A TIMBER Framework for Mining Urban Tree Inventories Using Remote Sensing Datasets

Yiqun Xie
Dept. of Computer Science
University of Minnesota
Email: xiexx347@umn.edu

Han Bao
Dept. of Geography
University of Minnesota
Email: baoxx095@umn.edu

Shashi Shekhar
Dept. of Computer Science
University of Minnesota
Email: shekhar@umn.edu

Joseph Knight
Dept. of Forest Resources
University of Minnesota
Email: jknight@umn.edu

***Abstract*—Tree inventories are important datasets for many societal applications (e.g., urban planning). However, tree inventories still remain unavailable in most urban areas. We aim to automate tree identification at individual levels in urban areas at a large scale using remote sensing datasets. The problem is challenging due to the complexity of the landscape in urban scenarios and the lack of ground truth data. In related work, tree identification algorithms have mainly focused on controlled forest regions where the landscape is mostly homogeneous with trees, making the methods difficult to generalize to urban environments. We propose a TIMBER framework to find individual trees in complex urban environments and a Core Object REduction (CORE) algorithm to improve the computational efficiency of TIMBER. Experiments show that TIMBER can efficiently detect urban trees with high accuracy.**

I. Introduction

Tree inventories contain meaningful information for urban planning, sustainability, natural resource management, etc. In recent years, the invasive emerald ash borer has caused tree deaths in the tens of millions [1], [2], and is estimated to cost over 10 billion US dollars to manage [1]. Many state and city governments have begun to identify and treat (or remove) all individual ash trees. In addition, fine-scale tree inventories are also important for green infrastructure management in sustainable community planning [3]. However, inventories of individual trees rarely exist in most urban areas due to the difficulty of manual collection (e.g., limited GPS signals under canopies, time-consuming).

We aim to automate the generation of individual tree inventories using high-resolution (e.g., one meter or lower) remote sensing datasets that are publicly available at large-scales. The scope of the present study is to identify the locations and sizes of individual trees in an urban area. The type of remote sensing data that we use is the Normalized Height Model (NHM), which is a single band image whose pixel values represent surface heights. NHMs are LiDAR-derivatives that has been collected and made publicly available at large scales (e.g., state-level or major urban areas across the US).

The tree identification problem is challenging: (1) Trees in urban environments are often mixed with buildings, low-vegetation, lawns, towers, etc; (2) Trees commonly appear in groups with heavy canopy overlaps; and (3) Individual tree inventories are rarely collected (or shared in public) at large scales or in different urban areas, making it difficult to train a generalizable machine learning model that can be applied robustly in different geographical regions.

In NHMs, the canopies of trees are dome-shaped, which makes them similar to mixtures of Guassians. However, Gaussian mixture models (e.g., k-means, expectation-maximization [4]) rely on an input number of clusters, which is unknown in this problem. These models also do not distinguish tree and non-tree structures (e.g., buildings, towers). In recent years, deep learning models have shown promising results for general computer vision problems as well as urban land-use classification [5], [6]. However, they require a large number of training samples from different geographies, which are not available for the tree detection problem. In addition, deep learning models typically target input images of specific sizes (e.g., 416×416). For remote sensing data, this requires additional space partitioning, which tends to break objects on the boundaries into pieces. Tree detection algorithms have also been studied in the field of remote sensing [7], [8], [9], [10]. These algorithms were mainly designed for landscapes that are mostly homogeneous with few types of trees (e.g., pine tree), and the data sources were specifically collected at very high resolution (e.g., centimeters, hyperspectral). Such datasets are still unavailable at larger scales due to their high costs and limited public availability. Typically these algorithms employ watershed segmentation or clustering to delineate tree canopies. However, the segmentation methods tend to get stuck in small local neighborhoods (e.g., sub-tree levels), and cannot avoid non-tree structures in complex urban environments.

We propose a two-phase TIMBER[1] (Tree Inference by Minimizing Bound-and-band ERrors) framework to identify individual trees in urban environments. The first phase infers the locations and sizes of tree-like structures by optimizing tree-like approximators (e.g., Gaussians) to minimize the difference with tree canopy bounds (i.e., bound errors). The second phase integrates additional city infrastructure data (e.g., buildings, roads) to train a deep convolutional neural network to filter out non-tree results. The deep network predictions are formed by the band values of the input remote sensing data, so we consider this training process as a minimization of band errors. A Core Object REduction (CORE) algorithm is also proposed to improve the computational efficiency.

[1]Source code: https://www-users.cs.umn.edu/%7exiexx347/timber.html

978-1-5386-9159-5/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDM.2018.00183

Through detailed experiments we show that the proposed TIMBER framework significantly improves precision, recall and F1-scores compared to related work and the CORE algorithm speeds up execution by 1.5x to 2x.

II. Problem definition

A. Key concept

A *Normalized height model (NHM)*, also known as a canopy height model [8], is a single-band image (satellite view) whose pixel values represent the height of objects (e.g., buildings, trees) instead of colors. It is a LiDAR-derivative and has been collected at large scales (e.g., many states in the US).

