A Hybrid Conditional Random Field for Estimating the Underlying Ground Surface from Airborne LiDAR Data

Wei-Lwun Lu, Kevin P. Murphy, James J. Little, Alla Sheffer, and Hongbo Fu

***Abstract*—Recent advances in airborne light detection and ranging (LiDAR) technology allow rapid and inexpensive generation of digital surface models (DSMs), 3D point clouds of buildings, vegetations, cars, and natural terrain features over large regions. However, in many applications, such as flood modeling and landslide prediction, digital terrain models (DTMs), the topography of the bare-earth surface, are needed. This article introduces a novel machine learning approach to automatically extract DTMs from their corresponding DSMs. We first classify each point as being either ground or non-ground, using supervised learning techniques applied to a variety of features. For the points which are classified as ground, we use the LiDAR measurements as an estimate of the surface height, but for the non-ground points, we have to interpolate between nearby values, which we do using a Gaussian random field. Since our model contains both discrete and continuous latent variables, and is a discriminative (rather than generative) probabilistic model, we call it a *hybrid Conditional Random Field*. We show that a MAP estimate of the surface height can be efficiently estimated by using a variant of the Expectation Maximization (EM) algorithm. Experiments demonstrate that the accuracy of this learning-based approach outperforms the previous best systems, based on manually tuned heuristics.**

***Index Terms*—Light detection and ranging (LIDAR) data filtering, digital terrain model (DTM), conditional random fields.**

I. Introduction

RECENT advances in airborne light detection and ranging (LiDAR) technology allow rapid and inexpensive generation of digital surface models (DSMs), 3D point clouds of buildings, vegetations, cars, and natural terrain features over large regions. However, in many applications, such as flood modeling and landslide prediction, digital terrain models (DTMs) are needed. Here, the DTM refers to natural terrain features of the bare-earth surface, excluding points from vegetations, buildings, cars, or any other man-made structures. In order to generate DTMs, non-ground points of DSMs have to be identified, and their underlying bare-earth elevations have to be estimated (See Fig. 1 for an illustration). This article introduces a novel machine learning approach to automatically extract DTMs from DSMs.

The DTM extraction problem is of great commercial interest, but it remains a challenging task. Many companies adopt a semi-automatic approach: they first use automatic bare-earth extraction algorithms to obtain initial DTMs, and then experienced operators come in to carefully identify and remove mis-classified points. This human editing process takes about 60-80% of the processing time for the production of DTMs [1]. Therefore, improving the accuracy of automatic bare-earth extraction algorithms would significantly impact this process and speed up the delivery time.

The DTM extraction problem is challenging for several reasons. Firstly, the LiDAR point cloud is irregularly sampled, and thus typical image processing techniques cannot be directly applied. Secondly, the scenes are usually very complex, consisting of buildings, cars, trees, slopes, rivers, bridges, cliffs, etc. For example, Fig. 2 shows a profile of a DSM with one building and some trees on a slope. We cannot simply declare points with high elevation to be non-ground, because hills have even higher elevation. Similarly, we cannot simply declare points in flat regions to be ground, because roof-tops are also flat regions. Hence, developing an algorithm that is able to deal with different kinds of terrain (urban and rural, flat and sloped, etc.) is a challenging task.

In this article, we introduce a novel machine learning approach to automatically extract DTMs from DSMs. In particular, we propose to model the problem as a hybrid Conditional Random Field. Conditional Random Fields (CRF) [2], [3] are a variant of Markov Random Fields, and are widely used in segmentation and recognition problems in computer vision and machine learning. Most CRFs assume all the hidden random variables are discrete. Our model is novel in that it contains both discrete *and* continuous hidden random variables; this is why we call it a "hybrid" CRF. The discrete random variables are binary, and represent whether a point is ground or non-ground. The continuous random variables represent the height of the underlying ground surface at that point. In addition, we have the observed variables, which are the LiDAR measurements. The correlations between all the variables are learned from a training set. We explain the model and training procedure in detail in Section 3.

Once the model has been trained, we can use it to estimate the ground surface (the hidden continuous variables) from the LiDAR data (the observed continuous variables). We show how to do this efficiently using a variant of the Expectation Maximization (EM) algorithm [4]. See Section 4 for the details.

Our experimental results in Section 5 show that, given suf-

W.L. Lu, K. Murphy, J. Little, and A. Sheffer are with the Department of Computer Science, University of British Columbia, BC, Canada V6T1Z4 (email: vailen,murphyk,little,sheffa@cs.ubc.ca).

H. Fu is with the School of Creative Media, City University of Hong Kong, Hong Kong, China (email: hongbofu@cityu.edu.hk).

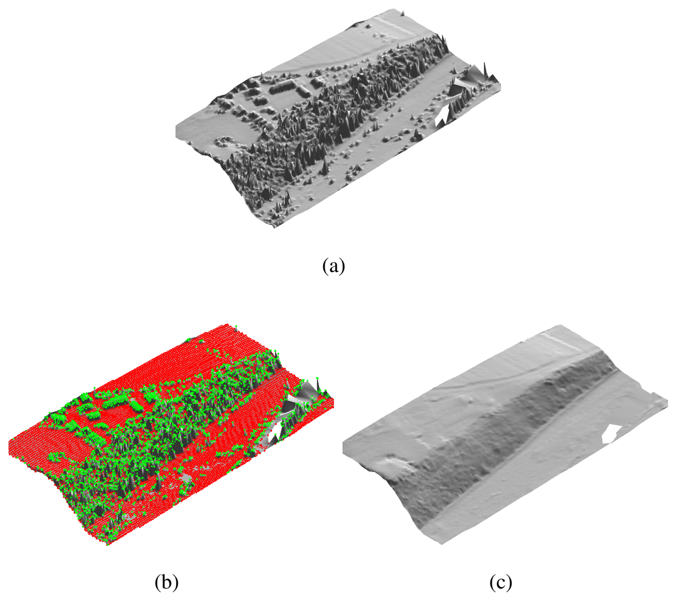


Fig. 1. Illustration of the DTM extraction problem. (a) The original DSM of a rural site with vegetation on a slope, and some small buildings. (b) The classification of 3D points to ground/non-ground (red/green). (c) The extracted DTM of the bare-earth surface. The small white patch near the top right corresponds to missing data.

ficient training data, the trained model can deal with different terrain types including dense vegetation, mountain, and urban regions. Furthermore, quantitative results using the standard dataset [1] demonstrate that the proposed algorithm outperforms the previous best systems (based on manually tuned heuristics) at both point classification and DTM extraction. We conclude in Section 6.[1]

II. Related Work

The bare-earth extraction problem has been studied extensively in the remote sensing community, and many algorithms have been proposed. Sithole and Vosselman [1] provides a review of the techniques and a detailed comparison of their performance.

Sithole and Vosselman classified the algorithms into four groups: slope-based [6]–[8], block-minimum [9], surface-based [10]–[13], and clustering [14]. The surface-based algorithms [10]–[13] assume that the surface is smooth, and that deviations from smoothness represent non-ground points, leading to the de-spike algorithm which iteratively removes deviations from a locally smooth surface. The slope-based approaches [6]–[8] use the slope of a point to its nearby points within a range as a criterion for classifying ground points. These methods are closely related to morphological operators, but the window size has to be carefully chosen in order to deal with different kinds of terrain. In order to deal with this problem, Dell'Acqua *et al.* [15] proposed to optimize the window size using some training data. Recently, Zhang *et al.* [16] introduced a progressive morphological filter where the window size can be changed adaptively. This technology has been also used to automatically construct building footprints from airborne LIDAR data [17]. Similar to morphological operators, curvature analysis can be used to identify non-ground points [18], but it only works for forested regions. The clustering approach is based on the idea that points belonging to non-ground clusters are higher than their neighbors [14], or the statistics of points over local regions is different [19]. In our framework, many of these ideas are used to create appropriate *features* for each 3D point. We then apply a powerful machine learning approach to figure out a mapping from the feature vector to a point classification.

[1]This article extends our previous work in [5] in the following main ways: we switch from triangle-based features to point-based features, we use a new hybrid model, we use a new inference algorithm, and we perform much more extensive experimentation.

