ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume II-3, 2014
ISPRS Technical Commission III Symposium, 5 – 7 September 2014, Zurich, Switzerland

DETECTION OF FALLEN TREES IN ALS POINT CLOUDS BY LEARNING THE NORMALIZED CUT SIMILARITY FUNCTION FROM SIMULATED SAMPLES

P. Polewski[a, b, *], W. Yao[a], M. Heurich[c], P. Krzystek[a], U. Stilla[b]

[a] Dept. of Geoinformatics, Munich University of Applied Sciences, 80333 Munich, Germany - (polewski, yao, krzystek)@hm.edu
[b] Photogrammetry and Remote Sensing, Technische Universität München, 80333 Munich, Germany- stilla@tum.de
[c] Dept. for Research and Documentation, Bavarian Forest National Park, 94481 Grafenau, Germany - marco.heurich@npv-bw.bayern.de

KEY WORDS: Fallen tree detection, graph cuts, virtual sample generation

ABSTRACT:

Fallen trees participate in several important forest processes, which motivates the need for information about their spatial distribution in forest ecosystems. Several studies have shown that airborne LiDAR is a valuable tool for obtaining such information. In this paper, we propose an integrated method of detecting fallen trees from ALS point clouds based on merging small segments into entire fallen stems via the Normalized Cut algorithm. A new approach to specifying the segment similarity function for the clustering algorithm is introduced, where the attribute weights are learned from labeled data instead of being determined manually. We notice the relationship between Normalized Cut's similarity function and a class of regression models, which leads us to the idea of approximating the task of learning the similarity function with the simpler task of learning a classifier. Moreover, we set up a virtual fallen tree generation scheme to simulate complex forest scenarios with multiple overlapping fallen stems. The classifier trained on this simulated data yields a similarity function for Normalized Cut. Tests on two sample plots from the Bavarian Forest National Park with manually labeled reference data show that the trained function leads to high-quality segmentations. Our results indicate that the proposed data-driven approach can be a successful alternative to time consuming trial-and-error or grid search methods of finding good feature weights for graph cut algorithms. Also, the methodology can be generalized to other applications of graph cut clustering in remote sensing.

1. INTRODUCTION

Fallen trees are considered an important part of forest ecosystems. Studies have revealed that dead wood has a role in providing habitat for various plants and animals (Freedman et al., 1996) as well as in tree regeneration (Weaver et al., 2009). Also, dead wood contributes significantly to the total carbon stock in forests (Woodall et al., 2008). For these reasons, qualitative and quantitative information about the spatial distribution of dead wood in forests is important for any organizations interested in monitoring biodiversity, carbon sequestration and wildlife habitats.

Airborne LiDAR has been shown to be a valuable tool in assessing various forest parameters at different scales. There have been many contributions dealing with extracting individual trees from the LiDAR data. The segmentations of the point clouds were then used for tree species classification (Reitberger et al., 2008; Heurich, 2008) or detection of standing dead trees (Yao et al., 2012). In this work, we describe a method of detecting entire fallen stems from ALS point clouds.

The problem of detecting fallen trees from ALS data has received some attention from the remote sensing community. Among the first methodological studies was the work of Blanchard et al. (2011). This approach relies on rasterizing the point cloud with respect to various point statistics as well as generating a vector thematic layer for object based image analysis to detect downed stems. The authors observe that their method has some difficulty in more complex scenarios like proximity of the stems to ground vegetation as well as large clusters of logs. Also, the processing pipeline relies on multiple user-defined parameters, making it more cumbersome to apply for an entirely new plot. Muecke et al. (2013) also perform the classification on a vectorized layer derived from the binarized point cloud filtered based on distance to the DTM. Additionally, to remove ground vegetation and shrubs, a pulse echo width filter is applied. The authors show that it is possible to reliably detect stems which distinguish themselves well upon the DTM. However, they note that applying the pulse echo width filter together with the vectorized object shape features is not always enough to separate stems from densely intertwined ground vegetation and piles of twigs. This study does not address the more complex scenario of multiple overlapping stems. In a recent study, Lindberg et al. (2013) perform line template matching directly in the point cloud, followed by height homogeneity analysis on rasterized versions of the found lines. An attempt to model the local neighborhood of stem candidates is made using a height statistic on points lying within a fixed radius. Upon validation on a complex test site, they find that although detecting the lines directly in 3D is advantageous, the method sometimes fails in the presence of large clusters of overlapping stems, and can also report false positives in dense vegetation, ditches or road fragments. Also, a multitude of user-defined thresholds restricts its application for other areas.

We believe that due to the complexity and variability of the target objects' appearance in the point cloud, methods which try to learn the appearance from reference data based on highly expressive shape descriptors could help solve some of the problems with the aforementioned threshold-based approaches. We propose a two-tiered procedure for detecting fallen trees. In the first step, stem segments of equal length are detected, yielding a set of primitives. In the second step, these segments are clustered together to form entire fallen trees. This is the next step in our previous work (Polewski et al., 2014), extending it from segment level to object (fallen tree) level. The focus of this paper is to present the clustering step in detail.

Our method of choice for merging the segments is the Normalized Cut algorithm (Shi and Malik, 2000), a spectral clustering

*Corresponding author.

procedure which has found applications in many fields, including computer vision, speech processing, and also remote sensing. Like several other clustering algorithms, Normalized Cut uses a pairwise object similarity function as input. Most authors employ the similarity model originally proposed by Shi and Malik, which aggregates different features using a set of weights. However, the individual feature weights are usually very application-specific, and considerable effort is required to find near-optimal weights for a new problem. The search for the weights is not part of the clustering algorithm itself, and in practice is carried out based on intuition, trial and error, or via a coarse grid search on discretized weight values. All of these methods are either very computationally intensive, or do not offer significant advantages over random weight generation. This motivates research towards finding ways to automatically learn the similarity function from reference segmentations. In this paper, we propose to regard the Normalized Cut similarity function as a probabilistic binary classifier which, for any pair of objects, determines how likely it is that they belong to the same cluster. We investigate the hypothesis that a high-performing binary classifier leads to a high-quality similarity function for Normalized Cut. Such a relationship would be useful in that it would allow us to reduce the problem of a learning similarity function to the much simpler and well-studied task of learning a classifier from labeled data. We then propose a virtual sample generation scheme for synthesizing complex stem scenarios and use the generated exemplars as training data to learn the similarity function for merging segments into entire fallen trees.