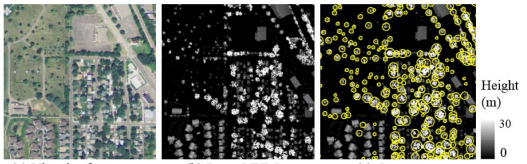

Fig. 1. Example of input and output. (best in color)

B. Formal problem formulation

Inputs:

– A normalized height model in a spatial domain D;
– Min and max tree sizes, r_{min} and r_{max}, to detect in D;

Outputs: Locations and sizes of trees in D;

Objectives:

– Accuracy of tree detection;
– Computational efficiency;

Constraint: Trees of minimum size r_{min} must be recognizable at the spatial resolution s_{nhm} of the NHM.

Fig. 1 shows an example of inputs and outputs, where the shapes of trees are approximated by circles.

III. A TIMBER Framework for Tree Identification

TIMBER has two phases. Phase 1 estimates the locations and sizes of tree-like structures using localized optimizations. To remove the non-tree structures, Phase 2 constructs a deep learning filter using a combination of the detections from Phase 1 and city infrastructure datasets (e.g., buildings, roads).

A. Phase 1: Optimization of tree locations and sizes

In a normalized height model, the structure of a tree is represented by a dome-shaped bound on the top of its canopy, which can be approximated by a mathematical approximator with a varying set of parameters (e.g., Gaussian). We consider the difference between a tree bound and an approximator as the bound error E_b. In Phase 1, TIMBER optimizes a given mathematical approximator to minimize E_b of each tree. Note that Phase 1 yields both trees and tree-like structures (e.g., buildings, towers). The non-trees are filtered out in Phase 2.

Flexible location initializer with local maxima: We use local maxima to initialize the rough locations of trees (i.e., center peaks of tree canopies) [8] in an urban environment. A local maximum in an NHM is defined as a pixel whose value is the largest in a local window.Since trees do not have perfect dome-shapes, the bounds of their canopies are not smooth and have small height fluctuations. This often results in multiple local maxima on top of a tree canopy, or makes a local maximum not at the center of a tree. Thus, the local maxima are just rough estimations of actual tree locations.

To achieve better estimations, our optimization formulation considers flexible locations of a local maximum. That is, all locations within its circular neighborhood of radius r_{min} become candidate centers for its corresponding object and the best location will be returned as the center.

Optimization of mathematical approximators: We use mathematical approximators (e.g., [11], [12]) to approximate the dome-shapes of trees in NHMs by minimizing the bound error E_b between the approximator and the actual tree bound. Our optimization method can be generally applied to dome-shaped mathematical surfaces. The general optimization formulation for a single approximator on a single tree is:

$$\min_{\boldsymbol{\alpha},\boldsymbol{\mu},r} \frac{\|\big(\mathbf{y}(\mathbf{X}) - f_{apx}(\mathbf{X},\boldsymbol{\alpha},\boldsymbol{\mu},r)\big)\cdot \mathbb{1}(\mathbf{X}^T,\boldsymbol{\mu},r)\|_2^2}{\|\mathbb{1}(\mathbf{X}^T,\boldsymbol{\mu},r)\|_1} \quad (1)$$

$$\text{s.t. } \|\boldsymbol{\mu} - \boldsymbol{\mu}_{max}\|_2 \le r_{min} \quad (2)$$

$$r \in \{\lceil \frac{r_{min}}{s_{nhm}} \rceil, \lceil \frac{r_{min}}{s_{nhm}} \rceil + 1, ..., \lceil \frac{r_{max}}{s_{nhm}} \rceil\} \quad (3)$$

$$\mathbb{1}_i(\mathbf{X}^T,\boldsymbol{\mu},r) = \begin{cases} 1, & \text{if } \|(\mathbf{X}_i)^T - \boldsymbol{\mu}\|_2 \le r \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

where the subscript i indicates the i^{th} row of a matrix or vector; $\mathbf{X} \in \mathbb{Z}_+^{N\times 2}$ is a matrix containing locations (i.e., row and column IDs in the input NHM) of all pixels, and N is the total number of pixels; $\boldsymbol{\alpha}$ is the parameter vector of an input approximator function $f_{apx}()$; $\boldsymbol{\mu} \in \mathbb{Z}_+^2$ is a vector of the center location (row and column IDs) of the approximator and r is its radius; $\mathbb{1}()$ is an indicator function defined in Eq. (4) that returns a vector indicating if a location in $\mathbf{X}$ is within the radius r from the center of the approximator; $\mathbf{y}(\mathbf{X})$ is a vector of the actual height values in the NHM at locations in $\mathbf{X}$; and $\boldsymbol{\mu}_{max}$ is the location of a local maximum on the tree.

The objective function aims to minimize the mean difference between the approximator and the tree bound inside a circular spatial domain of size r. Constraint (2) reflects the search distance used in the flexible center formulation. Its limit is set to r_{min} to avoid moving into the center of another tree. Constraint (3) defines the set of candidate radii (in pixels) constrained by the input minimum and maximum tree sizes, r_{min} and r_{max}, and the spatial resolution s_{nhm} of the NHM.

