Deforestation: Extracting 3D Bare-Earth Surface from Airborne LiDAR Data

Wei-Lwun Lu James J. Little Alla Sheffer Hongbo Fu

University of British Columbia
Vancouver, BC, Canada

Abstract

Bare-earth identification selects points from a LiDAR point cloud so that they can be interpolated to form a representation of the ground surface from which structures, vegetation, and other cover have been removed. We triangulate the point cloud and segment the triangles into flat and steep triangles using a Discriminative Random Field (DRF) that uses a data-dependent label smoothness term. Regions are classified into ground and non-ground based on steepness in the regions and ground points are selected as points on ground triangles. Various post-processing steps are used to further identify flat regions as rooftops and treetops, and eliminate isolated features that affect the surface interpolation.

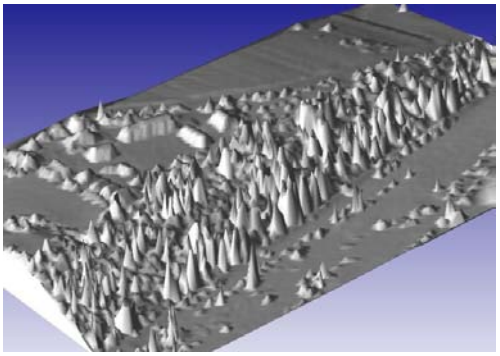
The performance of our algorithm is evaluated in its effectiveness at labeling ground points and, more importantly, at determining the extracted bare-earth surface. Extensive comparison shows the effectiveness of the strategy at selecting ground points leading to good fit in the triangulated mesh derived from the ground points.

1. Introduction

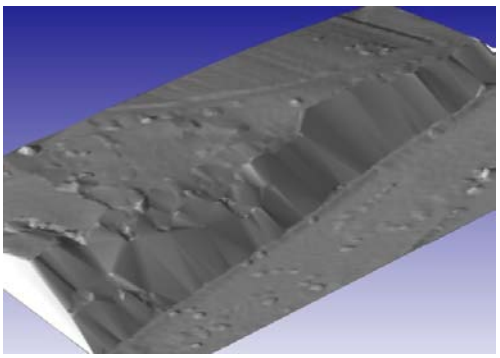
LiDAR (Light Detection and Ranging) systems generally return a three dimensional point cloud containing coordinates corresponding to elevations measured from overhead. In recent years, LiDAR data is increasingly available at high resolution and broad coverage, leading to applications in object recognition, forest measurement, and land use planning. This paper focuses on the analysis of the airborne LiDAR point cloud (Figure 1 (a)), and presents techniques of removing the non-ground objects and extracting the bare-earth surface (Figure 1 (b)).

This paper is motivated by the needs of reconstructing accurate 3D bare-earth surface from airborne LiDAR data. For the production of digital elevation models, the manual classification and quality control pose the greatest challenges, consuming an estimated of 60-80% of processing time [12]. The ability to automatically classify LiDAR points would significantly impact this process and speed up the delivery time.

Identifying ground points from the airborne LiDAR point cloud is challenging. Firstly, the LiDAR point cloud is irregularly sampled, and thus typical image processing techniques cannot be directly applied to analyze the LiDAR point cloud. Secondly, the scenes are usually very complex, consisting of buildings, cars, trees, slopes, rivers, bridges, cliffs, etc. Adequately modeling the ground surface and the non-ground objects is difficult.



(a)



(b)

Figure 1. (a) The original 3D mesh, a Delaunay triangulation of the 3D LiDAR point cloud. (b) The deforested 3D mesh after removing all non-ground data points.

This paper explores the use of probabilistic methods, in particular, Discriminative Random Fields [9], in developing methods for estimating the underlying bare-earth surface hidden in the point cloud of surface observations generated by airborne LiDAR sensors. We set out to use as much local structural information as possible, while avoiding commitments to particular models such as buildings or pre-determined vegetative cover models.

Our model begins with a triangulation of the point cloud. Section 3 describes the underlying model of the triangles and surfaces, and the Discriminative Random Field (DRF). Then it explains how post-processing of the segmentation determined by the DRF can eliminate some structural elements initially misclassified.

Section 4 reports the evaluation of the method regarding classification of all points from a well-known dataset, and comparison of the bare-earth triangulation with manually corrected data from our industrial partners, Terrapoint [15]. We conclude with a description of our implementation platform and comments on future work.