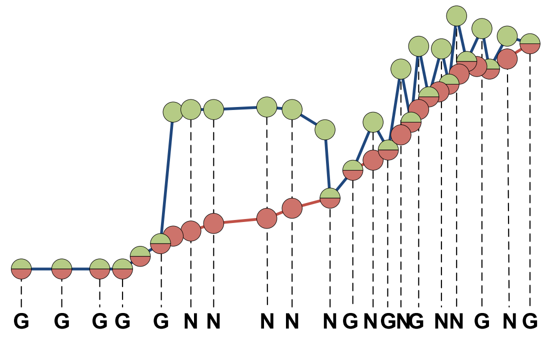


Fig. 2. An 1D profile of a DSM with one building and some trees on a slope. The goal is to estimate the true bare-earth surface (red points) from the measurements (green points). At the bottom of the figure, we have classified each point as G for ground or N for non-ground. This is part of the training data, but is not available at test time.

The problem of segmenting or classifying 3D point clouds has also been studied in the computer vision community. For example, [20] considered the problem of segmenting 3D point clouds using a generative model, where each region consists of a parametric model of the surface, such as planes, cylinders, cones, etc. Although we use segmentation in our approach (in order to derive segment-based features), we are not interested in segmentation *per se*. Baillard and Maitre [21] used a Markov Random Field and an iterative algorithm to classify 3D points, but their input was pairs of stereo images instead of LiDAR images. The problem of classifying 3D point clouds was also studied in [22] using a max-margin Markov network. This is similar to our work, but does not address the issue of estimating the underlying surface (the model only contains discrete label variables, not continuous height variables). Secord *et al.* [23] introduced a technique to detect trees using a support vector machine [24]. However, they used both airborne LiDAR and aerial images, and did not enforce spatial correlations between nearby points.

III. Model

A. Overview

A typical LiDAR measurement consists of the 3D location of the reflecting surface, the intensities of the return pulse, and the type of returns (i.e., first, middle, or last returns). In this article, we only use the 3D locations of the last returns received from the LiDAR sensor, and we omit other information such as intensity of the return pulses.[2]

Let us now define our notation. Let $\mathbf{O}_i \in \mathbb{R}^3$ be the (x, y, z) coordinates of point i, and let $\mathbf{O} = (\mathbf{O}_1, \ldots, \mathbf{O}_n)$ be all points in the observed DSM, where n is the number of points. For every sampled point i, we have a random variable $c_i \in \{0, 1\}$ indicating the point classification, i.e., whether the point belongs to the ground ($c_i = 0$) or non-ground ($c_i = 1$). We also have a random variable $g_i \in \mathbb{R}^+$ representing the estimated elevation of the bare-earth surface in the DTM at point i. Let $\mathbf{c} = (c_1, \ldots, c_n)$ and $\mathbf{g} = (g_1, \ldots, g_n)$ be all the unknown variables in a dataset. Note that $\mathbf{c}$ and $\mathbf{g}$ are unknown variables, and our goal is to estimate $\mathbf{g}$ (the DTM) given the observed $\mathbf{O}$ (the DSM), where the point classifications $\mathbf{c}$ serve as intermediate states to help us estimate $\mathbf{g}$ from $\mathbf{O}$.

Below we define the conditional probability over $\mathbf{g}$ (the DTM) and $\mathbf{c}$ (the point classification) of all points in the dataset given $\mathbf{O}$ (the observed DSM):

$$p(\mathbf{g}, \mathbf{c}|\mathbf{O}, \boldsymbol{\theta}) = p(\mathbf{g}|\mathbf{c}, \mathbf{O}, \boldsymbol{\lambda})\ p(\mathbf{c}|\mathbf{O}, \mathbf{w}) \tag{1}$$

where $\boldsymbol{\theta} = (\boldsymbol{\lambda}, \mathbf{w})$ are all the parameters in the model. Fig. 3 illustrates the proposed probabilistic model. The term $p(\mathbf{c}|\mathbf{O}, \mathbf{w})$ models the conditional probability over point classification given the observed DSM, and can be interpreted as "determine whether a point is ground or non-ground based on the observed DSM". The term $p(\mathbf{g}|\mathbf{c}, \mathbf{O}, \boldsymbol{\lambda})$ models the conditional probability over the estimated DTM given the point classification and observed DSM, and can be interpreted as "estimate the bare-earth surface given the observed DSM, assuming the point classification is known". Notice that $p(\mathbf{g}|\mathbf{c}, \mathbf{O}, \boldsymbol{\lambda})$ also enforces spatial correlations amongst nearby points, similar to Markov Random Fields (MRFs). However, since we only model the *conditional* probability, the proposed model is a *Conditional Random Field* (CRF) [2], [3]. Moreover, since the model consists of discrete random variables (point classification) and continuous random variables (the estimated DTM), we call it a *hybrid* CRF.

Below we explain the two conditional probabilities, $p(\mathbf{g}|\mathbf{c}, \mathbf{O}, \boldsymbol{\lambda})$ (height field) and $p(\mathbf{c}|\mathbf{O}, \mathbf{w})$ (label field), in more detail.

B. Height field

We define the conditional probability over the estimated DTM based on the following intuition: If we know point i is a ground point, then we want its estimated bare-earth elevation in the DTM to be close to the observed elevation in the DSM. On the other hand, if we know point i is a non-ground point, we want its estimated bare-earth elevation in the DTM to be similar to the heights of its neighbors (a locally smooth assumption). This idea is inspired by the *feature-preserving mesh smoothing* techniques in the computer graphics community [25], [26].

More formally, we define the following model:

$$p(\mathbf{g}|\mathbf{c}, \mathbf{O}, \boldsymbol{\lambda}) = \mathcal{N}(\mathbf{g}|\mathbf{0}, \tfrac{1}{\lambda_1}\boldsymbol{\Sigma}(\mathbf{c}))\ \prod_{i=1}^{n} \mathcal{N}(g_i|z_i, \tfrac{1}{\lambda_0})^{1-c_i} \tag{2}$$

[2]We chose last returns instead of first, since last returns are more likely to be from the ground surface. Extending the model to handle multiple returns is an interesting direction for future work.

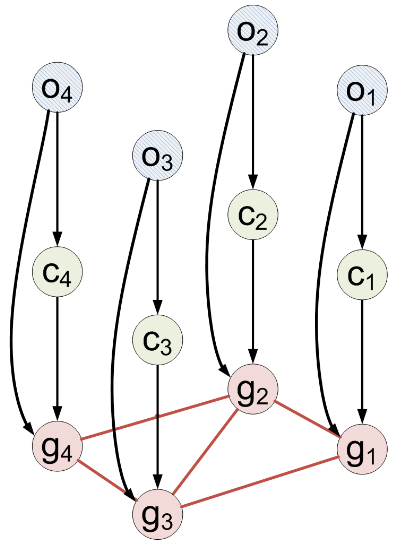


Fig. 3. Sketch of the hybrid conditional random field. In the probabilistic model, $\mathbf{o}_i$ is the observed 3D point in the DSM, c_i is its point classification, and g_i is the estimated bare-earth elevation in the DTM. Links correspond to the variable on which the distribution is conditioned.

where $\mathcal{N}$ represents a multivariate Gaussian, λ_1 is the precision of the Gaussian smoothing prior, and λ_0 is the precision of the observations. $\boldsymbol{\Sigma}(\mathbf{c})$ is a covariance matrix that depends on the labels. defined below. Note that, if $c_i = 0$ (ground point), the second term forces the corresponding g_i to be close to the observed height z_i, whereas if $c_i = 1$, g_i is only constrained by the other g's, not by the data. (Recall that z_i is the height component of the $\mathbf{o}_i$ measurement vector.)

We now define $\boldsymbol{\Sigma}(\mathbf{c})$, which controls the smoothness of the DTM. Our method is similar to an intrinsic Gaussian random field [27]. First we construct a 2D Delaunay triangulation of the points, based on their (x, y) values. Let $\mathbf{L}$ be the $n \times n$ Laplacian matrix defined as

$$L_{i,j} = \begin{cases} -1 & \text{if } i = j \\ \frac{1}{n_i} & \text{if } j \in \mathcal{N}_i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $\mathcal{N}_i$ are the neighbors of point i, and $n_i = |\mathcal{N}_i|$ is the number of such neighbors. Let $\mathbf{C} = \mathrm{diag}(\mathbf{c})$. Then we define

$$\boldsymbol{\Sigma}(\mathbf{c})^{-1} = \mathbf{L}^T \mathbf{C} \mathbf{L} \tag{4}$$

This enforces smoothness between neighboring points in regions where we have no data on the true bare-earth height, but allows for discontinuities when the bare-earth is "visible".