To make our discussion more concrete, we illustrate the optimization process with two approximators, i.e., a negative quadratic function and a Gaussian probabilistic function:

1) Optimization with quadratic surfaces: Negative quadratic functions form dome-shaped mathematical surfaces. Our quadratic approximator f_q is defined as follows:

$$f_q\big((\mathbf{X}_i)^T,\boldsymbol{\alpha},\boldsymbol{\mu},r\big) = -\boldsymbol{\alpha}_1 \cdot \|(\mathbf{X}_i)^T - \boldsymbol{\mu}\|_2^2 + \boldsymbol{\alpha}_2 \quad (5)$$

where subscript i denotes the i^{th} row of a matrix; $\boldsymbol{\alpha}_1 \in \boldsymbol{\alpha}$ controls the vertical stretch of the dome-shape; and $\boldsymbol{\alpha}_2 \in \boldsymbol{\alpha}$ adjusts the vertical intercept (height) of f_q; $\boldsymbol{\alpha} > \mathbf{0}$.

According to Eq. (1), the decision variables to optimize are $\boldsymbol{\alpha}$, $\boldsymbol{\mu}$ and r, among which $r \in \mathbb{Z}_+$ and $\boldsymbol{\mu} \in \mathbb{Z}_+^2$ are integer variables in units of pixels of the input NHM. In addition, r is a threshold in the indicator function $\mathbb{1}()$, making it difficult to derive its optimal value in close-form or by gradient descent.

Since this optimization is done individually for each local maximum, the total number of combinations of r and $\boldsymbol{\mu}$ to consider is in fact limited, especially considering that tree sizes are often not very large numbers in real-world urban environments. Thus, we enumerate through all combinations of $\boldsymbol{\mu}$ and r to obtain the optimal solutions. For parameter vector $\boldsymbol{\alpha}$, we have the following Theorems 1 and 2:

Theorem 1. *For a fixed combination of (r, $\boldsymbol{\mu}$), the solutions for $\boldsymbol{\alpha}$ can be evaluated in closed form:*

$$\boldsymbol{\alpha}_1 = \frac{(\sum_i \mathbb{1}_i)^{-1}(\sum_i \mathbf{Z}_i^2 \mathbb{1}_i)(\sum_i \mathbf{y}_i \mathbb{1}_i) - (\sum_i \mathbf{Z}_i^2 \mathbf{y}_i \mathbb{1}_i)}{(\sum_i \mathbf{Z}_i^4 \mathbb{1}_i) - (\sum_i \mathbb{1}_i)^{-1} \cdot (\sum_i \mathbf{Z}_i^2 \mathbb{1}_i)^2}$$

$$\boldsymbol{\alpha}_2 = [(\textstyle\sum_i \mathbf{y}_i \mathbb{1}_i) + \boldsymbol{\alpha}_1(\sum_i \mathbf{Z}_i^2 \mathbb{1}_i)] / \sum_i \mathbb{1}_i$$

where $\mathbf{Z}_i = \|(\mathbf{X}_i)^T - \boldsymbol{\mu}\|_2$, $\mathbb{1}_i = \mathbb{1}((\mathbf{X}_i)^T, \boldsymbol{\mu}, r)$, $\mathbf{y}_i = \mathbf{y}(\mathbf{X}_i)$.

Proof. For a valid and fixed combination (r, $\boldsymbol{\mu}$), we can get rid of Constraints (2) and (3) since they must be satisfied. Thus, we are left with:

$$\begin{aligned}
&\|\big(\mathbf{y}(\mathbf{X}) - f_q(\mathbf{X}, \boldsymbol{\alpha}, \boldsymbol{\mu}, r)\big) \cdot \mathbb{1}(\mathbf{X}^T, \boldsymbol{\mu}, r)\|_2^2 / \|\mathbb{1}(\mathbf{X}^T, \boldsymbol{\mu}, r)\|_1 \\
=&\|\big(\mathbf{y}(\mathbf{X}) + (\boldsymbol{\alpha}_1 \cdot \mathbf{Z}^{\circ 2} + \boldsymbol{\alpha}_2) \cdot \mathbb{1}(\mathbf{X}^T, \boldsymbol{\mu}, r)\|_2^2 / \|\mathbb{1}(\mathbf{X}^T, \boldsymbol{\mu}, r)\|_1 \\
=&\textstyle\sum_i \big[\big(\boldsymbol{\alpha}_1^2 \mathbf{Z}_i^4 - 2\boldsymbol{\alpha}_1\boldsymbol{\alpha}_2\mathbf{Z}_i^2 + \boldsymbol{\alpha}_2^2 + 2\boldsymbol{\alpha}_1\mathbf{Z}_i\mathbf{y}_i - 2\boldsymbol{\alpha}_2\mathbf{y}_i + \mathbf{y}_i^2\big) \\
&\cdot \mathbb{1}_i\big] / \textstyle\sum_i \mathbb{1}_i
\end{aligned}$$

where $\mathbf{Z}^{\circ 2}$ denotes the Hadamard (element-wise) square of $\mathbf{Z}$.