We consider the main contributions of this paper to be (i) the processing pipeline for obtaining entire stems from a set of overlapping segments, (ii) the fallen stem simulation which leads to high-quality classifiers capable of recognizing segments belonging to the same stem, and (ii) the idea to approximate the problem of learning the similarity function for Normalized Cut with the problem of training a binary classifier. The rest of this paper is structured as follows: in Section 2. we explain the details of our approach including the mathematical background. Section 3. describes the study area, experiments and evaluation strategy. The results are presented and discussed in Section 4. Finally, the conclusions are stated in Section 5.

2. METHODOLOGY

2.1 Overview of strategy for detecting fallen trees

We develop an integrated, data driven method for detecting single tree stems from unstructured ALS point clouds given by the three-dimensional coordinates of their constituent points. The approach is suitable for both discrete and full waveform data. The output of our method is a list of subsets of the original points which correspond to individual detected fallen trees. We assume that reference data in the form of accurate fallen stem positions, lengths and diameters is available. The entire processing pipeline

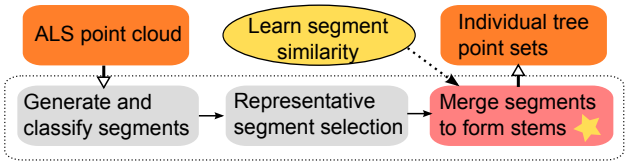


Figure 1: Overview of fallen tree strategy.

is depicted in Fig. 1. Conceptually, our approach tries to decompose the task of detecting entire stems into two simpler problems: first detecting stem segments in the point cloud, and then merging the segments to produce single fallen trees. This paper focuses on the latter task.

2.2 Generation and classification of segments

The goal of this stage is to detect linear structures in the ALS point cloud which are likely to correspond to segments of fallen stems. This is done in four steps:

DTM generation and filtering. We start by partitioning the area defined by the 2D bounding box of the input points into square cells of width 10 cm. For every cell c_k, we define the measured height H_k as the minimum height of all points $p \in c_k$. For calculating the DTM we use an Active Shape Model formulation (Elmqvist, 2002). In this setting, the terrain model is an elevation function $z(p)$ defined over an evenly spaced grid. We can associate a surface energy with every $z(p)$:

$$E(z) = \sum_p E_{img}(p) + E_{int}(p) \quad (1)$$

In Eq. 1 the summing is over the discrete grid points. The term E_{ext} refers to the image energy and usually is defined as sum of the distances between z and the measured heights on the grid. The term E_{int} is related to the energy of the surface. It introduces smoothness/rigidity constraints into the model. We obtain the DTM by minimizing the energy functional (Eq.1) on the grid defined by the cells c_k with their associated measured heights H_k with respect to the model heights $z(p)$. The minimization is carried out using the iteratively reweighted least squares method. Since this is a highly non-convex problem, attaining a global minimum is not guaranteed. Therefore, we use multiple randomized re-runs and pick the best solution. Finally, we retain only points lying within an interval of 15 to 100 cm over the ground.

Point-level classification. We use the 3D shape descriptor Point Feature Histograms (PFH) (Rusu et al., 2008) to train a probabilistic binary classifier in order to distinguish between fallen stem points and other points. We can then for each input point p obtain an estimate of the probability $P(p \in C_{stem}|PFH(p))$ that p belongs to a fallen stem (C_{stem} refers to the class of fallen stem points in the binary classification setting). See Polewski et al. (2014) for details.

Candidate segment generation. In this step, feasible candidates for fallen stem segments are generated. We restrict our attention to high-probability stem points (above 0.5) only. We exhaustively examine all pairs of such points which are close enough to each other. Each point pair determines a segment in 3D. We consider each segment as the axis of a fixed-radius cylinder, and retain only segments whose corresponding cylinders contain a minimal number of points.

Segment classification. We classify the candidate segments generated in the previous step into the classes 'good' and 'bad'. The 'good' class corresponds to segments which are really parts of fallen stems, and the 'bad' class involves DTM artifacts, ground vegetation, and other irrelevant objects. We train a classifier on manually labeled reference data using two groups of features: (i) a coarse histogram of the pointwise stem probabilities of points belonging to the segment, and (ii) a cylindrical variation of the shape descriptor 3D Shape Context (Frome et al., 2004).

2.3 Selecting representative segments

Because we generate segment candidates in an exhaustive manner, it is expected that many segments classified as 'good' will strongly overlap or even be duplicates in terms of the points they cover. In practice, we found that in a dense enough plot, hundreds of thousands of segments can still remain after the classification

phase. Therefore, we need a way to select a small subset of segments which will be representative of the entire set in the sense that it covers the same set of input points that is covered by the entire set. More formally, define $S = \{s_1, s_2, \ldots, s_K\}$ - the set of all 'good' segments, $P(s_i)$ - the set of points contained by segment s_i, and $P = \bigcup_{s_i} P(s_i)$ - the set-theoretic union of all input points belonging to any segment in S. Each of the s_i may now be viewed as a subset of P. This gives rise to the formulation of the classic set cover problem:

$$\begin{aligned} &\text{minimize} \sum_{i=1}^{K} x_i \\ &\text{subject to} \forall_{p \in P} \sum_{i:p \in P(s_i)} x_i \geq 1, \forall_i x_i \in \{0,1\} \end{aligned} \tag{2}$$

Unfortunately, the optimization version of this combinatorial problem is NP-hard, which means that an efficient polynomial algorithm probably does not exist. We use the MetaRAPS heuristic algorithm (Lan et al., 2007). It proceeds in an iterative fashion by constructing random covers of P based on a randomized greedy decision rule (highest probability of picking the segment which covers the most previously uncovered points). The algorithm then tries to improve the solution s by randomly removing a subset R of segments from s, approximately solving the smaller set cover problem induced by R, and then reintegrating the two partial solutions.

2.4 Segment merging

As a final step in our detection strategy, we perform merging of the representative segments. For this purpose, we define a set of differential features on the space of segment pairs (Sec. 2.4.2). Once the features have been defined, we can calculate the segment similarity matrix $(D_{i,j}) = f_s(s_i, s_j)$ (Eq. 4) for all pairs of segments s_i, s_j. We enforce a spatial constraint by calculating the similarity between a pair of segments only if there exists a cylinder with length 10 m and radius 2.4 m which contains both their midpoints (otherwise the similarity is assumed to be 0). We then employ the Normalized Cut algorithm (2.4.1) to cluster the segments into individual stems. Once this is done, the original input points corresponding to each detected tree can be calculated as the union of point subsets belonging to its constituent segments.