2. Related Work

In order to estimate the bare-earth surface, filtering algorithms [12] are applied to the point cloud to remove the points belonging to non-ground objects. Sithole and Vosselman [12] classify filtering algorithms into four groups: slope-based, block-minimum, surface-based (based on local parametric surface fits), and clustering. Their paper also provides a review of the techniques and a detailed comparison of the performance of the various filtering algorithms.

The surface-based algorithms ([7, 8] among others) assume that the surface is smooth, and that deviations from smoothness represent non-ground points, leading to the despike algorithm which iteratively removes deviations from a locally smooth surface. A widely used software package called SCOP [1] is implemented based on this idea.

However, the robust interpolation algorithm [8] relies on a good mixture of points of earth and non-earth, and it cannot handle the situations of large dense vegetation and large buildings. In order to tackle this problem, Briese *et al.* [3] presented hierarchic robust interpolation that iteratively performs robust interpolation [8] from a coarse-to-fine approach.

The filter developed by Vosselman [17] epitomizes the slope-based filtering approach. Vosselman uses the slopes of a points to its nearby points within a range as a criterion for classifying ground points. If any of its slopes is greater than a predefined threshold, Vosselman classifies the point as an object point. This method is closely related to the erosion operator used for mathematical gray scale morphology.

One of the problem of the slope-based filter [17] is its inability to correctly classify ground points on steep slopes. To tackle this problem, Sithole [11] extended the idea to a slope adaptive filter, which adaptive tunes the threshold according to the *slopes of the terrain*.

Our work, like several presented in that study, uses a triangulation of the point cloud to provide neighborhood information of data points. A triangle-based segmentation and several region-based post-processing techniques are then used to identify the ground points. The algorithm presented by Sohn and Dowman [13] is also based on triangulation. However, in their approach, they simplify and then densify the triangulation to develop a minimal triangulation that approximates a lower envelope of the point cloud.

3. Algorithm

3.1. Overview

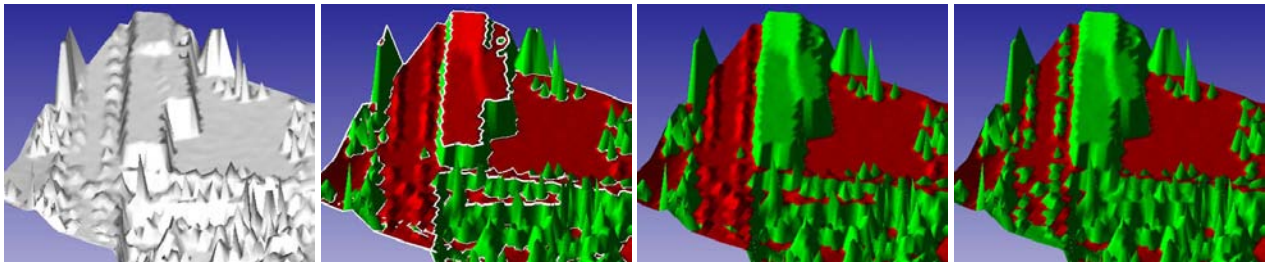
Instead of classifying vertices into ground and non-ground, we decided to work with triangles. Thus, we first triangulate the LIDAR data to obtain a 3D mesh. At the beginning, we assume every triangle belongs to the bare-earth. Then, we apply two algorithms to identify the non-ground triangles. The first algorithm finds the triangles belonging to buildings and high trees. The second algorithm locates the triangles belonging to low trees. Our algorithm then removes these non-ground triangles and performs Delaunay triangulation again to obtain the 3D mesh of the bare-earth.

3.2. Segmentation

The two non-ground triangle detecting algorithms start from segmenting the mesh into steep and flat regions. The feature we use is the *up-angle* of the triangle. The up-angle is defined as the angle between the normal of the triangle and the vector pointing to the sky (i.e., $(x, y, z) = (0, 0, 1)$). As a result, a flat triangle has a small up-angle while a steep triangle has a large up-angle. The segmentation can be achieved by first classifying each triangle into either steep or flat, and then clustering all nearby triangles with the same category into a single region.

Classifying triangles into steep and flat ones can be solved by minimizing an energy function inspired by the binary image segmentation algorithm presented in [14]. We construct a Discriminative Random Field [9] (a variant of the Conditional Random Field (CRF) [10]) to model the relationship between the observed 3D mesh and the categories of triangles. In the Discriminative Random Field, we construct a graph where nodes represent triangles and edges connecting two neighboring triangles. [1] The energy function has two kinds of potentials. Let l_p be the label of triangle p (in our case, the label can be either steep or flat).