Eq. (2) can be better understood if we look instead at the negative log probability, which is given by the following expression (dropping irrelevant constants):

$$\begin{aligned} &-\log p(\mathbf{g}|\mathbf{c}, \mathbf{O}, \boldsymbol{\lambda}) \\ &\quad \propto \lambda_1 (\mathbf{L}\mathbf{g})^T \mathbf{C} (\mathbf{L}\mathbf{g}) + \lambda_0 (\mathbf{g} - \mathbf{z})^T (\mathbf{I} - \mathbf{C})(\mathbf{g} - \mathbf{z}) \\ &\quad = \sum_{i=1}^{n} \lambda_1 c_i \Big[g_i - \frac{1}{n_i} \sum_{j \in \mathcal{N}_i} g_j \Big]^2 + \lambda_0 (1 - c_i)[g_i - z_i]^2 \end{aligned} \tag{5}$$

This can be interpreted as follows: if $c_i = 1$, meaning point i is not a ground point, then we force the bare-earth elevation to be similar to the average of its neighbors, and thus produce a locally smooth surface (from the first term in Eq. (5)); otherwise, we force the bare-earth elevation to be similar to the observed elevation from DSMs (from the second term in Eq. (5)). Notice that $\lambda = \lambda_0/\lambda_1$ determines how much we trust our data relative to the strength of our prior. Without loss of generality, we fix $\lambda_1 = 1$ and $\lambda_0 = 100,000$ for all experiments in this article. This gave good results over a variety of terrain types.

C. Label field

We now define $p(\mathbf{c}|\mathbf{O},\mathbf{w})$, the conditional probability over point classification given the observed DSM. We use the following simple model:

$$p(\mathbf{c}|\mathbf{O},\mathbf{w}) = \prod_{i=1}^{n} p(c_i|\mathbf{O},\mathbf{w}) \tag{6}$$

where $\mathbf{w}$ are the parameters of the distribution. This assumes that the classification labels are conditionally independent given the data. However, correlations between the labels are indirectly introduced via the Gaussian Markov random field part of the model. In particular, as is apparent from Figure 3, in the full model, the labels are no longer conditionally independent given $\mathbf{O}$ because $\mathbf{G}$ is not observed. In an earlier version of the model, we imposed direct correlation between the c_i's as well as the g_i's, but this requires the use of slow methods such as graphcuts at inference time (see [5] for an example of this approach), and did not yield better results than the simpler model we present here.

Each $p(c_i|\mathbf{O},\mathbf{w})$ term is a probabilistic binary classifier with parameters $\mathbf{w}$. The input of each classifier is a feature vector $\boldsymbol{\phi}_i$ derived from the neighborhood around point i. Section III-D will describe the feature vector used in this article in more detail. The binary classifier used in this article is an ensemble of decision trees, i.e.,

$$f(\boldsymbol{\phi}_i,\mathbf{w}) = \sum_{j=1}^{M} \varpi_j f_j(\boldsymbol{\phi}_i, \boldsymbol{\varrho}_j) \tag{7}$$

where $\mathbf{w} = (\boldsymbol{\varpi}, \boldsymbol{\varrho})$, ϖ_j is the weight associated with decision tree j, and $\boldsymbol{\varrho}_j$ are its parameters. Given labeled training data, the parameters of this ensemble classifier can be learned using the GentleBoost algorithm [28]. We could of course use other classifiers, such as a support vector machine (SVM) [24], although the training and classification time might increase. Boosted decision trees are widely used in the machine learning and computer vision communities in view of their simplicity, speed, and accuracy.

Since we need to combine the output of the classifiers with other sources of information, we convert the output of the binary classifier to a probability using the technique of [29]. Specifically, we set

$$p(c_i = 1|\mathbf{O},\mathbf{w}) = \sigma(af(\boldsymbol{\phi}_i,\mathbf{w}) + b) \tag{8}$$

where $f(\boldsymbol{\phi}_i,\mathbf{w})$ is the output of the boosted classifier, and $\sigma(\eta) = \frac{1}{1+e^{-\eta}}$ is the logistic or sigmoid function. We set the parameters a and b by maximum likelihood on a validation set, in order to get reasonably well calibrated probabilities.

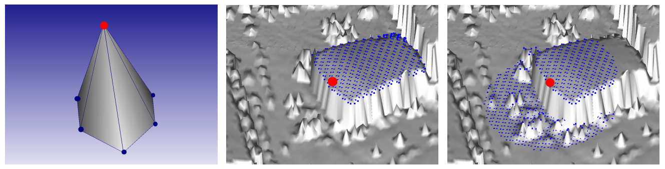

(a) vertex-based (b) segment-based (c) disc-based

Fig. 4. Features with different neighborhoods: (a) vertex-based features. (b) segment-based features. (c) disc-based features. In all figures, the red point represents the current vertex v_i and the blue points represent its supporting neighborhood.

TABLE I
SUMMARY OF THE FEATURES WE USED TO BUILD OUR CLASSIFIER.

No.	Type	Name	Description
1	Vertex	AT_{avg}	average arc tangent to neighbors
2	Vertex	AT_{min}	minimum arc tangent to neighbors
3	Vertex	AT_{max}	maximum arc tangent to neighbors
4	Vertex	VZD	z difference to global z mean
5	Vertex	$OUTL$	outlier confidence
6	Segment	SZV	segment z variance
7	Segment	RH	segment relative height
8	Segment	RH_{high}	segment average z difference to higher region neighbors
9	Segment	RH_{low}	segment average z difference to lower region neighbors
10	Segment	NRH_{high}	percentage of higher neighbors
11	Segment	NRH_{low}	percentage of lower neighbors
12	Segment	SNV	number of vertices
13	Disc	DZD	z difference to lowest neighbors

D. Features

We use a variety of features, illustrated in Figure 4 and summarized in Table I. We give a more detailed description below. Note that it is very common in applications of CRFs to use a large number of manually specified features — see e.g., [2]. The boosted decision tree automatically performs feature selection and feature construction (conjunctions, etc.), and sets the weights appropriately.

1) Point-based features: Some points can be classified by looking at very local information (see Fig. 4(a)). For example, if a point is much higher than its neighbors, it is probably a tree-top (and hence not a ground point). To formalize this, let $s(i,j) = \frac{z_i - z_j}{\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}}$ be the slope between vertices i and j, and let $a(i,j) = \arctan(s(i,j))$ be the angle of this slope. We define the following 3 features:

$$AT_{avg}(i) = \sum_{j\in\mathcal{N}_i} a(i,j) \tag{9}$$

$$AT_{min}(i) = \min_{j\in\mathcal{N}_i} a(i,j) \tag{10}$$

$$AT_{max}(i) = \max_{j\in\mathcal{N}_i} a(i,j) \tag{11}$$

2) Segmentation-based features: Clearly we cannot distinguish roof-tops from the ground by using point-based features, since locally roofs look flat just like the ground. So we need a more global feature. We therefore segment the point cloud into regions of similar height and derive a feature vector per segment.

Recently, researchers have introduced several graph-based segmentation algorithms. We choose to use the method of [30]

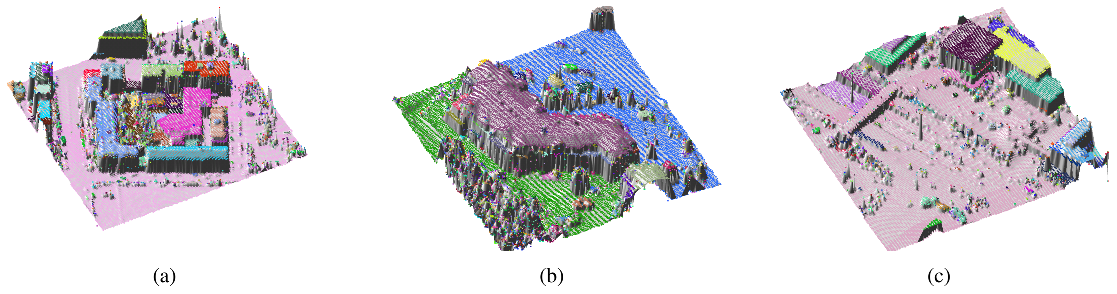


Fig. 5. Example outputs from the segmentation algorithm. (a) complex buildings; (b) a large building; (c) buildings and a bridge.

because it is very fast, and it does not require the number of segments to be known a priori. This algorithm takes a graph as input (derived from the Delaunay triangulation), and a set of edge weights, which we set to $s(i, j)$. Some examples of the output of the segmentation algorithm are shown in Fig. 5; we see it does a very reasonable job.