2.4.1 Normalized Cut The Normalized Cut algorithm, introduced by Shi and Malik (2000), belongs to the class of spectral methods for data clustering. These methods exploit the eigenstructure of the pairwise object similarity matrix to partition the data points into several disjoint clusters with high intra-cluster similarity and low inter-cluster similarity. Normalized Cut can be used for clustering arbitrary objects, since it only interacts with these objects through a pairwise similarity function $f_s(o_1, o_2)$ which is nonnegative and attains a maximum for $o_1 = o_2$. Formally, given a set of N objects $O = \{o_i\}_{i=1..N}$ represented by their pairwise similarity matrix $S = \{f_s(o_i, o_j)\}_{i=1..N, j=1..N}$, the objective is to partition the objects into K disjoint clusters $A_1, A_2, \ldots, A_K$ such that the following criterion, the Normalized Cut, is minimized:

$$Ncut(A_1, A_2, \ldots, A_K) \equiv \sum_{i=1}^{K} \frac{cut(A_i, O - A_i)}{assoc(A_i, O)} \tag{3}$$

where $cut(A, B) \equiv \sum_{a \in A, b \in B} f_s(a, b)$ and $assoc(A, V) \equiv \sum_{a \in A, v \in V} f_s(a, v)$. A very desirable property of the partition criterion (Eq. 3) is that maximizing the within-cluster similarity of objects corresponds to simultaneously minimizing the cross-cluster similarity, which leads to balanced clusters. Since finding an exact minimizer of Eq.3 is an NP-hard problem, usually an approximation based on a relaxed generalized eigenvalue problem is used instead. The classical choice for the pairwise object similarity function is the multiplicative exponential model:

$$f_s(o_i, o_j) \equiv \prod_{k=1}^{M} e^{-\frac{d_k(o_i, o_j)^2}{\sigma_k^2}} \tag{4}$$

The values $d_k(o_i, o_j)$ represent the M abstract differential features which quantify various aspects of the difference between objects o_i and o_j, e.g. spatial distance, color disparity etc. A popular variation of the Normalized Cut algorithm is the recursive bi-partitioning strategy, where the object set is always split into two parts which are then recursively bi-partitioned until a stopping criterion is met. Usually, a stopping criterion based on a threshold for the Normalized Cut value is used.

2.4.2 Differential features for merging In order to be able to model the concept of two segments belonging to the same or different stems, we introduce several groups of features.

Starting point and direction. This group of features concerns the distance between starting points and headings within the segment pair (s_i, s_j). Let $p_{S,k}, p_{E,k}$ denote respectively the start point and end point of segment s_k. Then, we define the unit direction vector of segment s_k as $v_k = \frac{p_{E,k} - p_{S,k}}{||p_{E,k} - p_{S,k}||_2}$. To achieve invariance with respect to choosing the start and end points, we make the direction vector v_i assume the sign which minimizes the norm of the difference in heading $\Delta v_{i,j} = v_i - v_j$. The elements of $\Delta v_{i,j}$ constitute our first 3 features. The last feature in this group corresponds to the Euclidean distance between the starting points of the direction-aligned segments. It is defined as $min(||q_i - q_j||_2, ||r_i - r_j||_2)$, where q_i and r_i are respectively the start and end points of segment s_i according to the direction vector v_i.

Spatial intersection. As already noted (Sec. 2.2), we can regard segments as the axes of fixed-radius cylinders. We make use of this perspective and define a differential feature based on the ratio of the spatial intersection volume of the two cylinders and the volume of one cylinder. Note that since the cylinders have equal lengths and radii, this ratio is the same for both of them. Analytically calculating this intersection volume is rather complex, therefore we turn to a Monte Carlo-type simulation. Formally, let C_1, C_2 denote two cylinders with length l and radius r and define q as the intersection volume ratio between C_1 and C_2. If we uniformly draw a point P from the interior of C_1, q is also the probability that $P \in C_1 \cap C_2$. Let $X_1, X_2, \ldots, X_N$ be binary random variables representing N uniform draws of points $P_1, P_2, \ldots, P_N$ from C_1. Each of the X_i variables assumes a value of 1 if $P_i \in C_1 \cap C_2$, otherwise it has a value of 0. We now see that the $X_1, X_2, \ldots, X_N$ are independent, identically distributed Bernoulli variables with probability of success q. We use the maximum likelihood estimator $\hat{q} = \frac{\sum_{i=1}^{N} X_i}{N}$ to estimate q. Note that because $E(\frac{\sum_{i=1}^{N} X_i}{N}) = \frac{1}{N} Nq = q$, the estimator $\hat{q}$ is also unbiased. To retain the convention of distance between segments, we define the spatial intersection feature value as $1 - \hat{q}$. It attains the minimum value of 0 when the two cylinders overlap completely, and the maximum value of 1 if there is no overlap at all. Fig. 2 shows the concept of the intersection volume feature.

Distance along axis. The final group of features is meant to accurately describe the spatial relationship between two segments. For this purpose, we define the asymmetric distance profile P_D of two segments s_1, s_2 relative to s_1. The profile $P_D(s_1, s_2)$ consists of a series of measurements of the distance between the seg-

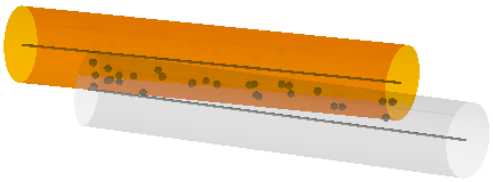


Figure 2: Estimating the cylinder intersection ratio.

ments along equally spaced points on s_1 (see Fig.3). The resulting descriptor is a concatenation of $P_D(s_1, s_2)$ and $P_D(s_2, s_1)$. To make our descriptor symmetric, we set the order of concatenation based on which segment in the pair has the starting point with the smallest x, y and z coordinate (tie breaking in this order).