Taking partial derivatives and setting them to 0s, we get:

$$\begin{cases} \boldsymbol{\alpha}_1(\sum_i \mathbf{Z}_i^4 \mathbb{1}_i) - \boldsymbol{\alpha}_2(\sum_i \mathbf{Z}_i^2 \mathbb{1}_i) + \sum_i \mathbf{Z}_i^2 \mathbf{y}_i \mathbb{1}_i = 0 \\ -\boldsymbol{\alpha}_1(\sum_i \mathbf{Z}_i^2 \mathbb{1}_i) + \boldsymbol{\alpha}_2(\sum_i \mathbb{1}_i) - \sum_i \mathbf{y}_i \mathbb{1}_i = 0 \end{cases} \tag{6}$$

Solving this linear system with two equations and two unknowns, we get the solutions stated in the theorem. □

Next we show that valid solutions are unique via Thm. 2.

Theorem 2. *The solutions of $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ are unique when the mathematical surface is a valid approximation of a tree-shape, and the result of the negative quadratic function (Eq. (5)) is in the valid range $(0, h_{max}]$, where h_{max} is the maximum height of a tree in the spatial domain of interest.*

Proof. First, trees have concave shapes rather than convex or flat shapes in the normalized height models. This requires $\boldsymbol{\alpha}_1$ be positive in the negative quadratic function (Eq. (5)). If $\boldsymbol{\alpha}_1 \leq 0$, the combination or hypothesis (r, $\boldsymbol{\mu}$) should be directly rejected. Then, based on the derivatives, the only situation when the linear system does not have a single solution appears when $(\sum_i \mathbf{Z}_i^4) - (\sum_i \mathbb{1}_i)^{-1} \cdot (\sum_i \mathbf{Z}_i^2)^2$ (last term of $\boldsymbol{\alpha}_1$'s solution) is 0: (1) Possible outcome 1: infinite number of ($\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2$) which satisfy the conditions in Eq. (6). However, in this case, the optimal solution for $\boldsymbol{\alpha}_1$ must be negative ($\boldsymbol{\alpha}_2$ must be positive) in order to achieve the minimum objective, according to the expansion of the objective function at the beginning of Thm. 1's proof (note that $\mathbf{y}(\mathbf{X}_i) \geq 0$, $\forall i$); (2) Possible outcome 2: $\boldsymbol{\alpha}_1$ becomes infinitely large. This is also invalid because $\boldsymbol{\alpha}_1 \to +\infty$ will make the result of Eq. (5) exceed h_{max} (also another type of non-tree shape). □

2) Optimization with Gaussian surfaces: Our TIMBER framework can be applied to general mathematical approximators. The negative quadratic function provides a concrete example for the polynomial family, and here we show TIMBER's use with a Gaussian approximator for the exponential family. We skip the proofs to reduce redundancy.

Since we use circles to approximate the 2D shapes of trees from a satellite view, the covariance matrix of the Gaussian function is diagonal and the diagonal elements are the same (i.e., same variance σ^2 for each direction). In addition, original Gaussian functions have inflection points (one dimension) or hyperplanes (multiple dimensions) where the second order derivatives change signs (e.g., from a concave dome-shape to a convex bow-shape). Since trees have concave shapes, our approximator uses only the "concave" part, that is, the dome-surface towards the inside of the inflection boundary. In each direction of a Gaussian, the inflection points are located at $(\mu \pm \sigma)$. Thus, each element in $((\mathbf{X}_i)^T - \boldsymbol{\mu})$ is rescaled to the range of $[0, \pm\sigma]$. The Gaussian approximator is:

$$f_g\big((\mathbf{X}_i)^T, \boldsymbol{\alpha}, \boldsymbol{\mu}, r\big) = \boldsymbol{\alpha}_1 \cdot \frac{\exp(-\mathbf{d}_i^T(\sigma^2\mathbf{I})^{-1}\mathbf{d}_i)}{(\sqrt{(2\pi)^2|\sigma^2\mathbf{I}|})} + \boldsymbol{\alpha}_2 \tag{7}$$

where $\mathbf{d}_i = \frac{\sigma((\mathbf{X}_i)^T - \boldsymbol{\mu})}{r}$. The solutions to $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ are given by (in the final solution σ is canceled out due to rescaling):

$$\boldsymbol{\alpha}_1 = \frac{\sum_i \mathbf{e}_i \mathbf{y}_i \mathbb{1}_i - (\sum_i \mathbb{1}_i)^{-1}(\sum_i \mathbf{y}_i \mathbb{1}_i)(\sum_i \mathbf{e}_i \mathbb{1}_i)}{(2\pi)^{-1}(\sum_i \mathbf{e}_i^2 \mathbb{1}_i) - (2r\pi)^{-1}(\sum_i \mathbf{e}_i \mathbb{1}_i)^2}$$

$$\boldsymbol{\alpha}_2 = \big[2(\textstyle\sum_i \mathbf{y}_i \mathbb{1}_i) - \boldsymbol{\alpha}_1 \pi^{-1}(\sum_i \mathbf{e}_i \mathbb{1}_i)\big] / \big[2(\sum_i \mathbb{1}_i)\big]$$

where $\mathbf{e}_i = \exp(-\frac{\|(\mathbf{X}_i)^T - \boldsymbol{\mu}\|_2^2}{2r^2})$.