After obtaining the segmentation, features of each segment are computed. An obvious feature is height of a segment S relative to its neighboring segments. This can be defined as:

$$RH(S) = \sum_{i \in \mathcal{B}(S)} h(i, S) \tag{12}$$

where $\mathcal{B}(S)$ are the boundary points of segment S, and

$$h(i, S) = z_i - \min_{j \in \mathcal{N}_i \cap j \notin S} z_j \tag{13}$$

We also divide $\mathcal{B}(S)$ into two sets: one set contains points lower than their neighboring segments, and another one contains points that are higher. We then compute the height of a segment relative to the lower and higher neighbors, and the relative sizes. Such features turn out to be useful for structures such as buildings with terraces (see Fig. 5(c)).

3) Disc-based features: We also use features computed by a disc-shaped neighborhood with a specified radius (Fig. 4 (c)). Currently, the only disc-based feature we use is the elevation difference between the current vertex v_i and the lowest vertex within a range. This is inspired by the morphological operator [16], but the window size is fixed. This feature is useful for handling trees, buildings, bridges, etc.

4) Discussion: Not suprisingly, no single feature can handle all cases. For example, in Fig. 5(c), the segmentation algorithm mistakenly puts the bridge and ground into the same segment (a very reasonable error!), and thus segment-based features may fail. However, by combining segment-based features with disc-based features, this problem can be solved. On the other hand, using disc-based features alone is not sufficient, especially in the middle of a large roof-top segment where points are locally similar to ground. This is the reason why we propose to learn a classifier that combines all three feature sets together.

IV. Inference

Once we have learned the parameters from labeled training data, we can use the model to extract DTMs from new DSMs. This problem can be formulated as finding the Maximum a Posterior (MAP) solution of $\mathbf{g}$, i.e.,

$$\hat{\mathbf{g}} = \arg\max_{\mathbf{g}} p(\mathbf{g}|\mathbf{O}, \boldsymbol{\theta}) = \arg\max_{\mathbf{g}} \sum_{\mathbf{c}} p(\mathbf{g}, \mathbf{c}|\mathbf{O}, \boldsymbol{\theta}) \tag{14}$$

where $\hat{\mathbf{g}}$ is the MAP solution of $\mathbf{g}$. Unfortunately, computing the exact posterior mode is NP-hard [31], since we have to sum over all possible $\mathbf{c}$, and there are 2^n possible combinations. However, we can compute an approximate locally optimal solution using a variant of the Expectation-Maximization (EM) algorithm [4]. The EM algorithm is an iterative procedure of two steps: the E-step estimates the probability distribution over point class labels based on the old surface (DTM) estimate, while the M-step uses the "soft labels" to estimate a new DTM (surface). This use of EM is somewhat unusual, because we are treating the estimated DTM like "parameters", and point classification like "hidden variables", but the method is still valid (see e.g., [32]). We give some of the details below. (Interested readers can refer to [33] for a detailed description of the EM algorithm.)

A. M-step: extract bare-earth surface

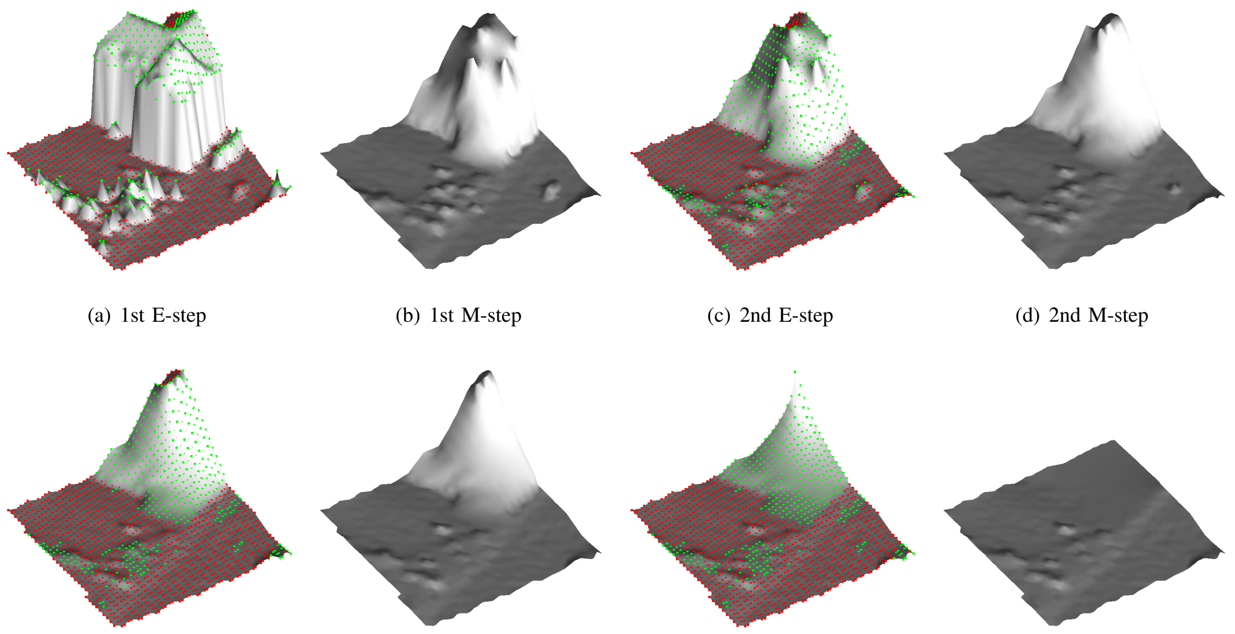
Define the expected negative log posterior at iteration t, where expectations are taken with respect to $\mathbf{g}^{t-1}$, as follows:

$$\begin{aligned} Q(\mathbf{g}, \mathbf{g}^{t-1}) &\propto -\mathbb{E}_{\mathbf{c}}\left[\log p(\mathbf{g}|\mathbf{c}, \mathbf{O}, \boldsymbol{\theta})\right] \\ &= \mathbb{E}_{\mathbf{c}}\left[\lambda_0(\mathbf{g}-\mathbf{z})^T(\mathbf{I}-\mathbf{C})(\mathbf{g}-\mathbf{z}) + \lambda_1(\mathbf{L}\mathbf{g})^T\mathbf{C}(\mathbf{L}\mathbf{g})\right] + K \\ &= \lambda_0(\mathbf{g}-\mathbf{z})^T(\mathbf{I}-\mathbf{W})(\mathbf{g}-\mathbf{z}) + \lambda_1(\mathbf{L}\mathbf{g})^T\mathbf{W}(\mathbf{L}\mathbf{g}) + K \end{aligned} \tag{15}$$

where $\mathbf{W} = \text{diag}(w_i)$, $w_i = p(c_i = 1|\mathbf{g}^{t-1}, \mathbf{O}, \boldsymbol{\theta})$, and $\mathbf{z}$ are the vertical components of the observations $\mathbf{O}$, and K is a constant. Taking derivatives of Eq. (15) with respect to $\mathbf{g}$, and setting to zero gives

$$\left(\lambda_0(\mathbf{I}-\mathbf{W}) + \lambda_1\mathbf{L}^T\mathbf{W}\mathbf{L}\right)\mathbf{g} = \lambda_0(\mathbf{I}-\mathbf{W})\mathbf{z} \tag{16}$$

(where we used the fact that $\nabla_{\mathbf{x}}\ (\mathbf{x}^T\mathbf{A}\mathbf{x}) = (\mathbf{A}+\mathbf{A}^T)\mathbf{x}$.) Although we can easily solve this for $\mathbf{g}$ by inverting the matrix on the left hand side, this takes $O(n^3)$ time, where n is the number of points in the dataset. We now show how to exploit the sparsity in the $\mathbf{W}$ and $\mathbf{L}$ matrices to compute exactly the same solution in $O(n^2)$ time.



(a) 1st E-step (b) 1st M-step (c) 2nd E-step (d) 2nd M-step

(e) 4th E-step (f) 4th M-step (g) 8th E-step (h) 8th M-step

Fig. 6. Illustration of the EM algorithm. The E-step determines the point classification (red = ground, green = non-ground), while the M-step estimates the bare-earth surface of the DTM using the point classification.

