Automated Detection of Three-Dimensional Cars in Mobile Laser Scanning Point Clouds Using DBM-Hough-Forests

Yongtao Yu, Jonathan Li, *Senior Member, IEEE*, Haiyan Guan, *Member, IEEE*, and Cheng Wang, *Member, IEEE*

***Abstract*—This paper presents an automated algorithm for rapidly and effectively detecting cars directly from large-volume 3-D point clouds. Rather than using low-order descriptors, a multilayer feature generation model is created to obtain high-order feature representations for 3-D local patches through deep learning techniques. To handle cars with different levels of incompleteness caused by data acquisition ways and occlusions, a hierarchical visibility estimation model is developed to augment Hough voting. Considering scale and orientation variations in the azimuth direction, a set of multiscale Hough forests is constructed to rotationally cast votes to estimate cars' centroids. Quantitative assessments show that the proposed algorithm achieves average completeness, correctness, quality, and F_1-measure of 0.94, 0.96, 0.90, and 0.95, respectively, in detecting 3-D cars. Comparative studies also demonstrate that the proposed algorithm outperforms the other four existing algorithms in accurately and completely detecting 3-D cars from large-scale 3-D point clouds.**

***Index Terms*—Car detection, deep learning, Hough forest, mobile laser scanning (MLS), point cloud, visibility estimation.**

I. Introduction

COMPARED to object detection and recognition in remotely sensed imagery [1]–[3], object detection and recognition directly from large-scale 3-D data is still on its early stages. The limitations and challenges include the following: 1) lack of available public 3-D data corpus covering large-scale real-world scenes; 2) high computational and spatial complexities for handling large volumes of discrete and irregularly distributed 3-D point sets; and 3) lack of abundant texture information. In addition, existing methods and software still cannot meet the requirements for rapidly and automatically processing huge-volume unstructured 3-D data toward object detection and recognition.

With an effective integration of laser scanning and positioning technologies, the acquisition of 3-D real-world data over a large area can be rapidly and cost-effectively accomplished by using laser scanning systems [4]–[6]. The collected highly dense and accurate point cloud data become a promising data source for object detection and recognition applications. Therefore, studies on 3-D object detection in large-scale point cloud scenes currently become a hot and significant topic. In this paper, we focus on the detection of 3-D cars directly from 3-D mobile laser scanning (MLS) point clouds.

Cars are very common and important tools in current transportation activities. Detection and statistics of cars can provide essential information to a variety of applications, such as traffic flow monitoring, intelligent transportation, autonomous driving, etc. Existing methods for car detection are basically categorized as follows: 1) segmentation- and feature-recognition-based methods; 2) model-driven methods; and 3) machine-learning-based methods.

In [7], a marked point process based method was proposed to detect cars in crowded urban areas. In this method, a marked point process of 2-D rectangles simulated by a multiple birth-and-death algorithm [8] was modeled to describe the positions, sizes, and orientations of cars. Similarly, two-level point processes of rectangles [9] were also developed to detect cars. In [10], on-road cars were located based on the detection of slope changes on the transversal profiles of the road. A context-guided method based on the geometric model of cars was proposed in [11]. In this method, marker-controlled watershed transformation assisted by morphological reconstruction was performed to isolate cars.

In [12], an adaptive 3-D segmentation method was proposed to detect cars for motion state and velocity estimation. This method featured a capability of detecting local arbitrary modes at multiple scales. In [13], an object-based point cloud analysis method was proposed for car detection. In this method, first, segmentation-based progressive triangular irregular network densification and 3-D connected component analysis were performed to group potential car points into segments. Then, cars were detected based on area, rectangularity, and elongatedness features. In addition, Hough forest frameworks [14], [15], 3-D object matching [16], invariant parameters of polar line-segments [17], grid-cell method [18], bottom-up and top-down descriptors [19], and implicit shape models [20] were also exploited for car detection. However, existing methods lack sufficient descriptors to depict high-order features of point

Manuscript received November 6, 2014; revised September 5, 2015 and January 6, 2016; accepted February 27, 2016. This work was supported in part by the National Natural Science Foundation of China under Grant 41471379 and in part by the Priority Academic Program Development and the Collaborative Innovation Center of Atmospheric Environment and Equipment Technology.

Y. Yu is with the Fujian Key Laboratory of Sensing and Computing for Smart Cities, Xiamen University, Xiamen 361005, China, and also with the Faculty of Computer and Software Engineering, Huaiyin Institute of Technology, Huaian 223003, China (e-mail: allennessy.yu@gmail.com).