3) Regularizations: The objective function (Eq. (1)) evaluates the mean square error of an approximator and selects the set of parameters that minimizes the mean error. While the mean is already a normalization of errors for different sizes (r) of an approximator, the objective function is still biased towards smaller sizes. Since the parameters $\boldsymbol{\alpha}$ are optimized to adjust the shape of the approximator to minimize the errors, it is easier to reduce the mean errors or even eliminate them with smaller number of $\mathbf{y}(\mathbf{X}_i)$. As an analogy, in linear regression, it is easier to fit a perfect hyperplane to a smaller number of points. We develop two regularizers to reduce the bias.

Vertical interval regularization: The bias towards smaller sizes mainly causes problems for large trees, which may have small "bumps" on top of their canopy, and this may lead to underestimation of the tree size r. One characteristic of such "bumps" is that they also tend to have a very small range of height values in the vertical direction. Denote

v_0 as $\inf\{\mathbf{y}(\mathbf{X}_i)\,|\,\forall i\,:\,\mathbb{1}((\mathbf{X}_i)^T,\boldsymbol{\mu},r)=1\}$, and v_1 as $\sup\{\mathbf{y}(\mathbf{X}_i)\,|\,\forall i:\,\mathbb{1}((\mathbf{X}_i)^T,\boldsymbol{\mu},r)=1\}$. The vertical interval is then (v_1-v_0). Since squared errors (L2 norm) are used in the objective function, here we use the squared vertical interval $(v_1-v_0)^2$ to regularize the error. If we denote f_{obj} as the objective function, the regularized form is: $f_{obj}/(v_1-v_0)^2$.

Minimum analysis window regularization: Vertical interval regularization cannot penalize the case when a small "bump" on a large tree can be perfectly fit by an approximator, because the zero error will be invariant to the ratio-based regularizer. Thus, we use a minimum analysis window size to further penalize this scenario. Denote w_{min} as the minimum size. If the local circular window formed by $\{(\mathbf{y}(\mathbf{X}_i),\mathbf{X}_i)\,|\,\forall i\,:\,\mathbb{1}((\mathbf{X}_i)^T,\boldsymbol{\mu},r)=1\}$ has a radius r smaller than w_{min}, it will be resampled to a higher resolution with size w_{min} (i.e., from a circular region inside a $(2r+1)\times(2r+1)$ window to a circular region inside a $(2w_{min}+1)\times(2w_{min}+1)$ window) using nearest-neighbor. Practically, we use the median of the input range of radii as w_{min}. The nearest-neighbor resampling method is used to reduce the chance of a small "bump" being perfectly fit by an approximator.

The two regularizers do not affect the closed form solutions for parameter $\boldsymbol{\alpha}$ in the approximators.

B. Phase 2: Deep Learning based Urban Tree Filter

The first phase finds tree-like structures using the approximators, but it can potentially include false detections of non-tree structures. In Phase 2, we remove the non-tree structures using a deep learning filter. One issue for machine learning in the tree detection problem is the unavailability of ground truth training data. Thus, rather than using a deep learning model to detect the trees directly, we only use it as a filter, which is not trained on actual ground truth data but instead on a combination of Phase 1 detections, NHMs, and available city infrastructure data in a subset of study areas.

Many cities routinely collect and update digital information about building footprints, roads and other infrastructure such as street lights and utility towers. Such data can help determine if a detection in Phase 1 is more likely to be a tree or non-tree.

CNN-based Urban Tree Filter: The deep learning framework we use for this phase is a Convolutional Neural Network (CNN) [13]. Our CNN architecture takes input image patches of size $32\times32\times1$, and has two convolutional layers (kernel size 5×5), two max-pooling layers with strides of 2, and two fully connected layers at the end. Its training data includes a set of image patches and their labels. To construct the tree filter, we first extract a local image patch from the NHM for each detection $(\boldsymbol{\mu}^*,r^*)$ in the areas where city infrastructure data is available. For our tree filtering purpose, we are only interested in a binary labeling: [1: tree, 0: non-tree]. Using the city infrastructure data, we label an image patch as a "non-tree" if the center $\boldsymbol{\mu}^*$ of its corresponding detection is within the polygons of non-tree infrastructures (e.g., buildings, roads). The training dataset we use here is not perfect ground truth data, and it may contain some noise such as incorrect labels. Thus, the goal of this trained CNN-filter is not to detect trees, but to help remove non-tree objects when city infrastructure data is not available or not complete.