[1] Two triangles are neighbors if they share the same edge.



(a) (b) (c) (d)

Figure 2. (a) The original 3D mesh. (b) Segmenting triangles into steep (green) and flat (red) regions. (c) Classification results after locating buildings and high trees (d) Classification results after locating low trees. In (c) and (d), red-colored regions represent the bare-earth while green-colored regions represent the non-ground objects.

The unary potential $D_p(l_p)$ measures the likelihood of label l_p given the observed features from the data, which is also known as the data cost. The pairwise potential $V(l_p, l_q)$ penalizes the difference of l_p and l_q, which can be considered as the smoothness cost. Specifically, the energy function is defined as:

$$E = \sum_p D_p(l_p) + \sum_{p,q \in N} w_{pq} V(l_p, l_q) \qquad (1)$$

where w_{pq} is a data-dependent weighting function and $p, q \in N$ means p and q are neighboring triangles.

The unary potential $D_p(l_p)$ measures the likelihood of label l_p given the observed features from the data. In particular, we define the data energy cost $D_p(l_p)$ as:

$$D_p(l_p) = \begin{cases} \|S(p) - \mu_{steep}\|^2 / \mu_{steep}^2 & \text{if } l_p = \text{steep} \\ \|S(p) - \mu_{flat}\|^2 / \mu_{flat}^2 & \text{if } l_p = \text{flat} \end{cases} \qquad (2)$$

where $S(p)$ is the up-angle of triangle p. The quantity $(\mu_{steep}, \sigma_{steep})$ and $(\mu_{flat}, \sigma_{flat})$ denote the mean and variance of the up-angles of the flat and steep triangles, respectively.

The pairwise potential $V(l_p, l_q)$ is a standard Potts model which penalizes the difference between l_p and l_q. Specifically, $V(l_p, l_q)$ is defined as:

$$V(l_p, l_q) = \begin{cases} \lambda_1 & \text{if } l_p \neq l_q \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

where λ_1 is a smoothness constant specified by the user. The standard Potts model favors smooth label assignment and ignores the observed data, and therefore the edge between the flat and steep regions will be over-smoothed. In order to tackle this problem, we introduce a data-dependent weighting function w_{pq} that reduces the influence of the Potts model on the edges. The weighting function w_{pq} is defined as:

$$w_{pq} = \exp(-\beta \|S(p) - S(q)\|^2) + \lambda_2 \qquad (4)$$

where $S(p)$ and $S(q)$ are the up-angles of the triangles p and q. The quantity of β is set to $(2\langle \|S(p) - S(q)\|^2 \rangle)^{-1}$ where the expectation denotes an average over the mesh. The purpose of λ_2 is to remove small and isolated areas that have high up-angle contrast.

The minimization of the energy function Eq. (1) can be solved exactly because l_p is binary, i.e., a triangle can be either steep or flat. We use graphcut [2] to minimize Eq. (1) because it is very efficient [14] and guaranteed to converge to the global optimum in the binary case.

After classifying every triangle into either steep or flat, the next step is to cluster nearby triangles with the same category into a single region. Figure 2 (b) illustrates the segmentation results. Observe that steep triangles consist of high trees, walls of buildings, and cliffs; while the flat triangles consist of bare-earth, roof-tops, and some low trees. Therefore, an intuitive approach is to first assume all steep triangles belong to non-ground and all flat triangles belong to bare-earth, and then apply a sequence to heuristics to refine the ground/non-ground classification. The following sections will discuss techniques of locating roof-tops and low trees.

3.3. Detecting Buildings and High Trees

This section focuses on the techniques of detecting buildings and high trees and classifying them as non-ground triangles.

By looking at Figure 4 (b), we can observe that a building consists of walls and roof-tops. The walls are usually steep regions with large up-angles while the roof-tops are flat regions with small up-angles. Similarly treetops often in dense groves of trees appear as relatively flat regions, classified as flat, surrounded by steep triangles. However, the

mountain hills in Figure 4 (a) also have the same characteristics. The difference between buildings and mountain hills is that the slopes of mountains are not very steep, i.e., their up-angles are not as large as those of walls.