J. Li is with the Fujian Key Laboratory of Sensing and Computing for Smart Cities, School of Information Science and Engineering, Xiamen University, Xiamen 361005, China, and also with the Department of Geography and Environmental Management, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: junli@uwaterloo.ca).

H. Guan is with the College of Geography and Remote Sensing, Nanjing University of Information Science and Technology, Nanjing 210044, China.

C. Wang is with the Fujian Key Laboratory of Sensing and Computing for Smart Cities, School of Information Science and Engineering, Xiamen University, Xiamen 361005, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TGRS.2016.2537830

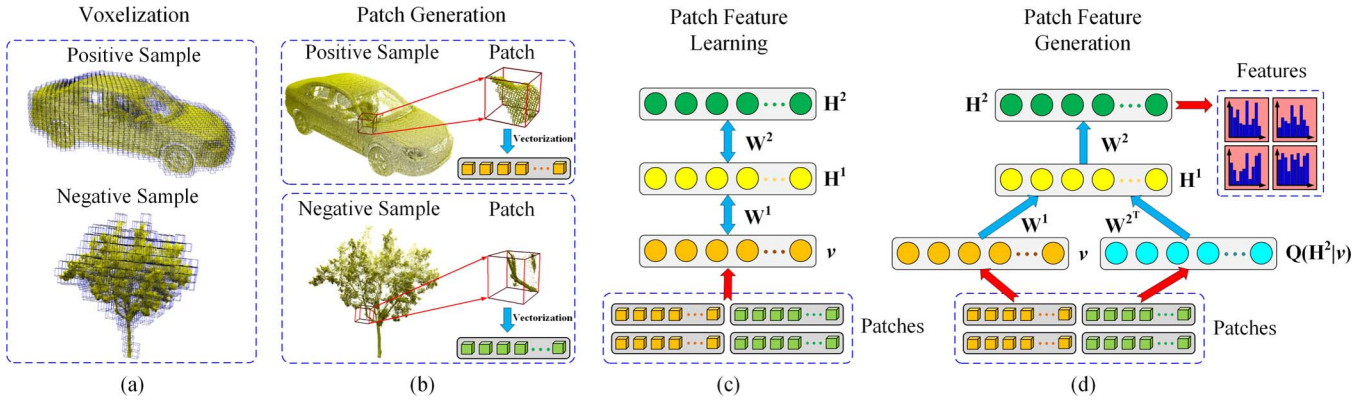


Fig. 1. (a) Voxelization of training samples. (b) Three-dimensional local patch generation. (c) Patch feature learning model. (d) Patch feature generation model.

cloud regions. In addition, these methods cannot obtain promising performance when dealing with complex urban scenes containing cars of different sizes and different levels of data incompleteness.

In this paper, we propose a rapid and automated algorithm for detecting 3-D cars from MLS point clouds. The proposed algorithm can effectively handle large-volume point clouds of complex urban scenes containing cars of different sizes and different levels of incompleteness. The proposed algorithm consists of three main models: 1) the multilayer feature generation model used for generating high-order features of 3-D local patches; 2) the multiscale Hough forest models used for handling scale variations of cars and estimating car centroids; and 3) the visibility estimation model used for handling cars with different levels of incompleteness. At the detection stage, first, ground points are removed using a voxel-based upward growing method. Then, off-ground points are voxelized to generate 3-D local patches, which are characterized by the multilayer feature generation model. Next, a set of multiscale Hough forests is applied to the high-order feature encoded 3-D local patches to estimate car centroids, and the visibility estimation model is performed to estimate the completeness of cars for augmenting Hough voting. Finally, cars are detected via a nonmaximum suppression process. The main contributions include the following: 1) a multilayer feature generation model for encoding high-order features of 3-D local patches; 2) a set of multiscale Hough forests with embedded part information for estimating car centroids; and 3) a visibility estimation model for handling cars with different levels of incompleteness.

II. Hierarchical-Deep and Hough Forest Models

In this section, we present detailed constructions of the three models used in the proposed car detection algorithm: multilayer feature generation model, Hough forest model, and visibility estimation model.

A. Multilayer Feature Generation Model

Most of the existing feature descriptors are developed to describe local low-order features of feature points [21]–[23]. However, only few studies have focused on analyzing distribution features of point cloud regions. Recently, deep learning models [24]–[26] have attracted great attention for their capability of learning hierarchical deep feature abstractions from large-volume unlabeled data. Among these models, deep Boltzmann machines (DBMs) [26] have proven to be a powerful feature generation model. A DBM is composed of a stack of restricted Boltzmann machines (RBMs) [27]. Therefore, in this paper, to obtain high-order feature representations of 3-D local patches, we use the DBM to construct a multilayer feature generation model, which has not been studied for 3-D point clouds in the literature.