TABLE I
SUMMATION UNITS IN $\boldsymbol{\alpha}$ SOLUTIONS

	$\mathbf{X}$, $\mathbb{1}$	$\mathbf{y}$, $\mathbb{1}$	$\mathbf{X}$, $\mathbf{y}$, $\mathbb{1}$	$\mathbb{1}$
$\boldsymbol{\alpha}_1$	$\sum_i \mathbf{Z}_i^2\mathbb{1}_i$, $\sum_i \mathbf{Z}_i^4\mathbb{1}_i$	$\sum_i \mathbf{y}_i\mathbb{1}_i$	$\sum_i \mathbf{Z}_i^2\mathbf{y}_i\mathbb{1}_i$	$\sum_i \mathbb{1}_i$
$\boldsymbol{\alpha}_2$	$\sum_i \mathbf{Z}_i^2\mathbb{1}_i$	$\sum_i \mathbf{y}_i\mathbb{1}_i$	-	$\sum_i \mathbb{1}_i$

C. TIMBER Acceleration

The CNN in Phase 2 is in general very efficient during prediction. For Phase 1, we propose a Core Object REduction (CORE) algorithm for acceleration.

TIMBER_Base: A direct observation is that, for each combination of size and location $(r,\boldsymbol{\mu})$ at a local maximum, we only need to check the elements (e.g., $\mathbf{X}_i$, $\mathbf{y}_i$) against the indicator function $\mathbb{1}((\mathbf{X}_i)^T,\boldsymbol{\mu},r)$ within the local square neighborhood of size $(2r+1,2r+1)$ centered at $\boldsymbol{\mu}$. Further, the optimization processes at different local maxima are independent so they can be parallelized on multiple CPU cores.

TIMBER_CORE: While $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ can be evaluated in closed form, their computations can be expensive. For simplicity, here we use the solutions for the negative quadratic approximator (Thm. 1) as an example. The strategy is the same for the Guassian approximator.

In Thm. 1, there are many summation operations needed to compute elements in $\mathbf{X}$, $\mathbf{y}$ and $\mathbb{1}$ as well as their cross products. These summations introduce most of the computations. Here we consider each summation as a summation unit. Table I lists all summation units without duplicates. The row names identify the parameter that the summation units belong to, and the column names show where the participating elements in the summations are from. The idea of the CORE algorithm is to identify the core objects in the solutions that can be shared across multiple optimization processes to reduce computation.

First, examining the values of $\mathbf{Z}_i=\|(\mathbf{X}_i)^T-\boldsymbol{\mu}\|_2$, we can find that although the values of $\mathbf{X}_i$ and $\boldsymbol{\mu}$ can all differ, their differences remain the same for a fixed radius r across all locations. For example, consider the local windows of radius r (in pixels) centered at all $\boldsymbol{\mu}$. If we compute the $\mathbf{Z}_i$ values for the pixels in all these local windows, all resulting matrices will be identical despite their different $\mathbf{X}_i$ and $\boldsymbol{\mu}$ values. Furthermore, all these local windows of a fixed size r will share the same indicator function values as well. Combining these two observations, we can conclude that the results of all summation units of $\sum_i \mathbf{Z}_i^2\mathbb{1}_i$, $\sum_i \mathbf{Z}_i^4\mathbb{1}_i$ and $\sum_i \mathbb{1}_i$ are the same for all optimizations with the same size r.

For summation units $\sum_i \mathbf{Z}_i^2\mathbf{y}_i\mathbb{1}_i$ and $\sum_i \mathbf{y}_i\mathbb{1}_i$, their values are different at each location $\boldsymbol{\mu}$ because $\mathbf{y}_i$ values can differ across locations and they are not neutralized by $\boldsymbol{\mu}$ as in $\mathbf{Z}_i$. Nevertheless, here we can see that all these summations still involve the values of the indicator function $\mathbb{1}$ and $\mathbf{Z}_i$. Since for each size r, the values of $\mathbb{1}$ and $\mathbf{Z}_i$ remain the same in all the local windows, we can keep their values in two matrices of

size $(2r+1, 2r+1)$, that is, the size of the minimum bounding square of a local window with a radius r. These matrices can then be shared across all optimizations of the same size r.

All these core objects for all candidate sizes $\{r\}$ only need to be computed once at the beginning of TIMBER; they then become inputs to the computations of $\boldsymbol{\alpha}$.

D. Computational Complexity Analysis

Here we evaluate the time complexity of the first phase of TIMBER as it is the computational bottleneck. Denote the number of local maxima and tree sizes as N and M, respectively, the maximum tree size (in pixels) as r_{max}, the window size for the flexible tree center search as w, and the number of CPU cores as C. The CORE algorithm reduces the necessary number of summations (Table I) to a constant portion ρ of all summations. While it does not change the asymptotic complexity of the optimization process, which is $O(C^{-1}NMw^2r_{max}^2)$, the constant scaling could still result in a significant amount of savings for a long execution time.

IV. Validation

We validated the solution quality of TIMBER through a case study, and confirmed the computational savings achieved by the CORE algorithm using controlled experiments.

A. Case Study

We conducted the case study in Minneapolis, US, which is a well populated and urbanized region with a well-maintained green infrastructure (e.g., trees) across its district zones.

