High Accuracy Terrain Reconstruction from Point Clouds Using Implicit Deformable Model

Jules Morel[1(✉)], Alexandra Bac[2], and Takashi Kanai[1]

[1] Kanai Laboratory, Graduate School of Arts and Sciences, The University of Tokyo, Tokyo, Japan
jules.morel@ifpindia.org

[2] Laboratoire d'Informatique et des Systèmes, Aix Marseille University, Marseille, France

Abstract. Few previous works have studied the modeling of forest ground surfaces from LiDAR point clouds using implicit functions. [10] is a pioneer in this area. However, by design this approach proposes over-smoothed surfaces, in particular in highly occluded areas, limiting its ability to reconstruct fine-grained terrain surfaces. This paper presents a method designed to finely approximate ground surfaces by relying on deep learning to separate vegetation from potential ground points, filling holes by blending multiple local approximations through the partition of unity principle, then improving the accuracy of the reconstructed surfaces by pushing the surface towards the data points through an iterative convection model.

Keywords: Implicit surface · Deformable model · Deep learning

1 Introduction

Digital terrain model (DTM) extraction is an important issue in the field of LiDAR remote sensing of Earth surface. Indeed, further data processing procedures (segmentation, surface reconstruction, digital volume computation) usually rely on a prior DTM computation to focus on features of interest (buildings, roads or trees for instance). In the last two decades, many filtering algorithms have been proposed to solve this problem using Airborne LiDAR sensors (ALS) data.

However, most previous works address DTM extraction for ALS data, which largely differs from terrestrial LiDAR sensors (TLS) data. Unlike ALS point clouds, TLS ones provide very dense sampling rates at the ground level, describing the micro-topography around the sensor. However, the presence of vegetation and the terrain topography itself generate strong occlusions causing large data gaps at the ground level, and a risk of integrating objects above the ground within the DTM. Additionally, the scanning resolution of TLS devices depends,

V. V. Krzhizhanovskaya et al. (Eds.): ICCS 2020, LNCS 12142, pp. 251–265, 2020.
https://doi.org/10.1007/978-3-030-50433-5_20

by nature, on their distance to the sensor, resulting in spatial variations in point density. Therefore, DTM extraction for TLS data requires dedicated approaches, in particular for 3D samples acquired in forest environments. [10] is a pioneering effort designed to reconstruct detailed DTMs from TLS data under forest canopies using implicit function. The basic idea of this work is (1) to approximate locally the ground surface through adaptive scale refinement based on a quad-tree division of the scene, (2) then to simultaneously filter vegetation and correct approximations based on the points distribution in each quad-tree cell, and (3) to blend the local approximations into a global implicit model. In the present paper, we build upon this previous work and propose several innovations in order to reconstruct high quality terrain models from laser scan point clouds.

2 Overview of the Method

Based on the previous work [10], our method relies on adaptive scale refinement through a quad-tree division of the scene, local approximations in the quad-tree cells, and estimation of a global model through the blending of local approximations, to compute the first approximation of the ground surface. The method presented in this paper offers several improvements of the previous work to enhance the accuracy of the reconstructed terrain model: First, we relegate the filtering out of vegetation points to a deep network based on PointNet++ [14]. Second, we attribute weights to the data points according to their local density in order to obtain a more robust local approximation. Third, we refine the blending of local approximations to make it compliant with the partition of unity concept with compactly supported radial basis functions (CSRBF). Finally, we introduce an original deformable model, based on a convection model under a moving least squares field and a proper functional basis. We thus push the implicit ground surface towards data points to obtain centimeter accuracy. The complete sequence of computational steps is shown in Fig. 1.

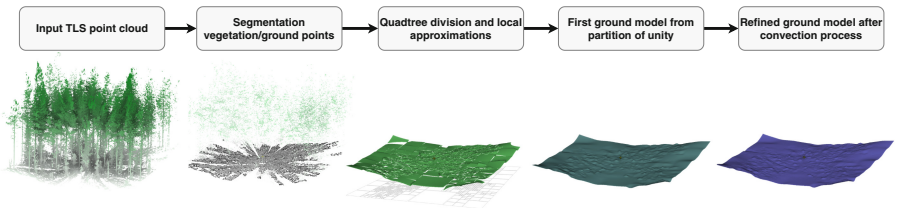


Fig. 1. Overview of method. The segmented ground points are colored according to their weights coming from the local density.