First, the training samples (positive and negative samples) are partitioned into a voxel structure with a voxel resolution of w_v (e.g., 3 cm) using the octree partition strategy, as shown in Fig. 1(a). Then, a set of local patches with a size of $n \times n \times n$ voxels is generated from the voxelized training samples, as shown in Fig. 1(b). To guarantee a complete and consistent coverage of each training sample, two adjacent patches are designed to have an overlapping size of n_o voxels along the patch's edge. Finally, the generated local patches are vectorized into a binary vector of length n^3, where the value of 1 indicates the existence of points in its corresponding voxel while 0 means an empty voxel [see Fig. 1(b)].

As shown in Fig. 1(c), a two-layer DBM is designed to train the binary vectors. Denote $\boldsymbol{v} \in \{0,1\}^{n^3}$ as a binary vector associated with a 3-D local patch. Let $\mathbf{H}^1 \in \{0,1\}^{D_1}$ and $\mathbf{H}^2 \in \{0,1\}^{D_2}$ be the lower and higher layer binary hidden variables, respectively. Here, D_1 and D_2 denote the numbers of hidden units in the lower and higher hidden layers, respectively. Then, the energy of the joint configuration $\{\boldsymbol{v}, \mathbf{H}^1, \mathbf{H}^2\}$ is defined as follows [26]:

$$E(\boldsymbol{v}, \mathbf{H}^1, \mathbf{H}^2; \theta) = -\boldsymbol{v}^T \mathbf{W}^1 \mathbf{H}^1 - (\mathbf{H}^1)^T \mathbf{W}^2 \mathbf{H}^2 - \boldsymbol{b}^T \boldsymbol{v} - (\boldsymbol{a}^1)^T \mathbf{H}^1 - (\boldsymbol{a}^2)^T \mathbf{H}^2 \quad (1)$$

where $\theta = \{\mathbf{W}^1, \mathbf{W}^2, \boldsymbol{b}, \boldsymbol{a}^1, \boldsymbol{a}^2\}$ are the model parameters. $\mathbf{W}^1$ and $\mathbf{W}^2$ represent the visible-to-hidden and hidden-to-hidden symmetric interaction terms, respectively, $\boldsymbol{b}$ is the visible bias, and $\boldsymbol{a}^1$ and $\boldsymbol{a}^2$ are the hidden biases [26]. The conditional distributions over the visible and two sets of hidden variables are defined as follows:

$$p\left(h_j^1 = 1 | \boldsymbol{v}, \mathbf{H}^2\right) = \sigma\left(\sum_{i=1}^{n^3} w_{ij}^1 v_i + \sum_{m=1}^{D_2} w_{jm}^2 h_m^2 + a_j^1\right) \quad (2)$$

$$p\left(h_m^2 = 1 | \mathbf{H}^1\right) = \sigma\left(\sum_{j=1}^{D_1} w_{jm}^2 h_j^1 + a_m^2\right) \quad (3)$$

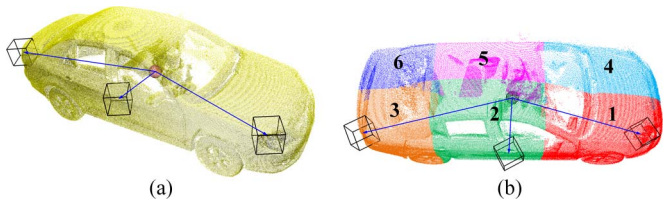


Fig. 2. Three-dimensional local patches with (a) offsets from the car's centroid and (b) car's part information.

$$p(v_i = 1|\mathbf{H}^1) = \sigma\left(\sum_{j=1}^{D_1} w_{ij}^1 h_j^1 + b_i\right) \tag{4}$$

where $\sigma(x) = 1/(1+e^{-x})$ is the logistic function.

Exact maximum likelihood learning in this DBM model is intractable. To effectively train this model, a greedy layerwise pretraining [28] is first carried out to initialize the model parameters θ. Then, an iterative joint training procedure based on variational and stochastic approximation approaches [26] is performed to fine-train the model parameters.