In order to differentiate between walls of buildings and slopes of mountains, we first run our segmentation algorithm with $\mu_{steep} = 80$, $\sigma_{steep} = 10$, $\mu_{flat} = 10$, $\sigma_{flat} = 10$, $\lambda_1 = 10$, and $\lambda_2 = 1$. The large μ_{steep} yields a classification of walls and high trees as steep regions, and leaves slopes and low trees as flat regions. Then we assume that all steep triangles belong to non-ground and all flat triangles belong to ground, and then search the ground regions for the roof-tops and re-classify them as non-ground.

In order to differentiate between roof-tops and bare-earth, we observe that the roof-top regions usually have higher elevation than their surrounding triangles. Thus, the *relative height* $H_{rel}(r)$ of a ground region r can be defined as $H_{rel}(r) = H_g(r) - H_n(r, w)$ where $H_g(r)$ is the average height of a ground region r and $H_n(r, w)$ is the average height of surrounding triangles of ground region r within a specified width w. By simply thresholding the relative height, we can effectively distinguish roof-tops from bare-earth.

Step-like structures in buildings can be located in a similar way. Rooftops of lower portions of buildings appear as flat regions with some surrounding walls higher than themselves and some surrounding walls lower than themselves. Therefore, we classify a ground region as steps if at least 20% of its surrounding triangles are 30 cm higher and at least 20% of its surrounding triangles are 30 cm lower than the ground region. We found out that this simple criterion works nicely in all our test datasets.

Figure 2 (c) shows the results after re-classifying the roof-tops as non-ground. We can observe that high trees and buildings (including their roof-tops) are correctly classified as non-ground objects while regions with low trees are still classified as bare-earth.

3.4. Detecting Low Trees

This section focuses on techniques for detecting low trees and forests. Isolated low trees are small cone-shaped structures. To distinguish isolated low trees from bare-earth, we run our segmentation algorithm again with $\mu_{steep} = 60$, $\sigma_{steep} = 10$, $\mu_{flat} = 10$, $\sigma_{flat} = 10$, $\lambda_1 = 3$, and $\lambda_2 = 0$, and then re-classify every steep region with area smaller than a threshold as a non-ground region. In order to locate the tree-tops, we run the roof-top detection algorithm again, but in this time, we enforce a new constraint that the area of tree-tops should be smaller than a threshold. These two criteria can effectively detect and remove isolated low trees from the bare-earth (Figure 2 (d)).

The most challenging part is to differentiate a low forest (a large region of connected low trees) from the slopes of mountains. The area of both low forests and mountain slopes are large, and their up-angles are similar. Fortunately, we can observe that the normals of a tree's triangles point to many directions, while the normals of slope's triangles usually point to a single direction. To utilize this observation, we first compute the variance of normals of steep regions and denoted it as (ν_x, ν_y, ν_z). If $(\nu_x + \nu_y)/2$ is greater than a threshold, then we re-classify the steep regions as non-ground, otherwise, they remain as bare-earth regions. Although simple, this criterion works nicely and further remove the non-ground objects that cannot be detected in the previous steps.

4. Evaluation

In order to evaluate the performance of the proposed algorithm, we test our system in two datasets: the Sithole *et al.* [12] dataset and the Terrapoint dataset [15]. We use the same parameter settings for all experiments in these two datasets.

4.1. Sithole *et al.* Dataset

The Sithole *et al.* dataset [12] consists of 15 sites with various terrain characteristics including buildings, steep slopes, bridges, terrain discontinuities, ramps, vegetation on slopes and many others (see the second column of Table 1 for a detailed description). Sithole *et al.* manually classified each data point well.

We evaluate the quantitative performance of our system by the classification errors and the distance between the extracted and ground-truth bare-earth surface. Note that the major goal of the system is to extract the bare-earth surface, and classifying triangles into ground/non-ground is just an intermediate step towards achieving this goal.

We evaluate the classification performance by Type I, Type II, and Total Errors. To convert classified points to classified triangles in the Sithole *et al.* dataset, we label a triangle as ground if all of its vertices are labeled as ground; otherwise, it is labeled as a tree triangle. Letting E_1 be the number of ground triangles that our algorithm mistakenly classifies as non-ground and E_2 be the number of non-ground triangles that our algorithm mistakenly classifies them as ground, the classification errors are defined as:

$$Err_1 = \frac{E_1}{N_1} \qquad Err_2 = \frac{E_2}{N_2} \qquad Err = \frac{E_1 + E_2}{N_1 + N_2} \quad (5)$$

where N_1 is the number of ground triangles and N_2 is the number of non-ground triangles. The quantities Err_1, Err_2, and Err denote the Type I, Type II, and Total Error, respectively.