3 Filtering of Vegetation

3.1 Selection of Ground Points Using Deep Learning

In the estimation of ground surface from LiDAR point clouds, it is common to first project the point cloud into a fine regular 2D (x,y) grid, of resolution res_{MIN}, and further select the points of minimum elevation in each cell. The resulting point cloud is denoted P_{min}. In order to ease the segmentation, we enriched each point of P_{min} by geometric descriptors encoding the local linearity and planarity. In practice, we consider for each points its Cartesian coordinates plus the averaged eigenvalues of the local covariance matrix of the 3D distribution of the K_{PCA} neighbors, K_{PCA} being defined by the user. In order to feed these enriched point clouds into a convolutional architecture, we define a partition of space allowing splitting of an unordered 3D point cloud into overlapping fixed size regions, each of them encoding the 3D local points distribution. In practice, we divide the point cloud P_{min} into a regular 2D (x,y) grid, then distribute collocation points to the centroid of every occupied voxel. Then, we form each batch $\mathscr{B}_i$ by considering the 1024 nearest neighbors of each collocation point in the (x,y) plane. These batches are designed to feed and train a deep learning model; this model eventually predicts a label for each point of each batch. As batches largely overlap, each input point belongs to several batches and thus receive multiple predictions, one for every batch it belongs to. In the last step, we define a voting process to extract a final segmentation from the multiple votes per point. The resulting set of points denoted by $P = \{\mathbf{p}_i\}_{i=1...N} \subset \mathbb{R}^3$ serves as raw data for the proposed algorithm.

3.2 Weighting of Points According to the Local Density

A common practice in surface reconstruction from in-homogeneous point clouds consists of applying a weight to each point according to the local density around $\mathbf{p}_i$. In order to take into account density irregularities due to overlapping scans and the confidence measures attributed to the points, the proposed method by [11] scales down the point influence in high density areas. We were inspired by this idea but implement the opposite scaling: in forest 3D scans, while considering terrain modeling, clusters of points tend to describe accurately ground topology whereas isolated points appear less reliable and their influence needs to be reduced. To do so, we assigned a weight d_i at each point $\mathbf{p}_i$ given by

$$d_i = 1 - \frac{1}{max_d} \sum_{\mathbf{p}_j \in \mathscr{P}_j^K} \|\mathbf{p}_i - \mathbf{p}_j\| \tag{1}$$

where P_j^K is the K-neighborhood of point $\mathbf{p}_i$ (from our experiments $K = 20$ appears as a good choice), max_d is defined as:

$$max_d = \max \left[\sum_{\mathbf{p}_j \in \mathscr{P}_j^K} \|\mathbf{p}_i - \mathbf{p}_j\|, \forall\ \mathbf{p}_i \in P \right] \tag{2}$$

4 First Surface Estimation by Blending Local Quadrics Approximations

4.1 Quadtree Division of the Plane

Initially, the minimum bounding rectangle of filtered minimum points is inserted as the root of the quadtree. Then, we iteratively subdivide each cell into four leaves. The process stops when a leaf presents a side smaller than $Size_{min}$, the minimal size allowed defined by the user, or when n_{cell} the number of points in the cell becomes inferior to the number of parameters of the local quadric patches (i.e. $n_{cell} < 6$) described in the following Sect. 4.2.

4.2 Local Approximations

Taking into account a quadtree that divides the data points, for each of its leaves $\mathscr{O}_i$, we denote by $\mathbf{c}_i$ the center of $\mathscr{O}_i$.

In each cell $\mathscr{O}_i$, an approximate tangent plane is computed using least square fitting. Let (u, v, w) denote a local orthonormal coordinate system such that the direction z_l is orthogonal to the fitting plane. We approximate $\mathscr{O}_i$ by a local implicit quadric function $g_i(u, v, w) = w - h_i(u, v)$ where, in the local coordinate system, h_i is a quadratic parametric surface of the following form: $h(u, v) = A \cdot u^2 + B \cdot u \cdot v + C \cdot v^2 + D \cdot u + E \cdot v + F$. In each leaf $\mathscr{O}_i$, considering $\mathbf{p}_j = (u_j, v_j, w_j) \in \mathscr{P}$, coefficients A, B, C, D, E, F are determined by the weighted least square minimization of:

$$\sum_{\mathbf{p}_j \in \mathscr{P}} d_i \cdot (w_j - h(u_j, v_j))^2 \cdot \Phi_{\sigma^i}(\|\mathbf{p}_j - \mathbf{c}_i\|) \tag{3}$$

where d_i is the weight defined in Sect. 3.2 and $\Phi_{\sigma^i}(\|x - c_i\|) = \phi(\frac{\|x-c_i\|}{\sigma^i})$ and $\phi(r) = (1-r)_+^4(1+4r)$ is a compactly supported Wendland's RBF function [16]. The function ϕ is $\mathscr{C}^2$ and radial on $\mathbb{R}^3$. The parameter σ^i controls the influence of the local approximation of the leaf $\mathscr{O}_i$. In our case, in order to always cover the bounding box of the points contained in the leaf $\mathscr{O}_i$ by a ball of radius σ^i, we chose $\sigma^i = a_i \times \sqrt{3} \times 0.75$, where a_i is the longest side of the leaf $\mathscr{O}_i$.