First we rewrite the above equation in the following equivalent form:

$$\begin{bmatrix}\lambda_0^{\frac{1}{2}}(\mathbf{I}-\mathbf{W})^{\frac{1}{2}}\\ \lambda_1^{\frac{1}{2}}\mathbf{W}^{\frac{1}{2}}\mathbf{L}\end{bmatrix}^T \begin{bmatrix}\lambda_0^{\frac{1}{2}}(\mathbf{I}-\mathbf{W})^{\frac{1}{2}}\\ \lambda_1^{\frac{1}{2}}\mathbf{W}^{\frac{1}{2}}\mathbf{L}\end{bmatrix}\mathbf{g} = \begin{bmatrix}\lambda_0^{\frac{1}{2}}(\mathbf{I}-\mathbf{W})^{\frac{1}{2}}\\ \lambda_1^{\frac{1}{2}}\mathbf{W}^{\frac{1}{2}}\mathbf{L}\end{bmatrix}^T \begin{bmatrix}\lambda_0^{\frac{1}{2}}(\mathbf{I}-\mathbf{W})^{\frac{1}{2}}\mathbf{z}\\ \mathbf{0}\end{bmatrix} \tag{17}$$

This has the form $\mathbf{A}^T\mathbf{A}\mathbf{g} = \mathbf{A}^T\mathbf{b}$. Solving this for $\mathbf{g}$ corresponds to finding a least squares solution to the linear system $\mathbf{A}\mathbf{g} = \mathbf{b}$, given by

$$\begin{bmatrix}\lambda_0^{\frac{1}{2}}(\mathbf{I}-\mathbf{W})^{\frac{1}{2}}\\ \lambda_1^{\frac{1}{2}}\mathbf{W}^{\frac{1}{2}}\mathbf{L}\end{bmatrix}\mathbf{g} = \begin{bmatrix}\lambda_0^{\frac{1}{2}}(\mathbf{I}-\mathbf{W})^{\frac{1}{2}}\mathbf{z}\\ \mathbf{0}\end{bmatrix} \tag{18}$$

Since the matrix on the left-hand side is very sparse, we can solve this system very efficiently. (This technique is known in the computer graphics community as Laplacian Mesh Optimization [25], [26].)

B. E-step: re-classify points

The E-step computes the probability of point classification needed in the M-step. By looking at the graphical model in Figure 3, we see that the c_i are conditionally independent given $\mathbf{g}$ and $\mathbf{O}$, so we can estimate each of them separately. Hence we need $w_i = \mathbb{E}[c_i|\mathbf{g}^{t-1}, \mathbf{O}, \boldsymbol{\theta}]$. Using Bayes rule, we have:

$$p(c_i = 1|\mathbf{g}, \mathbf{O}, \boldsymbol{\theta}) = \frac{p(c_i{=}1|\mathbf{O},\theta)\, p(\mathbf{g}|c_i{=}1,\mathbf{O},\theta)}{p(c_i{=}1|\mathbf{O},\theta)\, p(\mathbf{g}|c_i{=}1,\mathbf{O},\theta) + p(c_i{=}0|\mathbf{O},\theta)\, p(\mathbf{g}|c_i{=}0,\mathbf{O},\theta)} \tag{19}$$

Now

$$p(\mathbf{g}|c_i, \mathbf{O}, \boldsymbol{\theta}) = p(g_i|c_i, \mathbf{g}_{-i}, \mathbf{O}, \boldsymbol{\theta})\, p(\mathbf{g}_{-i}|\mathbf{O}, \boldsymbol{\theta}) \tag{20}$$

where $\mathbf{g}_{-i}$ refers to all the $\mathbf{g}$ variables except g_i. Note that the second term in the above expression is independent of c_i and hence will cancel in Equation 19. Also,

$$p(g_i|c_i = 1, \mathbf{g}_{-i}, \mathbf{O}, \boldsymbol{\theta}) \propto \exp\Big(-\tfrac{\lambda_1}{2}(g_i - \tfrac{1}{n_i}\sum_{j\in\mathcal{N}_i} g_j)^2\Big)$$
$$p(g_i|c_i = 0, \mathbf{g}_{-i}, \mathbf{O}, \boldsymbol{\theta}) \propto \exp\Big(-\tfrac{\lambda_0}{2}(g_i - z_i)^2\Big) \tag{21}$$

since if $c_i = 1$, g_i depends on its neighbors, whereas if $c_i = 0$, g_i depends on its observed elevation, z_i. Hence we can easily classify point c_i, by using the discriminative classifier to provide the prior $p(c_i|\mathbf{O}, \boldsymbol{\theta})$, and using the estimated heights of neighboring points, and the observed measurement of the current point z_i, in the likelihood term:

$$p(c_i = 1|\mathbf{g}, \mathbf{O}, \boldsymbol{\theta}) = \frac{P_1 \exp\Big(-\frac{\lambda_1}{2}(g_i - \frac{1}{n_i}\sum_{j\in\mathcal{N}_i} g_j)^2\Big)}{P_1 \exp\Big(-\frac{\lambda_1}{2}(g_i - \frac{1}{n_i}\sum_{j\in\mathcal{N}_i} g_j)^2\Big) + P_0 \exp\Big(-\frac{\lambda_0}{2}(g_i - z_i)^2\Big)} \tag{22}$$

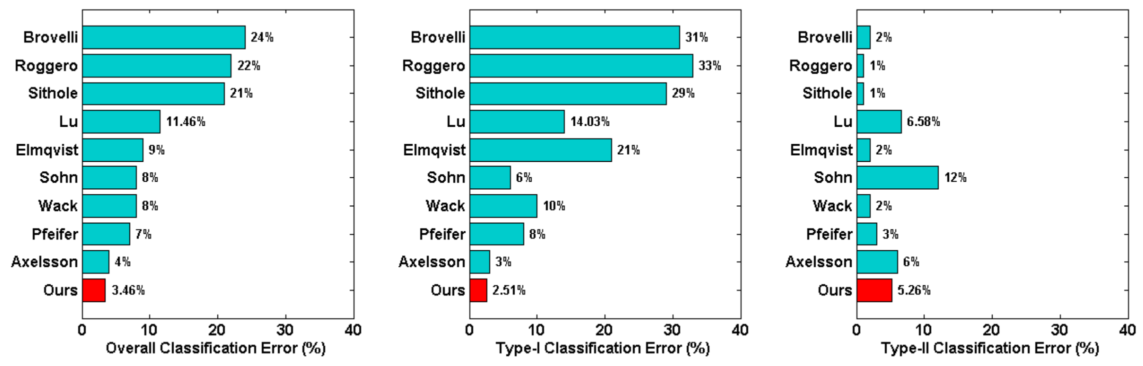
where $P_0 = p(c_i = 0|\mathbf{O}, \boldsymbol{\theta})$ and $P_1 = p(c_i = 1|\mathbf{O}, \boldsymbol{\theta})$.

C. First E-step: initial classification of points

To get the algorithm started, we need an initial estimate of the labels. This is very important so we don't get stuck in a poor local optimum. We can get a good initial guess by using our trained binary classifiers to compute $p(\mathbf{c}|\mathbf{O}, \mathbf{w})$, ignoring the $\mathbf{g}$ field.

D. Example

Fig. 6 illustrates an example of the EM algorithm in action. We can observe that, in the first iteration, the initial classification was inaccurate, since the roof-top was considered part



(a) (b) (c)

Fig. 7. Classification errors in estimating labels $\mathbf{c}$ on the Sithole dataset, compared with other methods evaluated in [1]. (a) Overall errors (Err), (b) Type-I errors (Err_1), (c) Type-II errors (Err_2).

of the ground surface. However, after optimizing the ground surface and re-classifying points, more and more points on the roof-top were correctly classified as non-ground. After eight iterations, the entire roof-top was classified as non-ground and then its underlying ground surface was correctly estimated. (In our experiments, the EM algorithm typically converges in 10-20 iterations.)

V. Experimental Results

In order to evaluate the performance of the proposed algorithm, which we implemented using the Graphite environment [34], we test our system on two datasets: the Sithole *et al.* [1] dataset and the Terrapoint dataset [35]. We evaluate both the quantitative and qualitative performance of the proposed algorithm.

A. Datasets

The Sithole dataset [1] consists of 15 sites with various terrain characteristics (buildings, steep slopes, bridges, terrain discontinuities, ramps, vegetation on slopes and many others), and contains 384,325 points in total. Sithole *et al.* manually classified each data point and thus the ground-truth labellings are very accurate.

The Terrapoint data [35] consists of three huge sites with 3 million data points. The first and third sites contain vegetation and roads, while the second site is composed of forests, buildings, and cars. Unlike the Sithole dataset that carefully classified every point into ground and non-ground, the Terrapoint dataset chose only a small set of points and labeled them as ground. Other points, including many that would be visually classified as ground under the standards of the Sithole dataset, are considered as non-ground, a point which we discuss further below.