Name	Special Features	# points	Type I Error	Type II Error	Total Error	Avg. Distance
1-1	vegetation & buildings on steep slopes	37937	51.75%	1.28%	21.49%	44.28 cm
1-2	buildings and cars	51984	16.65%	2.54%	8.15%	11.58 cm
2-1	narrow bridge	12910	12.71%	9.60%	11.65%	4.99 cm
2-2	bridges & gangway	32595	13.54%	9.51%	11.95%	9.67 cm
2-3	large buildings & disconnected terrain	25056	16.54%	4.14%	9.40%	11.27 cm
2-4	ramp	7469	20.58%	4.93%	14.10%	8.17 cm
3-1	large buildings	28805	7.50%	2.33%	4.57%	6.26 cm
4-1	outliers (multi-path error)	11160	23.42%	2.71%	11.65%	54.33 cm
4-2	rail station	42399	9.07%	3.09%	4.42%	42.30 cm
5-1	vegetation on slope	17845	5.84%	7.32%	6.40%	9.18 cm
5-2	slope	22474	7.59%	25.31%	10.97%	9.33 cm
5-3	disconnected terrain (cliffs)	34348	20.13%	23.05%	20.39%	12.61 cm
5-4	low resolution buildings	8608	6.90%	6.23%	6.40%	15.87 cm
6-1	sharp ridge & ditches	35060	6.63%	7.41%	6.69%	5.00 cm
7-1	bridge & terrain discontinuities	15645	1.47%	44.75%	9.59%	10.29 cm

Table 1. Quantitative evaluation of the Sithole *et al*. dataset [12].

Name	Special Features	# points	Avg. Distance
1	vegetations & roads	1347446	12.50 cm
2	buildings & cars	2797040	18.63 cm
3	vegetation	9830323	9.78 cm

Table 2. Quantitative evaluation of the Terrapoint dataset [15].

To measure the distance between the estimated and ground-truth bare-earth surface, we define the distance $dist(p, S)$ between a point p and a surface S as:

$$dist(p, S) = \min_{p' \in S} \|p - p'\| \tag{6}$$

The average distance between surfaces S_1 and S_2 thus can be defined as

$$dist_{avg}(S_1, S_2) = \frac{1}{|S_1|} \int_{p \in S_1} dist(p, S_2) dp \tag{7}$$

where $1/|S_1|$ is the area of S_1. In particular, we use a standard package named Metro [4] to compute the average distance between two 3D meshes.

Table 1 shows the quantitative performance of our system. The extracted bare-earth surfaces are usually very good, with average distance around 10 cm. Observe that the Type II errors (mistakenly classifying tree triangles as ground) are usually smaller than the Type I errors. This phenomenon is due to the fact that Type II errors usually have more negative effects on the final extracted bare-earth surface, and thus we focus more on minimizing the Type II errors. However, Type I errors simply reduce the amount of detail in the bare-earth surface. Another interesting observation is that good classification performance sometimes does not translate to a good extracted bare-earth surface (e.g., site 4-2), and vice versa (site 5-3). For instance, a small mis-classification on the roof of a building may significantly pollute the quality of the extracted bare-earth surface, while mis-classifying a lower tree as bare-earth does not influence the bare-earth surface that much.

Figure 4 shows the qualitative results of our system. Our algorithm can nicely deal with most of the cases including buildings, vegetation, slopes, vegetation on slopes, ramps, and cliffs. The major difficulties we encounter are buildings on slope (site 1-1), large pits on the roof-tops (site 4-2), and bridges (Figure 4 (c)). Bridges are a known problem in bare-earth classification [12].

4.2. Terrapoint Dataset

The Terrapoint data [15] consists of three huge sites with millions of data points. The first and third sites contain vegetation and roads, while the second site is composed of forests, buildings, and cars. Unfortunately, Terrapoint classifies dataset in a conservative way, i.e., they mark few points as ground in order to increase the quality of the bare-earth extraction. As a result, the ground-truth classification is not accurate because many ground points are classified as non-ground. However, in general, this is a wise strategy since, as noted above, incorrect classification of a non-ground point as ground can significantly affect the accuracy

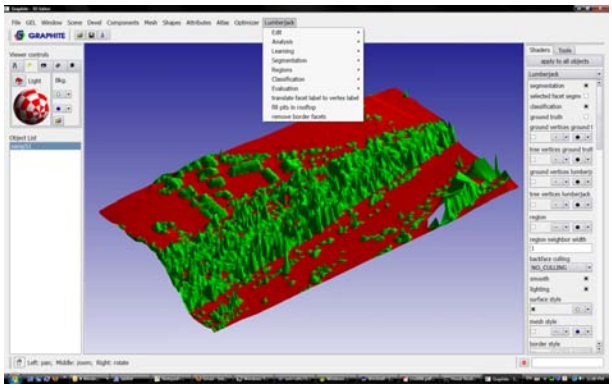


Figure 3. The Graphite [5] environment and the Lumberjack Toolbox.

of the extracted bare-earth surface.