4.3 Global Approximation by Blending Local Approximations

After estimating the local approximations of the data points for each leafs $\mathscr{O}_i$, the global implicit model by blending together local patches by means of Wendland's CSRBF. The global implicit model f is computed as:

$$f(\mathbf{x}) = \sum_{\mathbf{c}_i \in C} g_i(\mathbf{x}) \cdot \frac{\Phi_{\sigma^i}(\|\mathbf{x} - \mathbf{c}_i\|)}{\sum_{\mathbf{c}_j \in C} \Phi_{\sigma^j}(\|\mathbf{x} - \mathbf{c}_j\|)} \tag{4}$$

where $g_i(\mathbf{x})$ is the local approximation in the leaf $\mathscr{O}_i$ defined in the previous Section.

More precisely we use compactly supported Wendland's RBF functions [16] as a partition of unity to merge the local implicit functions (issued from parametric patches) and compute a global implicit model.

5 Improvement of Surface Model with a Convection Model

Because occlusion phenomenon forces large quadtree cells, and thus coarser local approximations, the first surface estimation described in Sect. 4 can be further improved by being pushed closer to the data point. Inspired by the level-sets formulations [12,15] in the field of image processing, our method builds particularly on the work of Gelas [6] and describes the evolution of the surface driven by a time-dependent partial differential equation (PDE) where the so-called velocity term reflects the 3D point cloud features. The level-set PDE is then solved through a collocation method using CSRBF.

5.1 From Blended Quadrics Model to CSRBF Model

In order to both express the refined implicit surface and discretize the numerical problem, we create a new base of functions. This basis is built according to the following process: collocation centers are regularly distributed on a 3D grid, of user-defined resolution r, around the zero-levelset of the implicit function 4. In order to limit the number of collocations while allowing the surface to move in the vicinity of data points but, we decimate the clouds of collocation by removing each center further than $2 \times r$ from a data point or further than r from the zero-levelset. Considering $\mathscr{Q}$ the set of collocation, the basis of function is then defined as the set of translates at each remaining collocation $\mathbf{o} \in \mathscr{Q}$ of Wendland's CSRBF $\Phi_{r_o}(\|\mathbf{x}-\mathbf{o}\|) = \phi(\frac{\|\mathbf{x}-\mathbf{o}\|}{r_o})$: where $r_o = r \times \sqrt{3} \times 0.75$.

First, we approximate the implicit function f defined in Eq. 4 in the new basis $Span\{\Phi_{r_o}\}$ as $g(\mathbf{x}) = \sum_{\mathbf{o}\in\mathscr{Q}} \alpha_{\mathbf{o}} \cdot \Phi_{r_o}(\|\mathbf{x}-\mathbf{o}\|)$.

To retrieve $\{\alpha_{\mathbf{o}}\}_{\mathbf{o}\in\mathscr{Q}}$, we consider the system:

$$\begin{vmatrix} \ddots & & \\ & \mathscr{A}_{\mathbf{o},\mathbf{o}'} & \\ & & \ddots \end{vmatrix} \begin{vmatrix} \vdots \\ \alpha_{\mathbf{o}} \\ \vdots \end{vmatrix} = \begin{vmatrix} \vdots \\ f(\mathbf{o}') \\ \vdots \end{vmatrix} \tag{5}$$

where o and o' are two collocations, $\mathscr{A}_{\mathbf{o},\mathbf{o}'} = \Phi_{r_o}(\|\mathbf{o}'-\mathbf{o}\|)$ and the implicit function f describing the blended quadrics model is expressed by Eq. 4. Thanks to the properties of Wendland's CSRBF Φ_{r_o}, the matrix of the $\mathscr{A}_{\mathbf{o},\mathbf{o}'}$ elements is sparse, self-adjoint and positive definite. Thus, the system is inverted by using the LDL^T Cholesky decomposition implemented by Eigen library.

5.2 Solving Convection Evolution Equation Using CSRBF Collocation

We define now the velocity vector $\mathscr{V}(\mathbf{p},t), \mathbf{p} \in \mathbb{R}^3, t \in \mathbb{R}$, a function reflecting the geometrical properties of the interface according to the data, and quantifying the local deformations over time. The velocity is actually computed at each

collocation $\mathbf{o} \in \mathscr{O}$ by minimization of an energy function inspired by the snake approaches introduced by Kass [8]. Indeed, we define the energy function $\forall \mathbf{q} \in \mathbb{R}^3, \xi(\mathbf{q}) = \gamma \cdot \xi_{data}(\mathbf{q}) + (1-\gamma) \cdot \xi_{surface}(\mathbf{q})$, where $\xi_{data}(\mathbf{q})$ gives rise to sample points force and $\xi_{surface}(\mathbf{q})$ gives rise to external constraint forces. A weighting scalar parameter γ, defined by the user, adjusts the influence of both terms.

At each collocation $\mathbf{o} \in \mathscr{O}$, we consider a local plane of normal $\mathbf{n}$ fitting the data points. For a given point $\mathbf{q} \in \mathbb{R}^3$, $\mathbf{p}$ is its projection on the this plane, and t the distance between $\mathbf{q}$ and the plane; so we have $\mathbf{p} = \mathbf{q} - t \times \mathbf{n}$. Figure 2 summarizes this setup.