B. Quantitative Evaluation

We evaluate the quantitative performance of our system in terms of classification performance, and in terms of the distance between the estimated and ground-truth bare-earth surface. The classification of points is determined by $p(\mathbf{c}|\hat{\mathbf{g}}, \mathbf{O}, \boldsymbol{\theta})$, where $\hat{\mathbf{g}}$ is the optimal one computed in the last iteration of the EM algorithm. In particular, we say point i is non-ground if $p(c_i = 1|\hat{\mathbf{g}}, \mathbf{O}, \boldsymbol{\theta}) \geq 0.5$; otherwise, we say point i belongs to the ground surface.

Let E_1 be the number of ground points that our algorithm mistakenly classifies as non-ground, and E_2 be the number of non-ground points that our algorithm mistakenly classifies as ground. We define the type I, type II, and overall error rates as follows:

$$Err_1 = \frac{E_1}{N_1} \quad Err_2 = \frac{E_2}{N_2} \quad Err = \frac{E_1 + E_2}{N_1 + N_2} \tag{23}$$

To measure the distance between the estimated and true bare-earth surface, let $dist(p, S)$ be the distance between a point p and a surface S, i.e.,

$$dist(p, S) = \min_{p' \in S} \|p - p'\| \tag{24}$$

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

$$dist_{avg}(S_1, S_2) = \frac{1}{|S_1|} \int_{p \in S_1} dist(p, S_2) dp \tag{25}$$

where $1/|S_1|$ is the area of S_1. We use a standard package named Metro [36] to compute the average distance between two 3D meshes.

Since we do not have an independent test set, we run 10-fold cross-validation on the Sithole dataset. More specifically, we partitioned the Sithole dataset into ten disjoint parts; we train the model using nine of them, and evaluate the performance using the remaining one, and then repeat this 10 times, recording the average test performance.

In Fig. 7, we compare the classification accuracy with the state-of-the-art filtering algorithms evaluated by Sithole *et al.* [1]. Our overall error rate, 3.46%, is slightly better than the best state-of-the-art hand-tuned algorithms (4%). Our type I error rate (2.51%) is also better than all the other algorithms (best rival is 3%), although our type II error rate is slightly higher (ours is 5.26%, best rival is 1%).

By changing the classification threshold, or equivalently the b offset parameter in the logistic function (see Eq. (8)), we can tradeoff type I and type II errors as shown in Table II. When b is larger, we are more likely to classify points as ground,

TABLE II
QUANTITATIVE EVALUATION OF THE SITHOLE DATASET, BY CHANGING THE CLASSIFICATION THRESHOLD (MODEL PARAMETER b).

b	Overall Err.	Type-I Err.	Type-II Err.	Err. Distance
0.03	8.00%	**0.15%**	22.91%	28.60cm
0.02	5.60%	0.43%	15.40%	20.08cm
0.01	3.78%	1.09%	8.89%	11.70cm
0.00	**3.46%**	2.51%	5.26%	**9.79cm**
−0.01	4.70%	5.83%	2.56%	15.79cm
−0.02	7.76%	11.14%	1.33%	25.60cm
−0.03	12.48%	18.73%	**0.62%**	38.84cm

yielding low Type-I error but high Type-II error. When b is smaller, we are more likely to classify points as non-ground, and thus increase the Type-I error but decrease the Type-II error. Many filtering algorithms in [1] focus on minimizing the Type-II error. As we can see from Table II, we can achieve a type-II error which is as low as 0.62%, beating all the other methods reported in [1]. However, we also notice that minimizing the Type-II error does not yield a smaller error distance. In our experiments, the smaller error distance is achieved when the overall classification error is minimized, which corresponds to setting $b = 0$.

Unfortunately, Sithole *et al.* did not provide the error distance between the estimated and true ground surface, and therefore we cannot compare our proposed algorithm with other methods using this measure. In order to get some kind of quantitative measure of the relative accuracy of our estimated surfaces, we decided to compare our method to our previous system described in [5]. Our old system was tuned by hand on the Terrapoint data. By contrast, our current system was trained automatically on the Sithole data.[3]

Despite this handicap, Fig. 8 shows that our new method gives lower classification and distance errors compared to our old method. Thus we can get better performance with less effort by using our new method.

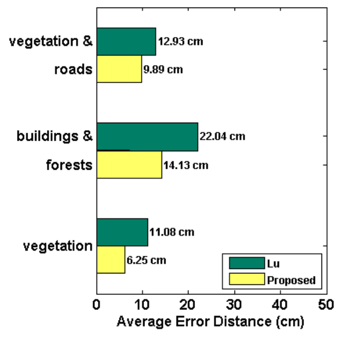
C. Qualitative Evaluation

We can gain a better appreciation of the performance of our method by looking at examples of it in action. Fig. 9 (a) shows a forested region on a slope, along with some houses. Notice that there are high trees as well as some bushes. The proposed algorithm is able to successfully identify most of the non-ground points and estimate their underlying bare-earth surface.

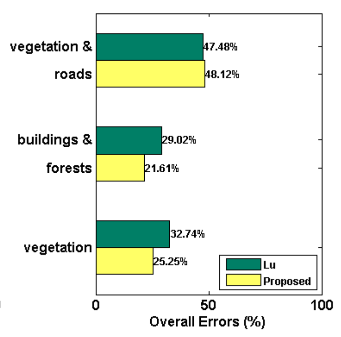
Fig. 9 (b) is an example of terrain discontinuity, with some vegetation and houses on one side. We can observe that the cliff has similar characteristic to the walls of buildings. However, by using the segmentation-based features, the proposed algorithm is able to classify the cliff as ground.

Fig. 9 (c) shows some houses with flat, gable, and irregular roofs along with some big trees (the cone-shaped objects). The proposed algorithm is able to extract accurate DTMs because our classifier combines different kinds of features.

[3]Since the Terrapoint data contained many points that were classified as non-ground, even though visually it was obvious that they were ground, we modified the b parameter in order to match the conservative labeling standards of Terrapoint. However, all the other parameters of the model (which were trained on the Sithole dataset) were left unchanged.



(a)



(b)

Fig. 8. Quantitative evaluation on the Terrapoint dataset [35]. In both figures, the green bars (top) represent the results of [5] (denoted Lu), and the yellow bars (bottom) represent the results of the proposed algorithm. (a) Average error distance. (b) Overall classification error rate.

Fig. 9 (d) is an example of complex buildings with flat and gable roofs. We can also observe some bushes in the middle of road. The proposed algorithm still works quite effectively.

In Fig. 9 (e), we can see a large building with a flat roof. If we only use vertex-based and disc-based features, then many points on the roof will be mis-classified as ground points because they are similar to ground on a local scale. However, by combining the segmentation-based features with vertex-based and disc-based features, points on the roof are successfully classified as ground, and their bare-earth elevation are accurately estimated.

Fig. 9 (f) shows a bridge. This example emphasizes the importance of disc-based features because the bridge cannot be successfully removed by using only segmentation-based and vertex-based features.

Fig. 10 shows two examples, one forested region and one urban region, from a bird's eye view. In Fig. 10 (a), we can see dense vegetation with some small houses. In Fig. 10 (b), we can see buildings with flat and gable roofs. In both cases, the proposed algorithm is able to remove most of the non-ground points and extract DTMs quite effectively.

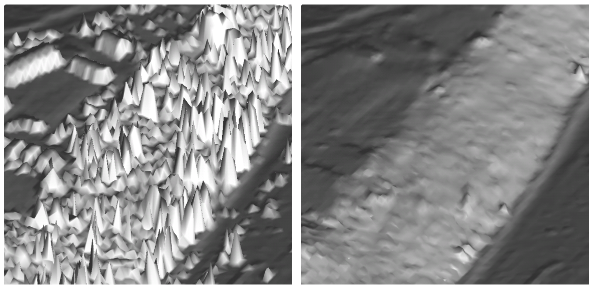
VI. CONCLUSION

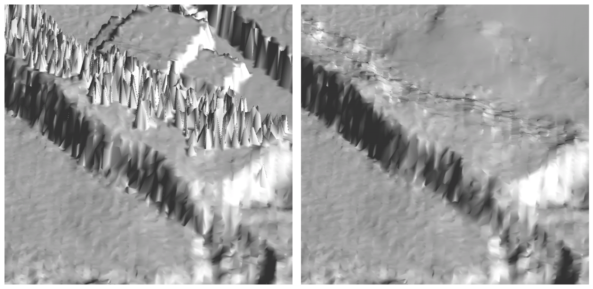
We presented a hybrid Conditional Random Field method for extracting bare-earth surfaces from airborne LiDAR data. Extensive testing against hand-classified data and commercial data shows that our results improve upon manually tuned classifiers, and produce surfaces that are better than previous bare-earth extraction systems.