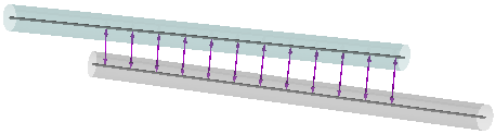


Figure 3: Distance profile for a pair of segments.

2.5 Learning the similarity function

Before calculating the object similarity matrix, it is necessary do define the feature weights σ_i (Eq. 4). This is usually done using trial and error. Another, more systematic approach relies on discretizing the weights and performing a grid search with respect to some measure of quality of the resulting Normalized Cut segmentation. Such an approach can be extremely computationally expensive, since for every combination of weights the entire segmentation must be calculated from scratch. Also, the number of weight combinations increases exponentially with the number of features (or feature groups), rendering this method intractable for large problems. For these reasons, we propose an alternative approach where the similarity is learned from external training data. Although there have been attempts to directly learn the object similarity matrix as part of the Normalized Cut optimization objective (Bach and Jordan, 2003), we take a different route and approximate this task with the one of training a probabilistic classifier which for any pair of segments labels them as belonging to the same or different stems. Our hypothesis is that classifiers which attain a high binary classification rate will also produce high quality segmentations as similarity functions for Normalized Cut. In the rest of this section, we show that manipulating the weights σ_i is equivalent to implicitly specifying a Generalized Linear Model classifier (Sec. 2.5.2). We also address the issue of obtaining the aforementioned 'external training data' for our application of fallen tree detection (Sec. 2.5.3). We show how complex scenarios with multiple overlapping lying stems can be generated by means of simulation. Our classifier is then trained on the simulated data.

2.5.1 Generalized Linear Models Generalized Linear Models are a regression technique which generalizes ordinary linear regression for certain cases where its assumptions do not hold. In particular, consider a regression task on N predictor variables $X_1, X_2, \ldots, X_N$ with a response variable Y. The task is to estimate the expectation of Y conditioned on the predictor variables $E(Y|X_1, X_2, \ldots, X_N)$. In a GLM, we assume the following relationship between the conditional mean μ of the distribution of Y and the linear regression sum of the predictor variables:

$$g(\mu(X)) = \theta_0 + \theta X \tag{5}$$

In the above, g is called the *link function*. The vector θ represents the linear regression coefficients, and θ_0 is the intercept. In order to define a concrete GLM, we need to specify the link function as well as the distribution which generates the response variable.

Binary Logistic Regression. Consider a regression task where the response variable Y is binary. This corresponds to the occurrence of some event, or the lack thereof. Such a binary random variable has a Bernoulli distribution with p probability of success (the event occurring). If we encode a success in a single trial as '1' and a failure as '0', then p is also the mean of Y: $E(Y) = 1 * p + 0 * (1 - p) = p$. Define the *logistic function* and its inverse (*logit*):

$$f_{log}(z) = \frac{1}{1 + e^{-z}}, f_{log}^{-1}(\mu) = g_{logit}(\mu) = ln\frac{\mu}{1 - \mu} \tag{6}$$

The logistic function maps the real line onto the interval $(0; 1)$, giving a probability interpretation to the regression sum. We thus arrive at the binary logistic model (compare to Eq. 5):

$$g_{logit}(\mu(X)) = ln\frac{\mu(X)}{1 - \mu(X)} = \theta_0 + \theta X \tag{7}$$

Fitting Binary GLMs. Fitting a binary GLM involves finding the weights θ which maximize the log-likelihood over the training data $(X_i, Y_i)_{i=1..N}$:

$$l_\theta = \sum_{i=1}^{N} lnP(Y = Y_i|X_i; \theta) \tag{8}$$

For a binary GLM, the conditional mean is equal to the probability of success. Therefore:

$$P(Y = 1|X; \theta) = \mu(X), P(Y = 0|X; \theta) = 1 - \mu(X) \tag{9}$$

From Eq. 5 we have that $\mu(X) = g^{-1}(\theta_0 + \theta X)$ Finally, by defining $h(X) = g^{-1}(\theta_0 + \theta X)$ we can rewrite Eq. 8 as:

$$l_\theta = \sum_{i=1}^{N} Y_i ln[h(X_i)] + (1 - Y_i) ln[1 - h(X_i)] \tag{10}$$

By plugging a specific link function into Eq. 10, we obtain the concrete optimization objective. Optimization (with respect to the weights θ) is usually carried out using Newton's method.

2.5.2 Similarity function as a GLM Shi and Malik (2000) note that the similarity function for a pair of primitive elements should reflect the likelihood that the two elements are part of the same cluster. Indeed, we may regard this as a regression task on the set of all object pairs $O_P = O \times O$ with the differential features $d_{i,j}$ (Sec. 2.4.1):

$$\{d_{i,j} \equiv [d_1(o_i, o_j), \ldots, d_M(o_i, o_j)]^T\}_{i=1..N, j=1..N} \tag{11}$$

For an element $o_{i,j} \in O_P$, we want to estimate the posterior probability that objects o_i and o_j belong to the same cluster A_x, conditioned on the differential features:

$$f_s(o_{i,j}) \approx P(o_i \in A_x \wedge o_j \in A_x|d_{i,j}) \tag{12}$$

The response variable Y is therefore binary. We can encode the event that the objects o_i and o_j belong to the same cluster with value 1 and the opposite event with value 0. Then, f_s approximates $P(Y = 1|d_{i,j}) = \mu_Y(d_{i,j})$, the conditional mean of Y.

Now we take a closer look at the canonical Normalized Cut sim-

ilarity function. We can rewrite Eq. 4 as:

$$f_s(o_i, o_j) = e^{-\sum_{k=1}^{M} \theta_k d^2_{i,j,k}} \quad (13)$$

where $\theta \equiv [\frac{1}{\sigma_1^2}, \frac{1}{\sigma_2^2}, \ldots, \frac{1}{\sigma_M^2}]^T$. Further, if we define $r_{i,j} \equiv [d^2_{i,j,1}, d^2_{i,j,2}, \ldots, d^2_{i,j,M}]^T$ and set $\theta_0 = 0$, we can achieve the following concise form:

$$f_s(o_i, o_j) = e^{-(\theta_0 + \theta^T r_{i,j})} \quad (14)$$

Note that because all the values $\theta_i, r_{i,j}$ are nonnegative, we can guarantee that the sum $\theta_0 + \theta^T r_{i,j}$ is also nonnegative. This coincides with the fact that the function $h(z) = e^{-z}$ maps the interval $[0; \infty)$ onto the interval $(0; 1]$, which corresponds to valid probability values. We can rewrite Eq. 14 as:

$$-ln[f_s(o_i, o_j)] = -ln[\mu_Y(r_{i,j})] = \theta_0 + \theta^T r_{i,j} \quad (15)$$

Comparing with Eq. 5, we immediately see that Eq. 15 specifies a Generalized Linear Model with a link function $g(\mu) = -ln(\mu)$ on the squared differential features. The model for generating the response variable is the Bernoulli distribution. This entire derivation shows that whenever we define a set of weights σ_i for the Normalized Cut similarity function, we are implicitly defining a probabilistic regression model. This could be interpreted as a theoretical justification for Eq. 4, but also as an explanation why it performs so well in practice. Throughout this paper, we will refer to the GLM defined by Eq. 15 as *negative exponential regression*.