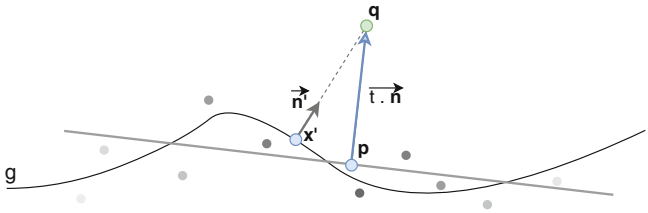


Fig. 2. The weight on an data point $\mathbf{p}_i \in P$, denoted here by different shades of gray, is a function of the distance from $\mathbf{p}_i$ to $\mathbf{q}$.

Thus, ξ_{data} and $\xi_{surface}$ are expressed at the point $\mathbf{q}$ as functions of t and $\mathbf{n}$:

$$\begin{aligned} \xi_{data}(t,\mathbf{n}) &= \sum_{\mathbf{p}_i \in P} \langle \mathbf{p}_i - \mathbf{q} + t \times \mathbf{n}, \frac{\mathbf{n}}{\|\mathbf{n}\|} \rangle^2 \quad \cdot \theta(\|\mathbf{p}_i - \mathbf{q} + t \times \mathbf{n}\|) \\ \xi_{surface}(t,\mathbf{n}) &= \langle \mathbf{x}' - \mathbf{q} + t \times \mathbf{n}, \mathbf{n}' \rangle^2 \end{aligned} \tag{6}$$

where $\mathbf{n}'$ is the gradient of the implicit function defining the continuous tubular model, $\mathbf{x}'$ is the projection of $\mathbf{p}$ on the zero level-set of this function, and θ is a compactly supported Wendland's RBF.

For clarity, $\xi_{surface}$ quantifies the distance between the continuous tubular model and MLS model [1] locally approximating the data. The gradient of ξ is then expressed as $\nabla\xi = \gamma \cdot \nabla\xi_{data} + (1-\gamma) \cdot \nabla\xi_{surface}$. Following Eqs. 7 and 8 detail the computation of both data and surface gradient terms, respectively ξ_{data} and $\xi_{surface}$.

$$\begin{aligned} \nabla\xi_{data}(t,\mathbf{n}) = \sum_{p_i} & 2 \cdot \langle \mathbf{p}_i - \mathbf{q} + t \times \mathbf{n}, \frac{\mathbf{n}}{\|\mathbf{n}\|} \rangle \cdot \nabla_{\mathscr{G}}(t,\mathbf{n}) \cdot \theta(\|\mathbf{p}_i - \mathbf{q} + t \times \mathbf{n}\|) \\ & + \langle \mathbf{p}_i - \mathbf{q} + t \times \mathbf{n}, \frac{\mathbf{n}}{\|\mathbf{n}\|} \rangle^2 \cdot \mathscr{M}^T_{(t,\mathbf{n})} \cdot \nabla\theta(\|\mathbf{p}_i - \mathbf{q} + t \times \mathbf{n}\|) \end{aligned} \tag{7}$$

$$\text{where} \begin{cases} \nabla_{\mathscr{g}}(t,\mathbf{n}) & = \left[\dfrac{1}{\|\mathbf{n}\|} \cdot (I_3 - \dfrac{\mathbf{n}\cdot\mathbf{n}^T}{\|\mathbf{n}\|^2}) \cdot (\mathbf{p}_i - \mathbf{q}) + t \cdot \dfrac{\mathbf{n}}{\|\mathbf{n}\|} \right] \\ & \qquad\qquad \|\mathbf{n}\| \\ \mathscr{M}_{(t,\mathbf{n})} & = \begin{bmatrix} & & n_x \\ I_3 \cdot t & & n_y \\ & & n_z \end{bmatrix} \\ \nabla\theta(\|\mathbf{r}\|) & = (-20 \cdot \|\mathbf{r}\|) \cdot (1 - \|\mathbf{r}\|)^3 \cdot \dfrac{\mathbf{r}}{\|\mathbf{r}\|} \end{cases}$$

The second gradient term is computed as:

$$\nabla\xi_{surface}(t,\mathbf{n}) = 2 \cdot \mathscr{M}^T_{(t,\mathbf{n})} \cdot \langle \mathbf{x}' - \mathbf{q} + t \times \mathbf{n}, \mathbf{n}' \rangle \cdot \mathbf{n}' \tag{8}$$

We retrieve $(t_{min}, \mathbf{n}_{min})$, minimizing $\xi(t,\mathbf{n})$ with the Fletcher-Reeves [5] conjugate gradient algorithm available in the GSL library. The minimizer is initiated at the point $\mathbf{q} \in \mathbb{R}^3$ to $(t_0, \mathbf{n}_0) = (\overrightarrow{\mathbf{q}-\mathbf{x}'} \cdot \mathbf{n}', \mathbf{n}')$. From this minimization process computed at each collocation $\mathbf{o}$ arises a deformation vector defined as $\mathbf{v_o} = -t_{min} \times \mathbf{n}_{min}$.