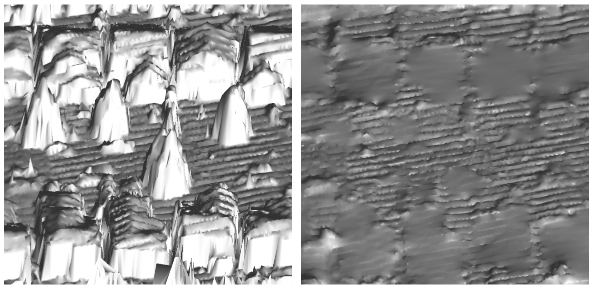
Currently our system does not include models of sensing errors due to scattering, and cannot handle multiple returns. In addition, we do not use any knowledge about regularities of man-made or other structures. We leave these extensions for future work.

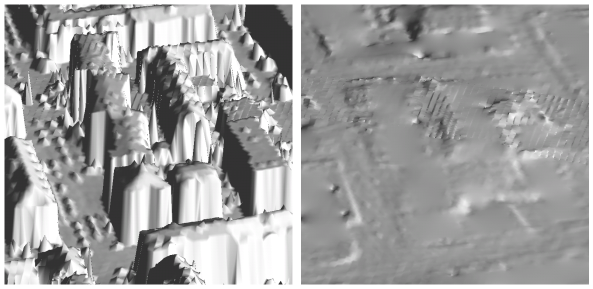
ACKNOWLEDGMENT

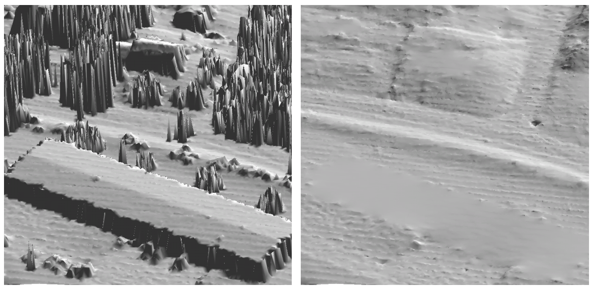
This work has been supported by grants from the GEOIDE Network of Centres of Excellence, Terrapoint Canada (A


(a) Trees on a slope


(b) Hill, trees, and house


(c) Houses with gable and irregular roofs


(d) Houses and brush on road


(e) Large building with flat roof

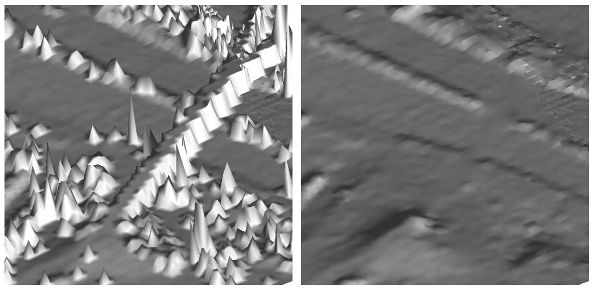

(f) Bridge

Fig. 9. Extracted DTMs of different kinds of terrain from a 45° view. For each pair of images, the left one is the original DSM, and the right one is the extracted DTM. (b), (c), (e) are from Terrapoint, (a), (d), (f) are from Sithole.

division of Ambercore), the National Science and Engineering Research Council of Canada, and the Canadian Institute for Advanced Research. We thank the ALICE team at Inria Nancy Grand-Est for providing us the Graphite software environment.

References

[1] 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*, vol. 59, pp. 85–101, 2004.

[2] 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*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 282–289.

[3] 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*, vol. 2, 2003, pp. 1150–1157.

[4] R. M. Neal and G. Hinton, "A new view of the EM algorithm that justifies incremental, sparse, and other variants," in *Learning in Graphical Models*, M. I. Jordan, Ed. Kluwer Academic Publishers, 1998, pp. 355–368.

[5] W.-L. Lu, J. J. Little, A. Sheffer, and H. Fu, "Deforestation: Extracting 3D Bare-Earth Surface from Airborne LiDAR Data," in *The Fifth Canadian Conference on Computer and Robot Vision*, Windsor, Ontario, May 2008, pp. 203–210.

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

[7] 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*, 2001, pp. 203–210.

[8] M. Roggero, "Airborne laser scanning: clustering in raw data," in *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXIV (Pt. 3/W4)*, 2001, pp. 227–232.

[9] R. Wack and A. Wimmer, "Digital terrain models from airborne laser scanner data – a grid based approach," in *International Archives of the Photogrammetry, Remote Sensing and Spatial Informational Sciences XXXIV (Pt. 3B)*, 2002, pp. 293–296.

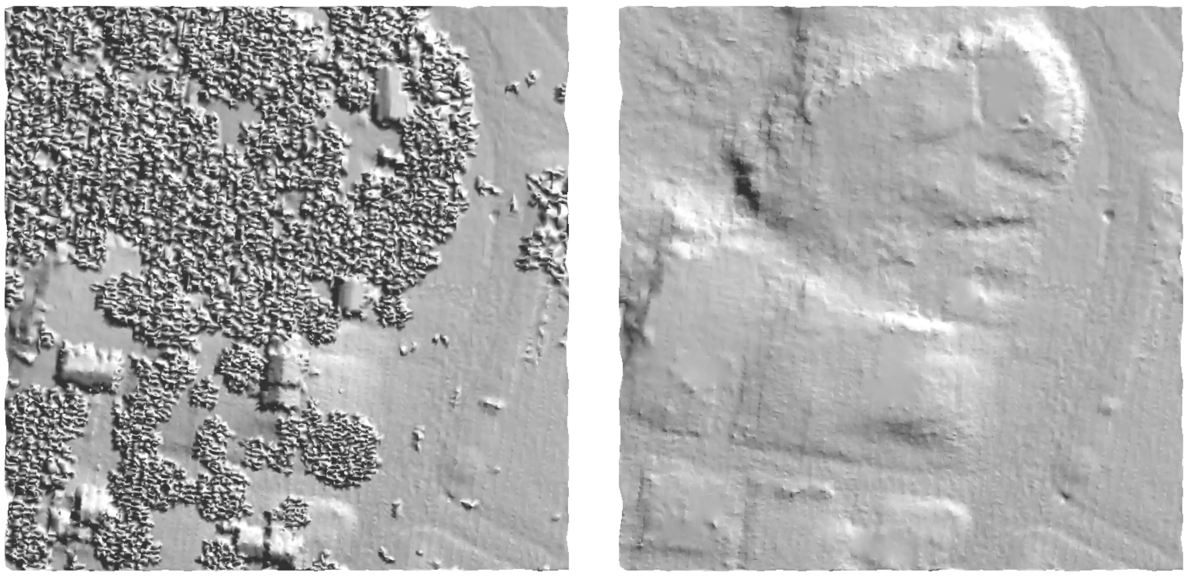
[10] R. Haugerud and D. Harding, "Some algorithms for virtual deforestation (VDF) of lidar topographic survey data," in *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXIV (Pt. 3/W4)*, 2001, pp. 211–217.