Optimizing the model. Since negative exponential regression is a binary GLM, we can apply the mathematical apparatus described in Section 2.5.1 in order to optimize the model on a training set. We could simply plug in the inverse of the link function $g(\mu) = -ln(\mu)$ into Eq. 10 and use any numerical optimization technique to obtain the maximum likelihood solution. However, due to how our GLM is formulated, we would have to include nonnegativity constraints for every weight θ, making the optimization problem harder than necessary. Therefore, we decided to slightly reformulate Eq. 14 to allow for negative weights:

$$f_{s'}(r) = e^{-|\theta_0 + \theta^T r|} \quad (16)$$

Now we can find the maximum likelihood weights by solving an unconstrained optimization problem. To do so, we apply a variation of Newton's method. Assume $\theta' = [\theta_0, \theta_1, \ldots, \theta_M]^T$ and $r'_i = [1, r_{i,1}, \ldots, r_{i,M}]^T$. Plugging $f_{s'}$ into Eq. 10 (substituting h) and differentiating twice with respect to the weights θ' yields:

$$\nabla l(\theta') = \frac{\partial l(\theta')}{\partial \theta'} = \sum_{i=1}^{N} r'_i \frac{sgn(f_{s'}(r_i))(f_{s'}(r_i) - Y_i)}{1 - f_{s'}(r_i)} \quad (17)$$

$$\nabla^2 l(\theta') = \frac{\partial^2 l(\theta')}{\partial \theta' \partial \theta'^T} = \sum_{i=1}^{N} r'_i r'^T_i \rho(r_i)[\frac{Y_i}{1 - f_{s'}(r_i)} - \rho(r_i) - 1] \quad (18)$$

In the above, the function $\rho(x) = \frac{f_{s'}(x)}{1 - f_{s'}(x)}$, and $sgn(x)$ refers to the sign function. The optimization is carried out in an iterative fashion starting from an initial guess θ'_0 and updating the solution using the Newton ascent direction $p_{i+1} = -[\nabla^2 l(\theta'_i)]^{-1} \nabla l(\theta'_i)$:

$$\theta'_{i+1} = \theta'_i + \alpha_{i+1} p_{i+1} \quad (19)$$

We apply a backtracking line search on the step lengths α_i to ensure they satisfy the Armijo condition (sufficient increase in the function value):

$$l(\theta'_i + \alpha_{i+1} p_{i+1}) \geq l(\theta'_i) + c_1 \alpha_{i+1} \nabla l(\theta'_i)^T p_{i+1} \quad (20)$$

where c_1 is a small positive constant (we used a value of 0.001). The result of this optimization is the regression model given by the weights θ_i.

2.5.3 Simulating fallen stem scenarios We set up a virtual fallen stem generation pipeline inspired by a similar approach due to Enzweiler and Gavrila (2008) applied to virtual pedestrian appearance synthesis in raster images. The goal is for the classifier to learn the decision boundary between pairs of segments which belong to the same stem versus pairs which belong to different stems. As input for the simulation, a set of sample fallen tree prototypes is required, in the form of their individual point subsets. We then calculate an approximate skeleton for each

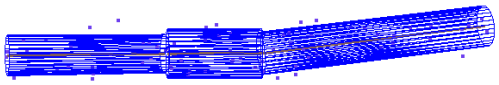
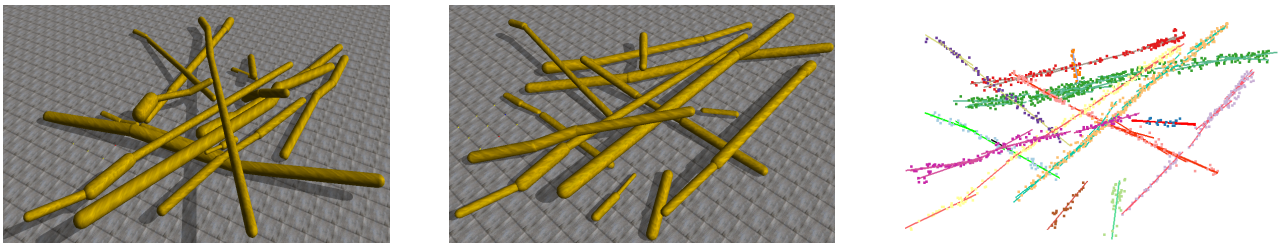


Figure 4: Cylinder-based representation of sample stem.

prototype based on least-squares line fitting. From the skeleton, a cylindrical representation can be computed (Fig.4). The cylinders define the physical behavior of the fallen stems in the simulation. We can now repeatedly randomly pick a sample tree and drop it from a certain height onto the ground plane. The force of gravity pulls the stem towards the ground, while the cylinder rigidity constraints prevent the stems from passing through each other (Fig. 5(a), 5(b)). Thus we can obtain rather complex scenarios with multiple interlaced stems lying on the ground in several layers. The translations and orientations defined by the final positions of the individual trees can then be applied to the point sets and segments associated with the corresponding tree prototypes (Fig. 5(c)). To generate new positive examples (segments belonging to the same stems), we recalculate the representative segments (see Section 2.3) for each tree copy, which can lead to new spatial configurations of segment pairs. The negative examples are defined by multiple stems lying in close proximity to each other. The set of segments is completely labeled (since the segment membership in the individual trees is known by construction) and it can be regarded as the training set in a supervised learning setting. We now specify the top-level simulation loop (Alg. 1): The algorithm initializes the training set with a random stem scenario and trains the classifier c on it. Then, the training set is iteratively augmented. In each iteration, a new stem scenario is generated and its elements are classified by c. Note that we obtain not only the class label, but a continuous decision value normalized on the interval $[0; 1]$. Elements whose continuous decision lies far enough (more than $pThr$) from the interval bounds 0 and 1 are considered to lie in the classifier's uncertainty region and are added to the training set, since they contain relevant information about the decision boundary. The classifier c is then retrained on the augmented training set and the iteration continues. We repeat the loop a predefined number of iterations. This approach can be viewed as a simple implementation of the active learning paradigm, since we are trying to actively guide the learning