To push the surface to the data points according to the deformation vector field, we propose to follow a convection model as proposed by Osher [12]:

$$\frac{dg(\mathbf{p},t)}{dt} = \mathbf{v_p} \circ \nabla g(\mathbf{p},t) \tag{9}$$

where $\mathbf{v_p}$ is the deformation vector expressed at $\mathbf{p}$, and $\circ$ is the element-wise product. To solve Eq. 9, we assume that space and time are separable. We decompose $g(\mathbf{p},t) = \boldsymbol{\alpha}(t) \cdot \boldsymbol{\Phi}(\mathbf{p})$ where $\boldsymbol{\Phi}(\mathbf{p})$ is made of the basis functions values evaluated at the point $\mathbf{p}$ and $\boldsymbol{\alpha}(t)$ composed of the respective α values. Equation 9 thus becomes the ordinary differential equation of evolution:

$$\frac{d\boldsymbol{\alpha}(t)}{dt} \cdot \boldsymbol{\Phi}(\mathbf{p}) = \mathbf{v_p} \circ [\boldsymbol{\alpha}(t) \cdot \nabla\boldsymbol{\Phi}(\mathbf{p})] \tag{10}$$

After the application of Euler's method, Eq. 10 becomes:

$$\boldsymbol{\alpha}(t+\tau) = \boldsymbol{\alpha}(t) - \tau \cdot \boldsymbol{\Phi}(\mathbf{p})^{-1} \cdot \mathscr{H}(t,\mathbf{p}) \tag{11}$$

where τ is the time step and $\mathscr{H}(t,\mathbf{p}) = \mathbf{v_p} \circ [\boldsymbol{\alpha}(t) \cdot \nabla\boldsymbol{\Phi}(\mathbf{p})]$. The evolution of the CSRBF coefficients is finally given by Eq. 11. From this set of CSRBF, we compute an implicit function and finally extract its zero level-set to produce the terrain surface model.

5.3 Re-normalization of Implicit Function

In order to prevent the apparition of new zero level components far away from the initial surface, periodically reshaping the implicit function is a common strategy. In order to bound the function g defined in Sect. 5.1, hence its gradient ∇g, we

bound the expansion of its coefficients $\boldsymbol{\alpha}_{o_c}$. In practice, inspired by [6], we bound $\|\boldsymbol{\alpha}(t)\|_\infty$ if $\|\boldsymbol{\alpha}(t)\|_\infty > \beta$, where β is a positive constant. The evolution equation becomes

$$\begin{cases} \boldsymbol{\alpha}(t+\tau) = \widetilde{\boldsymbol{\alpha}}(t) - \tau \cdot \boldsymbol{\Phi}(\mathbf{p})^{-1} \cdot \mathscr{H}(t, \mathbf{p}) \\ \widetilde{\boldsymbol{\alpha}}(t+\tau) = \frac{\beta}{\|\boldsymbol{\alpha}(t+\tau)\|_\infty} \cdot \boldsymbol{\alpha}(t+\tau), & \text{if } \|\boldsymbol{\alpha}(t+\tau)\|_\infty > \beta \\ \widetilde{\boldsymbol{\alpha}}(t+\tau) = \boldsymbol{\alpha}(t+\tau), & \text{otherwise} \end{cases} \tag{12}$$

5.4 Complexity

Using an octree data structure for the collocations layout, as advised by Wendland [16], the matrix $\boldsymbol{\Phi}(\mathbf{p})$ computation is $\mathscr{O}(NlogN)$ and the implicit function evaluation g is $\mathscr{O}(logN)$. According to Botsch [2], the inversion of $\boldsymbol{\Phi}(\mathbf{p})$ through the Cholesky factorization is $\mathscr{O}(nzf)$, where nzf is the number of nonzero factors, which depends on the CSRBF center position and on the CSRBF support size. The cost of the re-normalization described in Sect. 5.3 is $\mathscr{O}(N)$.

6 Experiment and Validation

In order to train our deep learning model, labeled point clouds of forest plot mock-ups are required. To generate such data-sets, we simulate 3D point clouds from artificial 3D terrains and trees models, as described in the following Sect. 6.1. As the accuracy of the terrain surface reconstructed by our method depends first on the efficiency of our deep network segmentation, we first measured its ability to segment vegetation points from ground points on local minima of simulated scenes. Section 6.3 describes the quantitative and qualitative results of terrain reconstruction on simulated and real TLS scans. Moreover, we evaluate the improvement brought by the deformable model.

6.1 Training Data Generation

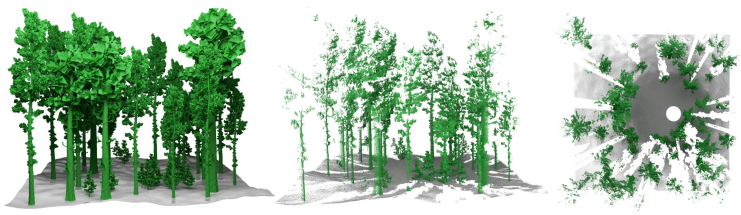


Fig. 3. Models of forest 32 m × 32 m plot: left side, mesh model; center and right side, side and top view of a simulated point cloud.