The total area of the zones in our case study was about 6,500 acres, one third of which was used to generate the training data of the second phase of TIMBER (i.e., CNN-based urban tree filter). Transfered learning was used to facilitate the convergence. Overall, we implemented 6 candidate algorithms for tree detection in this comparison:

(1) *TIMBER$_Q$:* TIMBER framework with the approximator of negative quadratic functions. (2) *TIMBER$_G$:* TIMBER with the Gaussian approximator. (3) *YOLO:* You-Only-Look-Once (YOLO) [6] is a state-of-the-art convolutional neural network for object detection. Since we did not have actual ground truth data, YOLO was trained with the same data used by our CNN classifier in Phase 2, which was the best that we could do. (4) *Watershed-SEG:* Watershed segmentation is the most used framework for tree detections in remote sensing communities [7], [8], [9]. (5) *Spatial-SEG:* Mean-shift segmentation [14], [10] is a commonly used method for segmenting spatial datasets which considers both spatial and spectral similarities. (6) *GMM-EM:* The Gaussian-Mixture-Model (GMM) clustering algorithm with Expectation-Maximization (EM) [4] has also been used to estimate tree canopy sizes. We used height and locations as the features in GMM, and initialized its EM optimization with the local maxima in the NHM to facilitate the convergence. Since we did not know the number of clusters (trees), we used the number of local maxima as an estimate.

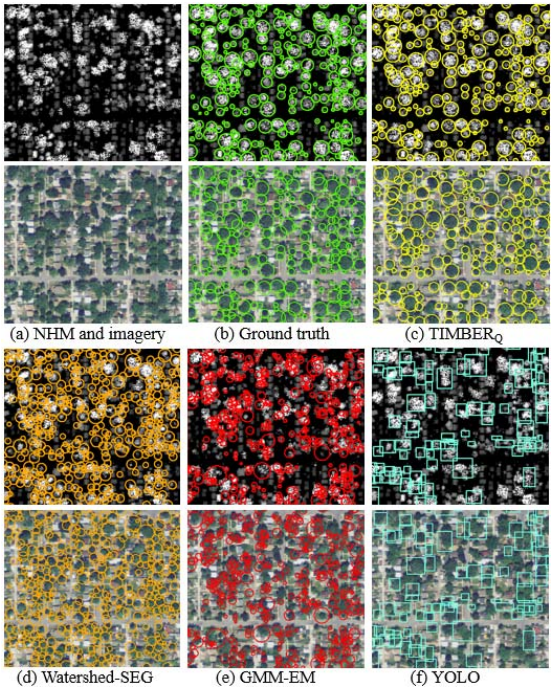

Fig. 2. Comparison of detections. The imagery was collected one year after the NHM, so several trees (removed) may only show up in the NHM. The imagery is only used for visual assistance. (best in color)

Result postprocessing: We used digital city infrastructure data (e.g., buildings, towers, roads, etc.) to filter out the non-tree detections in Watershed-SEG, Spatial-SEG and GMM-EM (otherwise much lower accuracy). This is not doable for regions without such data. In addition, the irregular tree shapes detected by the three methods were approximated to circles.

Since individual tree locations and canopy sizes are rarely collected or publicly available, we selected four areas in the test region (not seen in training), and went through a time-consuming manual inspection to generate the ground truth data for testing. We also involved spatial data experts to help improve the data collection using functionalities of spatial software (e.g., 3D profiling, visualization enhancement).

Fig. 2 visualizes an example of the data, ground truth, and representative results from four candidate methods. As we can see, TIMBER better captured the tree locations and canopy sizes in the ground truth. Since watershed segmentation is a rigid segmentation based on geometric properties, it got stuck in the local fluctuations on large tree tops, splitting a single tree into many smaller pieces. YOLO's main limitation is on detecting small objects appearing in groups, so it experienced difficulties in finding and separating out the trees.

The four test areas had different landscapes (e.g., mixtures of trees and small or large buildings). The number of trees in test Areas 1 to 4 was 1270, 1393, 972 and 1193, respectively. We evaluated the precision, recall and F1-scores for the 6 methods over the four test areas. The results are presented in Fig. 3 (a)-(c), and the F1 scores are detailed in Table II.

TABLE II
F1 SCORES OF DETECTIONS IN TEST AREAS

Methods	Area 1	Area 2	Area 3	Area 4
TIMBER$_Q$	84.1%	87.8%	90.2%	88.4%
TIMBER$_G$	71.9%	72.8%	76.6%	76.1%
YOLO	35.5%	31.5%	40.0%	31.4%
Watershed-SEG	51.9%	51.8%	52.3%	49.6%
Spatial-SEG	24.1%	23.2%	17.9%	18.5%
GMM-EM	23.6%	22.9%	22.7%	21.5%

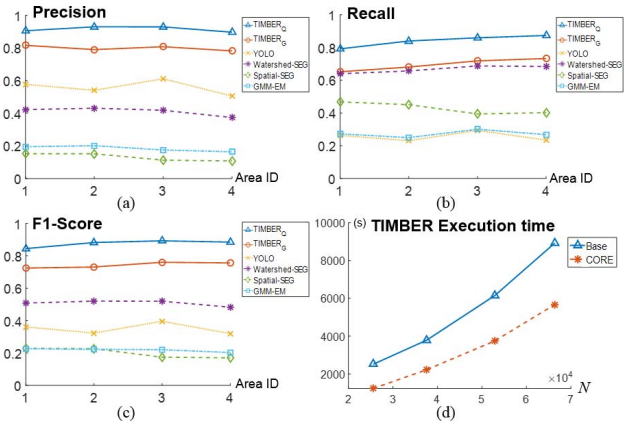

Fig. 3. Solution quality statistics and execution time.