Algorithm 1 Learning segment pair classifier

```
1:  procedure LEARNCLASSIFIER(pThr)
2:      trainSet ← createSampleScenario()
3:      classifier ← trainClassifier(trainSet)
4:      for i = 1..Niter do
5:          auxSet ← createSampleScenario()
6:          for o ∈ auxSet do
7:              p(o) ← getContinuousDecision(classifier,o)
8:              if p(o) > pThr ∧ p(o) < 1 − pThr then
9:                  trainSet ← trainSet ∪ o
10:         classifier ← trainClassifier(trainSet)
```



(a) (b) (c)

Figure 5: Simulating fallen stems. (a) and (b) depict randomly generated scenarios, (c) shows the ALS points transformed according to the rotations and translations of the stems in scenario (b).

process by selecting the most informative (most uncertain) exemplars. The described strategy can use any classifier which is capable of supplying a continuous decision value as its output. The result of Alg. 1 is the probabilistic classifier c, which can be applied as the similarity function for Normalized Cut either directly or after undergoing a power transformation (Sec. 2.5.4).

2.5.4 Power transformations. During our initial experiments with setting the weights σ_i in Eq. 4, we noticed that similarity functions with values more closely clustered around 0 resulted in better segmentations. To model this, we introduce a power transformation $u(x) = x^z, z > 1$ of the similarity function. Since the similarity values lie in $[0; 1]$, the transformed values will also be contained in this interval. Also, by forcing the exponent z to be greater than one, the power transformation shifts all values (except 0 and 1) closer to zero. Note that this transformation changes the relative distances between elements, but not the ordering of elements. In practice, we use $u(x)$ as a post-processing function for the output of the classifier c described in Sec. 2.5.3.

3. EXPERIMENTS

3.1 Material

The developed approach was tested on two sample plots located in the Bavarian Forest National Park ($49°3'19''$ N, $13°12'9''$ E), which is situated in South-Eastern Germany along the border to the Czech Republic. From 1988 to 2010, a total of 5800 ha of the Norway spruce stands died off because of a bark beetle (*Ips typographus*) infestation (Lausch et al., 2013). The dead trees were not removed from the area and form the basis for the study. The airborne full waveform data were acquired using a Riegl LMS-Q560 scanner in April 2011 in a leaf-off condition with a nominal point density of 30 points/m^2. The vertical sampling distance was 15 cm, the pulse width at half maximum reached 4 ns and the laser wavelength was 1550 nm. The flying altitude of 400 m resulted in a footprint size of 20 cm. The collected full waveforms were decomposed according to a mixture-of-Gaussians model (Reitberger et al., 2008) to obtain a 3D point cloud. The plot characteristics are summarized in Table 1, whereas the point cloud of Plot B is depicted in Fig.6.

Plot	Size [ha]	Trees/ha	Fallen stems/ha	Deciduous [%]	Overstory cover [%]
A	1.2	230	200	28	20
B	0.86	210	150	45	65

Table 1: Properties of sample plots

3.2 Reference data

To create reference data for our merging algorithm, we first generated a set of probable stem segments according to the processing pipeline described in Sec. 2.2. We then selected the segments

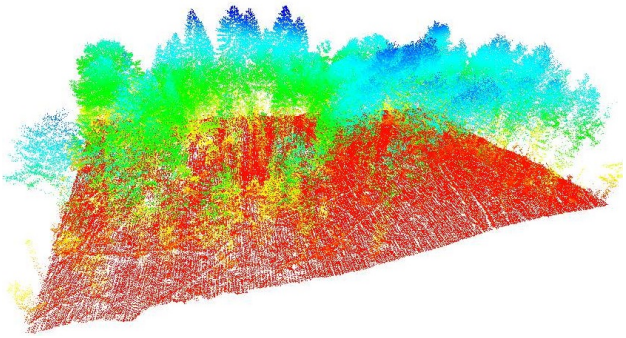


Figure 6: Point cloud of Plot B colored by height over DTM.

actually corresponding to stems in the ALS data and manually labeled them into groups forming individual trees. The total number of stems obtained in this manner was 235 and 196 respectively for Plot A and Plot B.

3.3 Evaluation strategy

In order to be able to make quantitative statements about the quality of a segmentation produced by Normalized Cut in relation to a reference clustering, we need a way to compare two clusterings of the same set of objects. Designing such methods and studying their properties is an active area of research. For a comprehensive review, see Albatineh et al. (2006). For this study, we chose two methods based on counting pairwise relationships between objects: the adjusted Rand index and the Jaccard index. They are both normalized on the interval $[0; 1]$ with the value of 1 indicating that the clusterings are identical, and a value of 0 representing total dissimilarity.

3.4 Detailed experiments

We designed and conducted three groups of experiments to verify several hypotheses proposed in this work. As a baseline method for all experiments, we used a coarse grid search on the space defined by four weights $\{\sigma_i\}_{i=1..4}$ discretized on the set $[0; 1]$ into 5 equally spaced values. This yields a total of $5^4 = 625$ configurations per plot. The four weights correspond to the feature groups: starting point distance, difference in direction, spatial intersection, distance profile (Sec. 2.4). For each weight configuration (for both plots), we recreated the similarity matrix according to Eq. 4 and calculated the Normalized Cut segmentation. We recorded the adjusted Rand index and Jaccard index with respect to the manual reference segmentation and the number of clusters (fallen stems) found.

3.4.1 Similarity function as classifier This group of experiments was supposed to answer the question whether a similarity function's performance as a binary classifier is correlated with the quality of the segmentation it produces (Sec. 2.5). For each plot

we trained three distinct classifiers on the entire data and used them as the similarity function for normalized cut after applying the power transformation (2.5.4). To eliminate the influence of the stopping criterion, we used a 'perfect' stopping criterion which had knowledge of the actual segment labels and thus was able to terminate the recursive bi-partitioning once all segments in a cluster had the same label.