To build a training data-set whose point clouds resemble the real TLS forest scans, described later in Sect. 6.2, we designed realistic tree mesh models of

pines, spruces and birches using SpeedTree [7]. This software is providing nowadays high quality 3D renderings for the gaming and film industries as well as architectural visualization projects. Ground surfaces meshes were produced using hybrid multi-fractal terrain method [4] implemented within Meshlab [3]. Then, we associated ground and trees models to build 3D models of complete forest plots, and used a LiDAR simulator based on PBRT [13] to generate point clouds. In order to simulate the point distribution close to the ground, which is scattered and complex due to leaves and very small vegetation, we applied a Gaussian noise (std = 10 cm) on points originating from ray hitting the ground surface. Figure 3 shows an example of a meshed forest plot mock-up along with two views of the resulting simulated point clouds. As the goal of our deep network is to learn how to handle different patterns of trees occlusions, we artificially increased the training data by repeating the simulation process for six virtual scan positions evenly distributed on a 4 m circle centered on the plot. Each simulated point cloud has around 7 million points.

6.2 Real LiDAR Scans

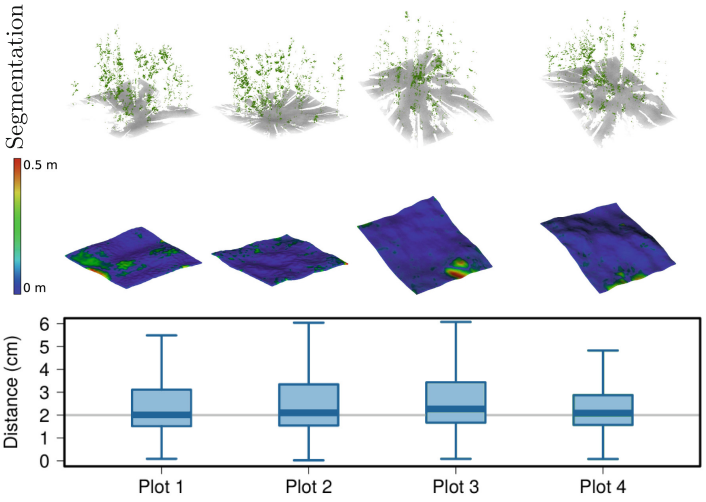
The real TLS scans used in the validation are the ones used in the benchmark [9]. They were acquired in a southern boreal forest of Finland. Each of the originating plot, of a fixed size of 32 m × 32 m, was selected from varying forest-stand conditions representing different developing stages with a range of species, growth stages, and management activities. The plots were divided into three categories (two plots per category) based on the complexity of their structure (from the point-of-view of a TLS survey): easy (plot 1 and 2), medium (plot 3 and 4) and difficult (plot 5 and 6). This data-set comes with a reference ground model, fully checked by the operator, composed of 3D points laying over a 2D grid of 20 cm resolution. Each plot was scanned from five positions: one scan at the plot center and four scans at the four quadrant directions.

6.3 Experiments and Discussions

As pointed out earlier, the performance of our method relies first on its ability to filter out vegetation points based on the deep learning segmentation. In order to assess this performance, we first compute the mean kappa indicator of the classification of all the simulated scenes, for different values of res_{MIN} the resolution of the 2D grid used in the extraction of the minimum points P_{min} and K_{PCA} the number of neighbors used in the local PCA analysis described in Sect. 3.1. Through sensitivity analysis, we obtained the best average kappa (0.977) on the simulated data-set for $res_{MIN} = 10$ cm and $K_{PCA} = 128$. We use those parameter values for the rest of the validation. Using a pre-trained model with these parameters, we analyzed the distance between the input data of our method and the reconstructed surfaces it produces.

Simulated Data. In each step of the validation, we checked the robustness of our method by asserting that the reconstructed surfaces were made of a single continuous patch without holes covering the full extent of the input data. Then, we measured its accuracy by computing the distance between the input data and the surface model. To do so, we defined a distance distribution by computing the euclidean distance in between each vertices of the resulting surfaces to the closest vertex of the reference surfaces. The Table 1 presents the segmentation result and the distance distribution for one scan of each artificial forest scan.

Table 1. Distance between reconstructed terrain surface and reference: (Top) Results of segmentation of the simulated point cloud. The vegetation points that are filtered out are represented in green. (Middle) The euclidean distance to the reconstructed surface represented as a color gradient at each vertex of the reference surface. (Bottom) The distribution of distances represented as box-plots (without outliers).



For every artificial plot, our model efficiently filters out the vegetation points. Moreover, the reconstructed surfaces are placed at a mean distance of 2 cm from the terrain model produced with Meshlab, except in large occlusions where ground points are missing and where the reconstructed surface is approximated from the surroundings.