Table 2 shows the quantitative performance of our system on the Terrapoint dataset. Our algorithm is especially accurate when there is only vegetation (site 1 and 3). In the case of buildings, cars, and forest (site 2), our system can still work effectively. Figure 5 visualizes the results of our system. Observe that our system can detect and remove trees and buildings to obtain an accurate estimation of the bare-earth surface.

5. Implementation

We implemented the entire system in C++ and developed a toolbox called *Lumberjack* for Graphite [5], a research software platform for computer graphics, 3D modeling and numerical geometry. Figure 3 displays a snapshot of the Graphite environment and the Lumberjack toolbox. The users can utilize Graphite and Lumberjack to visualize the 3D mesh of the surface, running the proposed bare-earth extraction algorithm, tuning the parameters, and visualize the results in an interactive way.

6. Conclusion and Future Work

We have demonstrated a bare-earth identification system based on segmentation of triangulated LiDAR point clouds. A Discriminative Random Field segments the surface into steep and flat regions of triangles using data-dependent label smoothness term. Regions are classified into ground and non-ground based on steepness in the regions and ground points are selected as points on ground triangles. Various post-processing steps are used to further identify flat regions as rooftops and treetops, and eliminate isolated features that affect the surface interpolation.

The performance of our algorithm is evaluated in its effectiveness at labeling ground points and, more importantly, at determining the extracted bare-earth surface. Extensive comparison shows the effectiveness of the strategy at selecting ground points leading to good fit in the triangulated mesh derived from the ground points.

Sithole and Vosselman [12] argued that the most successful filters for deriving bare earth involve local estimation of the surface over a region of some size. We agree and plan to extend our analysis to incorporate larger context, most likely by a coarse to fine analysis.

Complex cityscapes form a true challenge to these filtering methods. In order to address the problem of identifying structures, much more specific model-based information can be applied, i.e., verticality, rectangularity, and parallelism. Much progress has already been made[16, 6], in which local fitting of simple parametric surfaces suggests structures. These same fits can select slope regions as well.

7. Acknowledgments

This work has been supported by grants from the GEOIDE Network of Centres of Excellence and Terrapoint Canada Inc. Thanks to Ciaran Llachlan Leavitt for her assistance in developing our software.