3.4.2 Similarity function from virtual data In this group of experiments, we investigated the quality of the similarity function learned from the virtually generated data (Sec. 2.5.3). To obtain sample fallen tree prototypes for the simulation, we manually segmented ca. 180 stems from ALS data. We applied the learned function to the two test plots and compared the results against the grid search. We also use the 'perfect' stopping criterion for this category.

3.4.3 Segmentation with real stopping criterion Finally, we studied the effects of applying a 'real' stopping criterion to the Normalized Cut segmentation based on the similarity function obtained from the virtually generated data. We used a threshold on the Normalized Cut value combined with an oriented bounding box condition as the stopping criterion.

4. RESULTS AND DISCUSSION

In this section, we present the quantitative results of the experiments described in Section 3.4. We use the following abbreviations for the classifiers: SVM - support vector machine, LR - logistic regression, NER - negative exponential regression, GS - the baseline grid search. The two clustering similarity measures are referred to as: ARI - adjusted Rand index, JI - Jaccard index.

4.1 Similarity function as classifier

We used the manually labeled reference data to train three classifiers (Section 2.5): an SVM (in regression mode to obtain a continuous decision value), the NER as well as the LR model. For each plot, all of the segment pairs were used as training data since the GS was also performed with respect to the entire segment set. The exponent of the power transformation was found using a line search. The results are summarized in Table 3, whereas Table 2 characterizes the best solutions found by the baseline GS. For Plot B, all three classifiers are able to beat the best GS weights. Although the LR and NER models are slightly inferior to the best solution found by GS on Plot A, it should be noted that both of them are still significantly better than choosing random weights (compare to mean value in Tab. 2). Furthermore, the LR and NER models had respectively a 99.5% and 93.2% chance of being better then a randomly drawn weight configuration on the grid. The results from this section show that there is a correlation between classification accuracy and the segmentation quality as measured by the both ARI and the JI. Moreover, the segmentation quality obtained with the classifiers is similar to the best results obtained with GS. Together, these two facts support the hypothesis put forward in Sec. 2.5 about learning the similarity function via training a classifier. Also, the quality of the baseline GS results is perhaps surprisingly high, which can explain the widespread use of the Normalized Cut algorithm without a formal procedure for specifying the attribute weights.

Plot	max_{ARI}	μ_{ARI}	σ_{ARI}	max_{JI}	μ_{JI}	σ_{JI}
A	0.948	0.912	0.02	0.903	0.840	0.03
B	0.982	0.872	0.06	0.965	0.779	0.10

Table 2: Grid search - results

Classifier	Classification accuracy	ARI	JI	Num. trees
Plot A				
SVM	0.954	0.956	0.917	277
LR	0.931	0.944	0.894	270
NER	0.919	0.934	0.877	310
Plot B				
SVM	0.959	0.994	0.989	200
LR	0.930	0.990	0.980	203
NER	0.919	0.984	0.969	210

Table 3: Classifier with power transformation - results

4.2 Similarity function from virtual data

We now discuss the quality of the similarity function trained on the simulated virtual stem data (Sec. 2.5.3). We carried out the learning procedure (Alg. 1) for two models, the LR and the NER. The power transformation was also applied with the exponent found using a linear search procedure. For the LR model, the test classification accuracy was 92.8% and 92.5% respectively on Plot A and Plot B, whereas the NER model attained a value of 91.8% on both plots. We applied each model to both plots obtaining the Rand and Jaccard index values presented in Table 4.

Model	Exponent z	ARI	JI	Num. trees
Plot A				
LR	8	0.956	0.917	272
NER	2.5	0.957	0.918	277
Plot B				
LR	13.5	0.962	0.927	216
NER	2.5	0.982	0.965	211

Table 4: Segmentation quality - learning from virtual samples

The NER model was able to outperform the GS baseline for both plots, despite the GS's advantage of having access to the test data. This confirms the suitability of this similarity model for use with Normalized Cut. For Plot A, LR performs almost equally as well, while exhibiting a somewhat weaker performance on Plot B. The fact that the single NER model leads to such strong segmentation quality on both test plots supports the hypothesis that the proposed simulation is accurate enough so that the learned classifier can generalize well. Furthermore, it is also an argument in favor of the ideas described in Sec. 2.5, i.e. learning the similarity function reduced to learning the classifier. One small weakness of our method is the necessity to perform the line search for the optimal exponent in the power function on test data. We envision making this a part of the learning procedure in the hope that the exponent learned from training data will also generalize well across plots. The equality of the optimal exponents for NER bolsters this hope.

4.3 Segmentation with real stopping criterion

As described in Sec. 3.4, for the first two experiment groups we used an unrealistic stopping criterion which knew the actual segment labels. We now apply a 'real' stopping criterion using a threshold on the Normalized Cut value to investigate how much the segmentation quality will drop. A line search was performed on the threshold values to find the one maximizing the adjusted Rand index for each plot, for the NER and LR models. The NER model experienced a drop in the ARI to values of 0.926 and 0.962 respectively on Plot A and Plot B. The LR model seemed to be less affected, attaining ARI values of 0.951 and 0.952. While there is some room for improvement in the stopping criterion, the

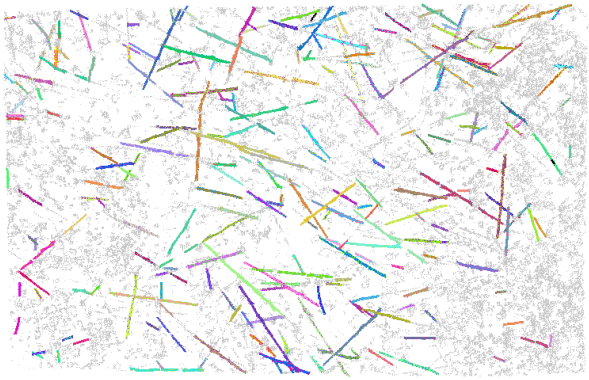


Figure 7: Segmentation result - Plot A.