Real Data. While dealing with real TLS scans, we reconstructed, for each plot, the terrain surface from the single central scan and also from a point cloud resulting of the fusion of all five available scans per plot. This allows evaluation of the method performance for various point density in different forest environments. Having access to the reference terrain model, we computed the distance in between surfaces, vertex to vertex. In the case of real data, we first

extract the ground points using our deep learning network, then compute the distance between the vertices of the reconstructed surface and the ground points packaged with the LiDAR scans. The results are shown in Table 2.

Finally, we analyzed the evolution of terrain model the iteration of deformation, by measuring its distance distribution to the segmented ground points. The results are presented in Tables 3 and 4, for single and multiple scans respectively.

Table 2. Mean Hausdorff distance (cm) between reconstructed surface from single/multiple scan(s) and reference ground model.

Plot	1	2	3	4	5	6
Single - Mean Distance (cm)	6.04	5.82	22.23	26.89	12.85	21.76
Multiple - Mean Distance (cm)	2.58	3.14	5.93	8.27	2.88	7.17

As pointed out in the Table 2, the efficiency of our method to reconstruct a terrain model close to the reference depends on the terrain complexity and on the nature of the scans: for the simpler forest plots (1 and 2) single scans, our method produces accurate terrain models that are positioned 6 cm from the reference. Due to the missing points in the occlusion that expands with complexity, this distance increases for plots 3, 4, 5, and 6. However, in the case of multiple scans, the occlusion phenomenon lessens, which makes available anchor points allowing sharper terrain surfaces to be produced. In such cases, with our method, the mean distances are around 3 cm on the simpler plot, and from 3 to 8 cm on a more complex topology.

In Tables 3 and 4, we analyzed the contribution of the deformation produced by the convection in the final accuracy of our terrain surfaces. For every plot, the mean of the distance distribution and its standard deviation decreases for both single and multiple scans. The stronger drops occurs at the first iteration, the following ones remaining light in comparison. Except for plot 3 and 4 single scans, the reconstructed surface sets at 2 cm on average from the segmented ground points after five iterations of convection.

The single scans of plots 3 and 4 highlight the limitation of our method: while the mean distance from the segmented ground points remained at 2 to 3 cm, the distance distribution presents extreme values above 20 cm. This is due to the failure of our deep learning model to correctly filter out vegetation points. Those two particular plots, while being scans from a single point presents problematic 3D points patterns, which vanish if scanned from multiple points of view. The forest mock-ups are crucial; they need to take proper account of the actual reality. Indeed, they are used to produce simulated data training and determine the quality of the segmentation, which is the first step of our method and condition the reconstruction of the overall scene.

Table 3. Estimation of forest plots ground surface based on TLS single scans. The two first columns show the side and top view of the point cloud. For visualization purpose, the point clouds have been colorized according to the point elevation and the local point density. The third column present the segmentation of the minimum points. The last column show the evolution of the distance between the reconstructed terrain model and the segmented ground points for several iteration of deformation.