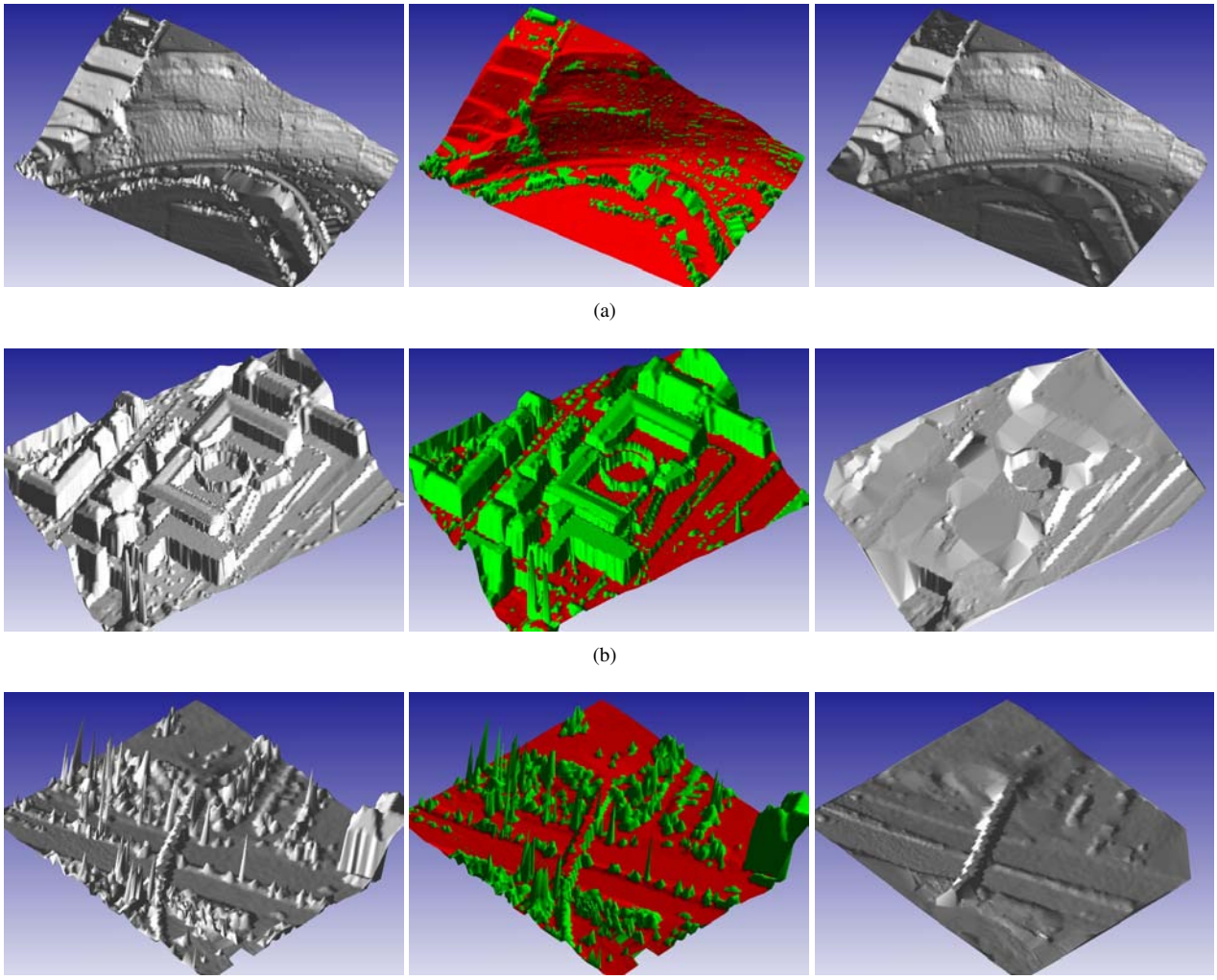
References

[1] http://www.ipf.tuwien.ac.at/products/produktinfo/scop/.

[2] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[3] C. Briese, N. Pfeifer, and P. Dorninger. Applications of the robust interpolation for DTM determination. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXIV, 3A*, pages 55–61, 2002.

[4] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces. In *Computer Graphics Forum*, volume 17, pages 167–174, 1998.



(a)

(b)

(c)

Figure 4. Qualitative evaluation of the Sithole *et al*. dataset [12]. The first column is the original 3D mesh. The second column is the classification results. The third column is the deforested 3D mesh. In all figures, red-colored regions represent the bare-earth while green-colored regions represent the non-ground objects.

[5] Graphite, 2003. http://www.loria.fr/ levy/Graphite/index.html.

[6] F. Han, Z. W. Tu, and S. C. Zhu. Range Image Segmentation by an Effective Jump-Diffusion Method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1138–1153, 2004.