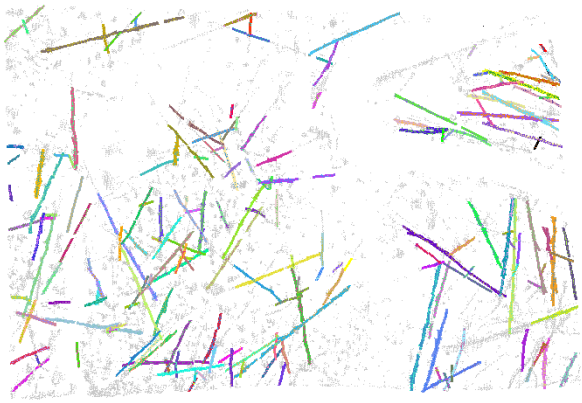


Figure 8: Segmentation result - Plot B.

overall performance is still good enough for a practical application. The segmentation results are depicted in Figures 7 and 8.

5. CONCLUSIONS

In this work we have presented an integrated method for detecting fallen stems from ALS point clouds. A set of differential features has been proposed which can be the basis for merging stem segments into entire stems using the Normalized Cut algorithm. We have then shown that manipulating the attribute weights in the Normalized Cut similarity function is equivalent to specifying a regression model, which gave rise to the idea of reducing the problem of learning the similarity function from a reference segmentation to training a binary classifier on the labeled data. Our results show that such an approximation can yield high-quality segmentation results, while saving vast amounts of computational effort necessary to optimize the weights in a grid search scheme. Finally, we have set up a simulation where virtual fallen stems are randomly generated to produce complex scenarios, which are then used to train a similarity function. Interestingly, this similarity function trained on synthesized data is able to outperform the grid search methods (which have access to the real data), a result that attests to its generalization capability. This would allow us to perform the learning only once and reuse its output on multiple plots. We believe that the proposed methodology can be utilized for other applications in remote sensing which require learning the concept of similarity between object parts that belong to a greater whole. As a future step, we would like to validate our entire processing pipeline on data for which ground truth is available in order to obtain estimates of how reliably our method can detect single fallen trees.

REFERENCES

Albatineh, A. N., Niewiadomska-Bugaj, M. and Mihalko, D., 2006. On similarity indices and correction for chance agreement. *J. Classif.*, 23(2), pp. 301–313.

Bach, F. R. and Jordan, M. I., 2003. Learning spectral clustering. In: *Adv. Neur. In.*, Vol. 16.

Blanchard, S. D., Jakubowski, M. K. and Kelly, M., 2011. Object-Based Image Analysis of Downed Logs in Disturbed Forested Landscapes Using Lidar. *Remote Sensing*, 3(12), pp. 2420–2439.

Elmqvist, M., 2002. Ground surface estimation from airborne laser scanner data using active shape models. In: *Photogrammetric Computer Vision-ISPRS Commission III Symposium*, Vol. XXXIV, Part A, pp. 114–118.

Enzweiler, M. and Gavrila, D., 2008. A mixed generative-discriminative framework for pedestrian classification. In: *Computer Vision and Pattern Recognition*, pp. 1–8.

Freedman, B., Zelazny, V., Beaudette, D., Fleming, T., Johnson, G., Flemming, S., Gerrow, J. S., Forbes, G. and Woodley, S., 1996. Biodiversity implications of changes in the quantity of dead organic matter in managed forests. *Environ. Rev.*, 4(3), pp. 238–265.

Frome, A., Huber, D., Kolluri, R., Bülow, T. and Malik, J., 2004. Recognizing Objects in Range Data Using Regional Point Descriptors. *Computer Vision - ECCV 2004*, 3023, pp. 224–237.

Heurich, M., 2008. Automatic recognition and measurement of single trees based on data from airborne laser scanning over the richly structured natural forests of the Bavarian Forest National Park. *Forest Ecol. Manag.*, 255, pp. 2416–2433.

Lan, G., DePuy, G. W. and Whitehouse, G. E., 2007. An effective and simple heuristic for the set covering problem. *Eur. J. Oper. Res.*, 176, pp. 1387–1403.

Lausch, A., Heurich, M. and Fahse, L., 2013. Spatio-temporal infestation patterns of Ips typographus (L.) in the Bavarian Forest National Park, Germany. *Ecological Indicators*, 31, pp. 73–81.

Lindberg, E., Hollaus, M., Mücke, W., Fransson, J. E. S. and Pfeifer, N., 2013. Detection of lying tree stems from airborne laser scanning data using a line template matching algorithm. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. II-5, Part W2, pp. 169–174.

Muecke, W., Deak, B., Schroiff, A., Hollaus, M. and Pfeifer, N., 2013. Detection of fallen trees in forested areas using small footprint airborne laser scanning data. *Can. J. Remote Sens.*, 39, pp. S32–S40.

Polewski, P., Yao, W., Heurich, M., Krzystek, P. and Stilla, U., 2014. Detection of fallen trees in ALS point clouds of a temperate forest by combining point/primitive-level shape descriptors. In: *Gemeinsame Tagung 2014 der DGfK, der DGPF, der GfGI und des GiN. DGPF Tagungsband*, Vol. 23.

Reitberger, J., Krzystek, P. and Stilla, U., 2008. Analysis of full waveform LIDAR data for the classification of deciduous and coniferous trees. *Int. J. Remote Sens.*, 29, pp. 1407–1431.

Rusu, R., Marton, Z., Blodow, N. and Beetz, M., 2008. Learning informative point classes for the acquisition of object model maps. In: *10th International Conference on Control, Automation, Robotics and Vision*, pp. 643–650.

Shi, J. and Malik, J., 2000. Normalized cuts and image segmentation. *IEEE T. Pattern Anal.*, 22(8), pp. 888–905.

Weaver, J. K., Kenefic, L. S., Seymour, R. S. and Brissette, J. C., 2009. Decaying wood and tree regeneration in the Acadian Forest of Maine, USA. *Forest Ecol. Manag.*, 257, pp. 1623–1628.

Woodall, C., Heath, L. and Smith, J., 2008. National inventories of down and dead woody material forest carbon stocks in the United States: Challenges and opportunities. *Forest Ecol. Manag.*, 256(3), pp. 221–228.

Yao, W., Krzystek, P. and Heurich, M., 2012. Identifying Standing Dead Trees in Forest Areas Based on 3D Single Tree Detection From Full Waveform Lidar Data. In: *ISPRS Annals of the Protogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. I-7, pp. 359–364.