O(\log(N) \cdot \log(n_b))$
AlignedChildrenBranches	$O(n_b)$
ConnectAlignedBranches	$O(n_b \cdot \log(n_b))$
Clustering	$O(N \cdot \log(n_b))$
MainFunction	$O(N \cdot \log(N) \cdot \log(n_b))$

Table 9. Cost of the functions. Being N the total number of points of the cloud, n_b the total number of branches and O the worst case scenario in terms of computing time according to the dimension of input data.

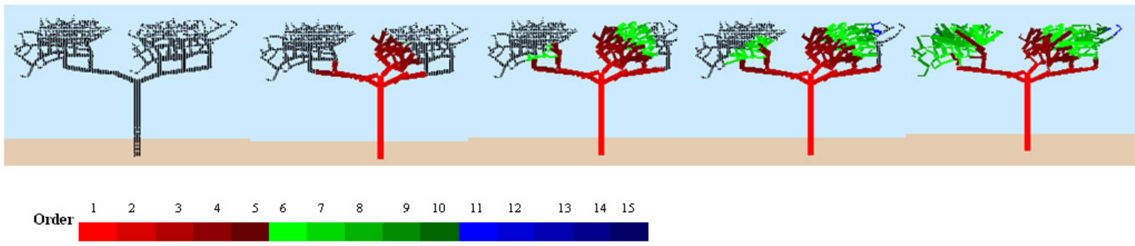


a b c

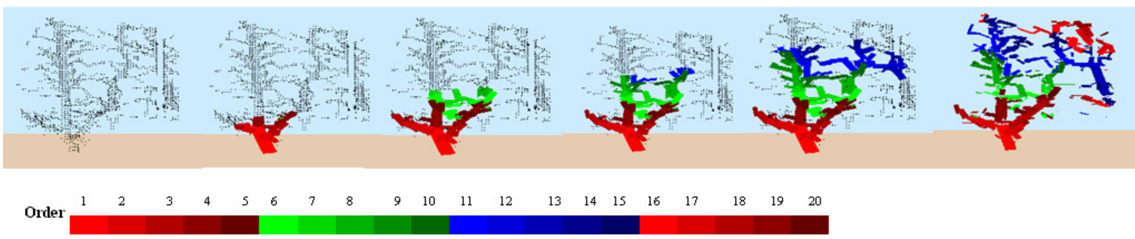
Fig. 1. MTLS unstructured point cloud simulated with SimLidar (a), where a triangulated irregular network (TIN) has been calculated. The broad capillarity prevents that a filters of initial tetrahedrons (b) by size (c) could be used to characterize the stems structure.



(a)



(b)

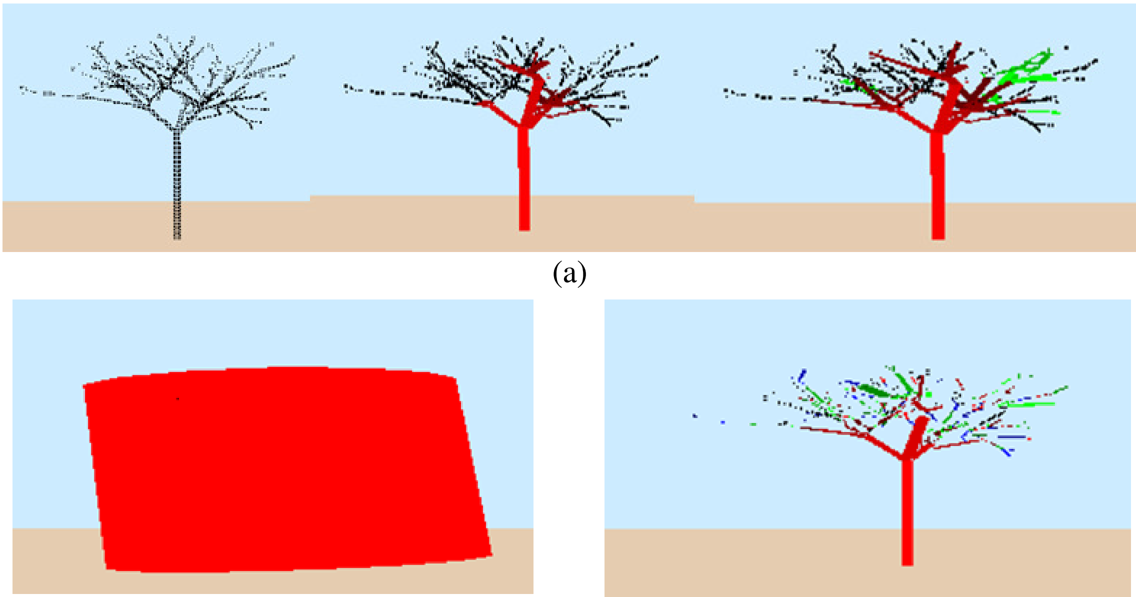


(c)

Fig 2. Rebuildings of a virtual apple-tree (a) and vineyard (b), from its simulated T-LiDAR. Rebuilding of a real pear-tree (c) from their T-Lidar. The order number is represented as cycles of red, green and blue colors.

```
class CBranch
{
        CPoint3D * m_points;
        long          NPoints;
        CPoint3D * m_P1;
        CPoint3D * m_P2;
        CPoint3D * m_G;
        CPoint3D * m_direct;
        float          m_radius;
        CRama      * m_predecessor;
        CRama    ** m_successor;
        int              NSuccessor;
        int              m_order;
}
```

Fig. 3. Data model of CBranch class.



(a)

(b)

(c)

Fig 4. Effect of the input parameters in the rebuilding of tree models: branches of the model wider than the actual tree (a); one unreal big trunk containing all the points (b); too much branching (c).