The statistics are consistent with our visual comparison in Fig. 2. TIMBER$_Q$ achieved the highest precision (90-95%), recall (80-85%) and F1-scores (85-90%) in all four test areas. While TIMBER$_G$ also performed consistently better than related work, there was about a 10% gap with TIMBER$_Q$. This is an interesting result since it suggests that negative quadratic function is a better approximator of tree shapes compared to Gaussian. Among approaches from related work, YOLO achieved the highest precision (50-60%) but had low recall ($\sim$25%), leading to low F1 scores, while Watershed-SEG had good recall ($\sim$65%) but suffered from low precision (40-45%). This could be due to its tendency to split individual large trees into smaller pieces. Neither Spatial-SEG nor GMM-EM performed well in the test areas.

B. Computational Performance

In the complexity analysis of the optimization process (Sec. III-D), most of the parameters are related to the maximum tree size r_{max}. In real world applications, especially in urban environments, r_{max} is often fairly limited (e.g., 15, 20 meters). Thus, compared to the total number of local maxima N, r_{max} and other parameters are relatively stable (similar to constants). Thus, our analysis focused on the effect of N, which was varied by changing the size of the study area.

We evaluated the proposed algorithms on a 24-core computing node in a Linux environment. Since the results were very similar for TIMBER$_Q$ and TIMBER$_G$, here we present those of the former for illustration purposes. Fig. 3(d) shows the execution time for the baseline and CORE algorithms. As discussed in Sec. III-D, CORE does not change the asymptotic complexity but reduces the number of summation units (Table I) to a constant proportion. Through Fig. 3(d) we can see that the speedup is about 1.5X to 2X in the experiments. In the largest area studied (i.e., a 3.5$\times$2.5km^2 region corresponding to the maximum N), the CORE algorithm saved more than 20 hours of CPU time (about one hour of wall-time).

V. CONCLUSIONS AND FUTURE WORK

We proposed a two-phase TIMBER framework to generate individual tree inventories from remote sensing datasets as well as a CORE algorithm to improve computational efficiency. Through experiments, we showed that TIMBER can significantly improve accuracy compared to related work and the CORE algorithm can speed up the computation. In future work, we aim to generate a benchmark tree inventory to facilitate future research on tree species classification at a real-world large scale. In addition, we aim to design new acceleration methods to further reduce the computational time.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grants No. 1737633, 1541876, 1029711, IIS-1320580, 0940818 and IIS-1218168, the US-DOD under Grants HM0210-13-1-0005, ARPA-E under Grant No. DE-AR0000795, USDA under Grant No. 2017-51181-27222, NIH under Grant No. UL1 TR002494, KL2 TR002492 and TL1 TR002493 and the OVPR U-Spatial and Minnesota Supercomputing Institute at the University of Minnesota. We also thank Kim Koffolt for improving the paper's readability.

REFERENCES

[1] "Emerald ash borer," https://www.nrs.fs.fed.us/disturbance/invasive_species/eab/effects_impacts/cost_of_infestation/, 2018.
[2] BBC News, "Ash tree set for extinction in europe," http://www.bbc.com/news/science-environment-35876621, 2016.
[3] "Green infrastructure," www.esri.com/about-esri/greeninfrastructure.
[4] D. Reynolds, "Gaussian mixture models," *Encyclopedia of biometrics*, pp. 827–832, 2015.
[5] K. Nogueira, O. A. Penatti, and J. A. dos Santos, "Towards better exploiting convolutional neural networks for remote sensing scene classification," *Pattern Recognition*, vol. 61, pp. 539–556, 2017.
[6] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 6517–6525.
[7] M. Maltamo, J. Peuhkurinen *et al.*, "Predicting tree attributes and quality characteristics of scots pine using airborne laser scanning data," *SILVA FENNICA*, vol. 43, no. 3, 2009.
[8] H. Kaartinen, J. Hyyppä, *et al.*, "An international comparison of individual tree detection and extraction using airborne laser scanning," *Remote Sensing*, vol. 4, no. 4, pp. 950–974, 2012.
[9] L. Wallace *et al.*, "Evaluating tree detection and segmentation routines on very high resolution uav lidar data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 12, pp. 7619–7628, 2014.
[10] "Segment mean shift," http://desktop.arcgis.com/en/arcmap/latest/manage-data/raster-and-images/segment-mean-shift-function.htm, 2018.
[11] Y. Xie, G. Tang *et al.*, "Crater detection using the morphological characteristics of chang'e-1 digital elevation models," *IEEE Geoscience and Remote Sensing Letters*, vol. 10, no. 4, pp. 885–889, 2013.
[12] P. Tittmann, S. Shafii *et al.*, "Tree detection and delineation from lidar point clouds using ransac," in *Proceedings of SilviLaser*, 2011.
[13] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.
[14] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 5, pp. 603–619, 2002.