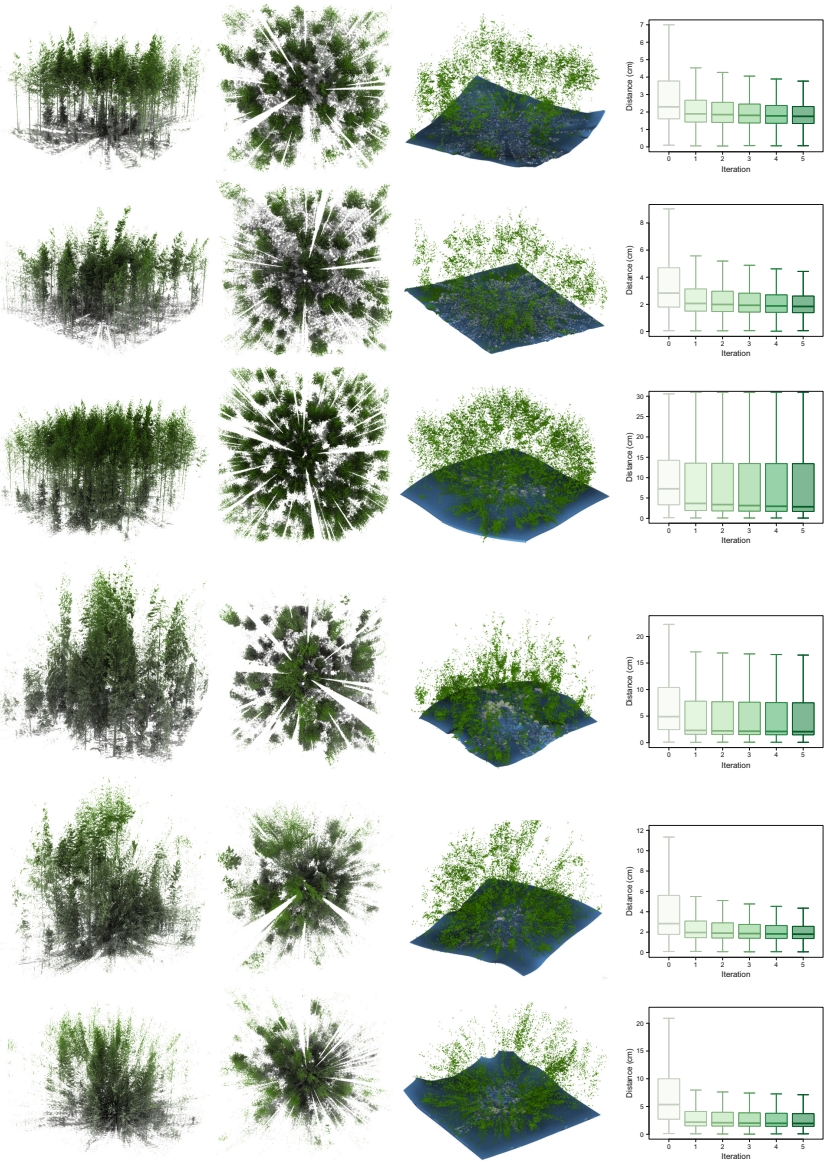


Table 4. Estimation of forest plots ground surface based on TLS multiple scans. The two first columns show the side and top view of the point cloud. For visualization purpose, the point clouds have been colorized according to the point elevation and the local point density. The third column present the segmentation of the minimum points. The last column show the evolution of the distance between the reconstructed terrain model and the segmented ground points for several iteration of deformation.

7 Conclusion

In this work, we propose an efficient method designed to recover terrain surface model from 3D sample data acquired in forest environments. Our approach relies on deep learning to separate ground points from vegetation points. It handles the occlusion and builds a first approximation of the ground surface by blending implicit quadrics through the partition of unity principle. Then our method gets rid of the rigidity of the previous model by projecting it in a CSRBF basis before deforming it by convection. These contributions enable us to achieve state of the art performance in terrain reconstruction. In the future, it is worthwhile thinking of how to design forest mock-ups adapted to train networks able to filter different forest environments.

Acknowledgments. The authors would like to thank the reviewers for their thoughtful comments and efforts towards improving our manuscript and the Japan Society for the Promotion of Science (JSPS) for providing Jules Morel fellowship.

References

1. Amenta, N., Kil, Y.J.: Defining point-set surfaces. In: ACM SIGGRAPH 2004 Papers, pp. 264–270 (2004)
2. Botsch, M., Hornung, A., Zwicker, M., Kobbelt, L.: High-quality surface splatting on today's GPUs. In: Proceedings Eurographics - IEEE VGTC Symposium Point-Based Graphics 2005, pp. 17–141. IEEE (2005)
3. Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., Ranzuglia, G.: MeshLab: an open-source mesh processing tool. In: Eurographics Italian Chapter Conference, vol. 2008, pp. 129–136 (2008)
4. Ebert, D.S., Musgrave, F.K.: Texturing & Modeling: A Procedural Approach. Morgan Kaufmann, Burlington (2003)
5. Fletcher, R., Reeves, C.M.: Function minimization by conjugate gradients. Comput. J. **7**(2), 149–154 (1964)
6. Gelas, A., Bernard, O., Friboulet, D., Prost, R.: Compactly supported radial basis functions based collocation method for level-set evolution in image segmentation. IEEE Trans. Image Process. **16**(7), 1873–1887 (2007)
7. Interactive Data Visualization, Inc.: SpeedTree. IDV, 5446 Sunset Blvd. Suite 201 Lexington, SC 29072 (2017). https://store.speedtree.com
8. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: active contour models. Int. J. Comput. Vis. **1**(4), 321–331 (1988)
9. Liang, X., et al.: International benchmarking of terrestrial laser scanning approaches for forest inventories. ISPRS J. Photogramm. Remote Sens. **144**, 137–179 (2018)
10. Morel, J., Bac, A., Véga, C.: Terrain model reconstruction from terrestrial LiDAR data using radial basis functions. IEEE Comput. Graphics Appl. **37**(5), 72–84 (2017)
11. Ohtake, Y., Belyaev, A., Seidel, H.P.: 3D scattered data approximation with adaptive compactly supported radial basis functions. In: Proceedings Shape Modeling Applications 2004, pp. 31–39. IEEE (2004)
12. Osher, S., Fedkiw, R.: Level Set Methods and Dynamic Implicit Surfaces, vol. 153. Springer, Heidelberg (2006)

13. Pharr, M., Jakob, W., Humphreys, G.: Physically Based Rendering: From Theory to Implementation. Morgan Kaufmann, Burlington (2016)
14. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural Information Processing Systems, pp. 5099–5108 (2017)
15. Tsai, R., Osher, S., et al.: Review article: level set methods and their applications in image science. Commun. Math. Sci. **1**(4), 1–20 (2003)
16. Wendland, H.: Scattered Data Approximation. Cambridge University Press, Cambridge (2005)