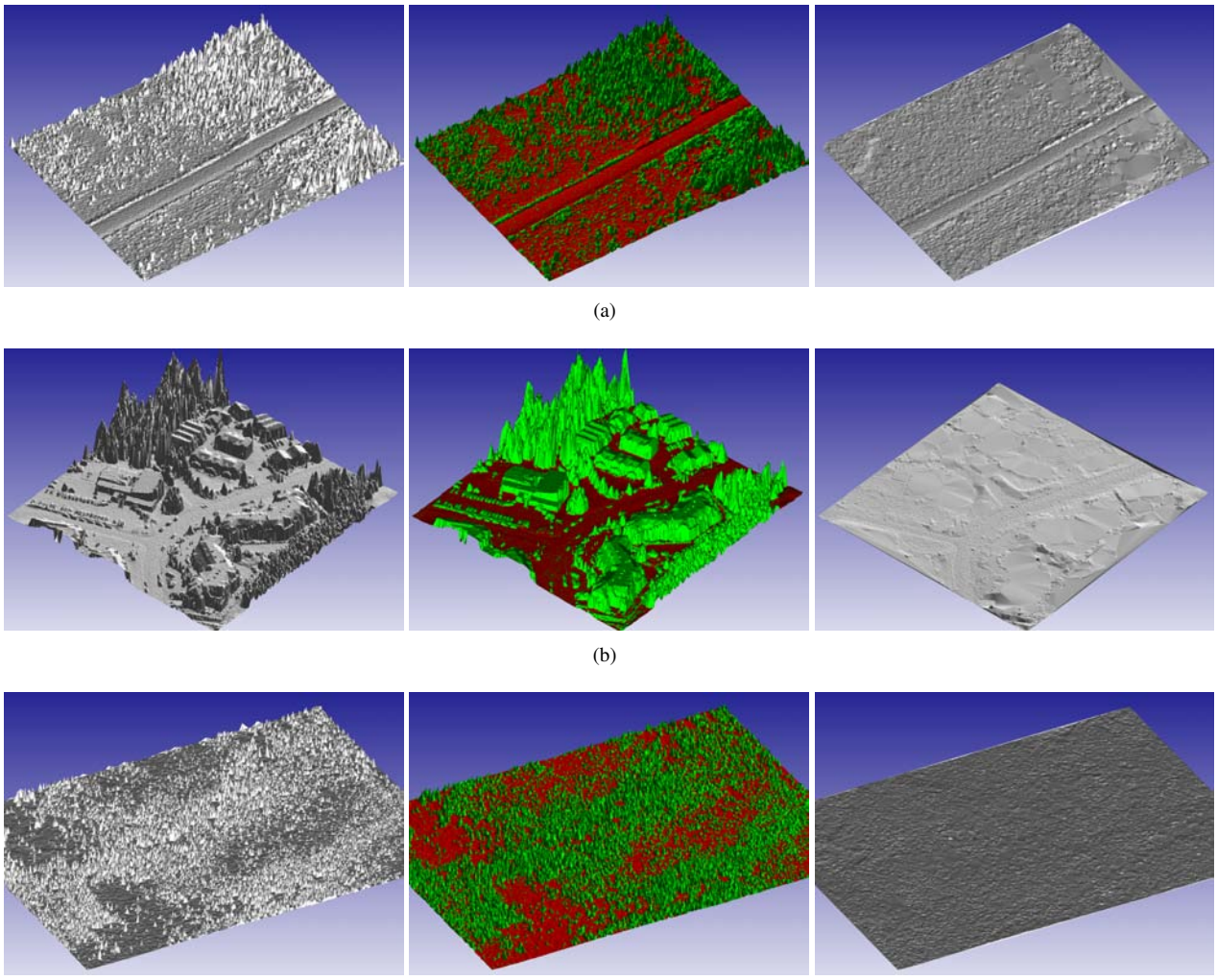
[7] R. A. Haugerud and D. J. Harding. Some algorithms for virtual deforestation (VDF) of LIDAR topographic survey data. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXIV Pt. 3/W4*, pages 211–218, 2001.

[8] K. Kraus and N. Pfeifer. Determination of terrain models in wooded areas with airborne laser scanner data. *ISPRS Journal of Photogrammetry & Remote Sensing*, 53:193–203, 1998.

[9] S. Kumar and M. Hebert. Discriminative Random Fields: A Discriminative Framework for Contextual Interaction in Classification. In *Proceedings of the 9th IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157, 2003.

[10] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, 2001.

[11] G. Sithole. Filtering of laser altimetry data using a slope adaptive filter. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXIV, 3/W4*, pages 203–210, 2001.



(a)

(b)

(c)

Figure 5. Qualitative evaluation of the Terrapoint dataset [15]. The first column is the original 3D mesh. The second column is the classification results. The third column is the deforested 3D mesh. In all figures, red-colored regions represent the bare-earth while green-colored regions represent the non-ground objects.

[12] G. Sithole and G. Vosselman. Experimental comparison of filter algorithms for bare-Earth extraction from airborne laser scanning point clouds. *ISPRS Journal of Photogrammetry & Remote Sensing*, 59:85–101, 2004.

[13] G. Sohn and I. Dowman. Terrain surface reconstruction by the use of tetrahedron model with the MDL Criterion. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXIV*, pages 336–344, 2002.

[14] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A Comparative Study of Energy Minimization Methods for Markov Random Fields with Smothness-Based Priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, to appear.

[15] Terrapoint. http://www.terrapoint.com/.

[16] V. Verma, R. Kumar, and S. Hsu. 3D Building Detection and Modeling from Aerial LIDAR Data. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2213–2220, 2006.

[17] G. Vosselman. Slope based filtering of laser altimetry data. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXIII, B3*, pages 935–942, 2000.