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

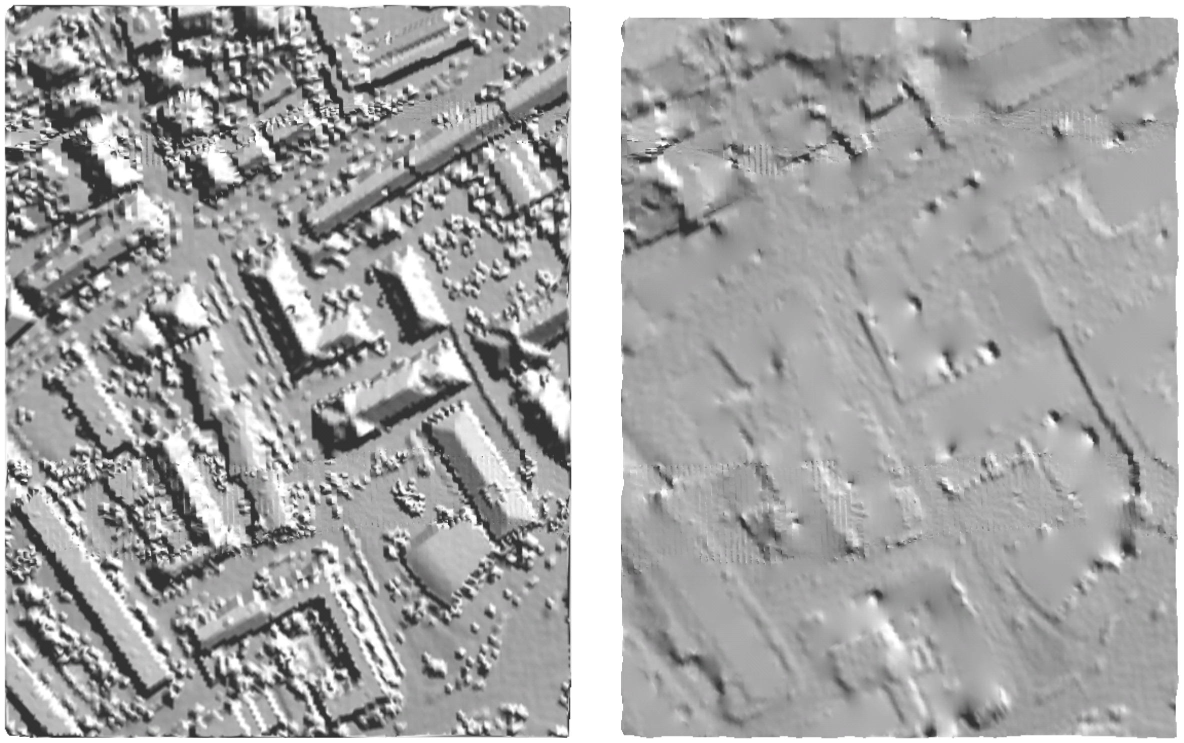
[12] C. Briese, N. Pfeifer, and P. Dorninger, "Applications of the robust interpolation for DTM determination," in *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXIV, 3A*, 2002, pp. 55–61.

[13] G. Sohn and I. Dowman, "A Model-based Approach for Reconstructing Terrain Surface from Airborne LiDAR Data," *The Photogrammetric Record*, vol. 23, no. 122, pp. 170–193, 2008.

[14] M. Brovelli, M. Cannata, and U. Longoni, "Managing and processing lidar data within grass," in *Proceedings of GRASS Users Conference*,



(a) Dense vegetation and some houses



(b) Complex buildings

Fig. 10. Extracted DTMs of different kinds of terrain from a bird's eye view. For each pair of images, the left one is the original DSM, and the right one is the extracted DTM. (a) is from Terrapoint, (b) is from Sithole.

Trento, Italy, 2002.

[15] F. Dell'Acqua and P. Gamba, "Preparing an Urban Test Site for SRTM Data Validation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 10, pp. 2248–2256, 2002.

[16] K. Zhang, S.-C. Chen, D. Whitman, M.-L. Shyu, J. Yan, and C. Zhang, "A Progressive Morphological Filter for Removing Nonground Measurements From Airborne LIDAR Data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 4, pp. 872–882, 2003.

[17] K. Zhang, J. Yan, and S.-C. Chen, "Automatic Construction of Building Footprints From Airborne LIDAR Data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 9, pp. 2523–2533, 2006.

[18] J. S. Evans and A. T. Hudak, "A Multiscale Curvature Algorithm for Classifying Discrete Return LiDAR in Forested Environments," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 4, pp. 1029–1038, 2007.

[19] R. A. McLaughlin, "Extracting Transmission Lines From Airborne LIDAR Data," *IEEE Geoscience and Remote Sensing Letters*, vol. 3, no. 2, pp. 222–226, 2006.

[20] F. Han, Z. Tu, and S.-C. Zhu, "Range Image Segmentation by an Effective Jump-Diffusion Method," *IEEE Trans. on Pattern Analysis and*

Machine Intelligence, vol. 26, no. 9, pp. 1138–1153, 2004.
[21] C. Baillard and H. Maitre, "3-D Reconstruction of Urban Scenes from Aerial Stereo Imagery: A Focusing Strategy," *Computer Vision and Image Understanding*, vol. 76, no. 3, pp. 244–258, 1999.
[22] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng, "Discriminative Learning of Markov Random Fields for Segmentation of 3D Scan Data," in *CVPR*, 2005.
[23] J. Secord and A. Zakhor, "Tree Detection in Urban Regions Using Aerial Lidar and Image Data," *IEEE Geoscience and Remove Sensing Letters*, vol. 4, no. 2, pp. 196–200, 2007.
[24] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, September 1995.
[25] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "Laplacian Mesh Optimization," in *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques*, 2006, pp. 381–389.
[26] L. Liu, C.-L. Tai, Z. Ji, and G. Wang, "Non-iterative approach for global mesh optimization," *Computer-Aided Design*, vol. 39, pp. 772–782, 2007.
[27] H. Rue and L. Held, *Gaussian Markov Random Fields: Theory and Applications*, ser. Monographs on Statistics and Applied Probability. London: Chapman & Hall, 2005, vol. 104.
[28] J. Friedman, T. Hastie, and R. Tibshirani, "Additive Logistic Regression: A Statistical View of Boosting," *The Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.
[29] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers*, A. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, Eds. MIT Press, 1999.
[30] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient Graph-Based Image Segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.
[31] U. Lerner and R. Parr, "Inference in hybrid networks: Theoretical limits and practical algorithms," in *UAI*, 2001.
[32] M. Tanner, *Tools for statistical inference*. Springer, 1996.
[33] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
[34] Graphite, "http://ralyx.inria.fr/2006/raweb/alice/uid28.html."
[35] Terrapoint, "http://www.terrapoint.com/."
[36] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: measuring error on simplified surfaces," in *Computer Graphics Forum*, vol. 17, no. 2, 1998, pp. 167–174.

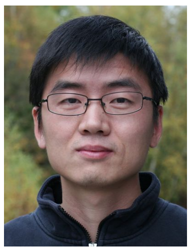
Wei-Lwun Lu is currently pursuing the Ph.D. degree at the University of British Columbia, Canada. He received M.Sc. degree in computer science from the University of British Columbia in 2007 and the B.S. degree in computer science from National Tsing Hua University, Taiwan, in 2002. His primary research interests fall in the field of machine learning and computer vision, especially in the problems of geospatial data processing and multi-target tracking.

Kevin P. Murphy has been an assistant professor at UBC (University of British Columbia in Vancouver, Canada) in the departments of computer science and statistics since September 2004. He holds a Canada research chair in machine learning/ computational statistics. Prior to coming to UBC, Kevin did a postdoc at MIT, his PhD at UC Berkeley, his MSc at U. Pennsylvania, and his BSc at U. Cambridge. Kevin develops and applies Bayesian inference techniques to problems in computational biology and computer vision. He is best known for his work in the area of Bayesian networks/ graphical models.

James J. Little received the A.B. from Harvard College in 1972 and M.Sc. and Ph.D. degrees in Computer Science from the University of British Columbia in 1980 and 1985.

From 1985 to 1988, he was a Research Scientist at the MIT Artificial Intelligence Laboratory. Currently, he is a Professor of Computer Science at the University of British Columbia. His research interests include computational vision, robotics, and spatio-temporal information systems. Particular interests are stereo, motion, tracking, mapping, and motion interpretation.

Alla Sheffer is an associate professor in the Computer Science department at the University of British Columbia. Dr. Sheffer investigates algorithmic aspects of digital geometry processing, focusing on several fundamental problems of mesh manipulation and editing. Her recent research addresses algorithms for mesh parameterization, processing of developable surfaces, mesh editing, reconstruction, and shape analysis. Her work on these topics had been published at top venues, including Siggraph, Eurographics, and the Symposium on Geometry Processing. She co-authored several mesh processing methods which are used in popular 3D modelers including Blender, Maya, and Catia. Alla Sheffer received her PhD from the Hebrew University of Jerusalem in 1999. Prior to moving to UBC in 2003, she was a postdoc at the University of Illinois at Urbana-Champaign and an assistant professor at Technion, Israel.



Hongbo Fu is currently an Assistant Professor in the School of Creative Media, City University of Hong Kong. Before joining CityU, he had postdoctoral research trainings at the Imager Lab, University of British Columbia, Canada and the Department of Computer Graphics, Max-Planck-Institut Informatik, Germany. He received the PhD degree in computer science from the Hong Kong University of Science and Technology in 2007 and the BS degree in information sciences from Peking University, China, in 2002. His primary research interests fall in the field of computer graphics with an emphasis on digital geometry processing.