After the DBM model is learned, a multilayer feature generation model is constructed by replacing the stochastic activities of binary features in each hidden layer with deterministic real-valued probability estimations [see Fig. 1(d)]. For each vector $\boldsymbol{v}$, mean-field inference [28] is applied to compute an approximate posterior distribution $Q(\mathbf{H}^2|\boldsymbol{v})$. Then, the marginal $q(h_m^2 = 1|\boldsymbol{v})$ of this approximate posterior is input as an augmentation to this feature generation model. Finally, the output of this multilayer feature generation model encodes a high-order feature representation for vector $\boldsymbol{v}$ associated with a 3-D local patch

$$I = \sigma\Big(\sigma\Big(\boldsymbol{v}^T\mathbf{W}^1 + q(\mathbf{H}^2|\boldsymbol{v})^T(\mathbf{W}^2)^T + (\boldsymbol{a}^1)^T\Big)\mathbf{W}^2 + (\boldsymbol{a}^2)^T\Big) \in [0,1]^{D_2}. \tag{5}$$

B. Hough Forest Model

Developed from implicit shape model and random forests, Hough forests [29] have become a promising model for learning mappings from patch features to object centroids. They have been successfully applied to object detection [29], [30]. In this paper, to detect 3-D cars, we modify the Hough forest model to learn mappings from high-order features of 3-D local patches to possible locations of cars and car's part distributions. The embedded car's part distribution information can be further used to estimate the visibility of a car.

First, the positive and negative training samples are voxelized to generate 3-D local patches using the octree partition strategy. However, as shown in Fig. 2, for local patches generated from positive samples, offset and part information of each local patch is computed simultaneously. The offset of a local patch is defined as a direction vector starting from a car's centroid and ending at the patch's centroid [see Fig. 2(a)]. As shown in Fig. 2(b), a car sample is vertically partitioned into six parts. Thus, the part information of a local patch is defined as the index of the part that the patch's centroid belongs to. Then, the generated local patches are input to the multilayer feature generation model to generate high-order features. Finally, the obtained high-order patch features, along with their class labels, part information, and offsets, form the training data for constructing a Hough forest [see Fig. 3(a)].

For a Hough forest, each tree is constructed based on a set of high-order featured 3-D local patches $\{f_i = (I_i, c_i, p_i, d_i)\}$ [see Fig. 3(a)], where I_i denotes the high-order feature representation, c_i denotes the class label (0 means a negative patch and 1 means a positive patch), p_i denotes the part information, and d_i denotes the offset. For a negative patch, p_i is set to be 0, and d_i is set to be (0, 0, 0). As shown in Fig. 3(b), each internal node of a constructed tree represents a binary test function, which partitions the patches reaching this node into two subsets according to their feature representations and distributes them to its left and right children nodes, respectively. The binary test function is defined as follows:

$$\text{BiTest}(I) = \begin{cases} 0, & \text{if } I(e) < \tau \\ 1, & \text{otherwise} \end{cases} \tag{6}$$

where $I(e)$ denotes the eth feature channel of feature vector I and $\tau \in (0,1)$ is a real handicap value. Therefore, the patches with test values of 0 are distributed to the left child node, while the others are distributed to the right child node. For each leaf node, the information including a label proportion C_L, a part proportion list P_L, and an offset list D_L is computed and stored. $C_L \in [0,1]$ is the proportion of positive patches, $P_L \in [0,1]^6$ is a six-entry vector where each entry stores the proportion of each part in the positive patches, and $D_L = \{d_i\}$ stores a list of offsets corresponding to the positive patches.

At the training stage, each tree is constructed recursively starting from the root. During construction, each node receives a group of training patches. If the depth of the node reaches a predefined maximal depth $d_{\max}$ or the number of patches lies below a threshold $N_{\min}$, the constructed node is regarded as a leaf node. Then, the information $\{C_L, P_L, D_L\}$ is computed and stored at this leaf node. Otherwise, an internal node is constructed, and an optimal binary test function is determined to bipartition the training patches. Finally, the split two subsets are, respectively, distributed to two newly created children nodes. To suppress the class label and offset uncertainties, we adopt the same way as in [29] to design the binary test functions.

C. Visibility Estimation Model

Due to the data acquisition mode and parking orientations of cars, different cars exhibit different levels of incompleteness in the resultant data. Fig. 4 shows five common types of incompleteness of cars in the scanned scenes. Thus, to effectively estimate the completeness or visibility of a car, we develop a hierarchical visibility estimation model (see Fig. 5). This model consists of a visible layer and three hidden layers. The hidden part constructs a deep belief net [31]. The visible layer is a part proportion vector representing the distribution of the six parts of a car [see Fig. 2(b)]. This information can be obtained from the leaf nodes of the Hough forest. For a high-order feature encoded training patch, it is passed through each tree in the Hough forest. Then, the part proportion lists from all leaf nodes that the patch arrives at are aggregated to form the visible input to the visibility estimation model

$$\mathbf{P} = \sum_{t=1}^{T} C_L^t P_L^t \tag{7}$$

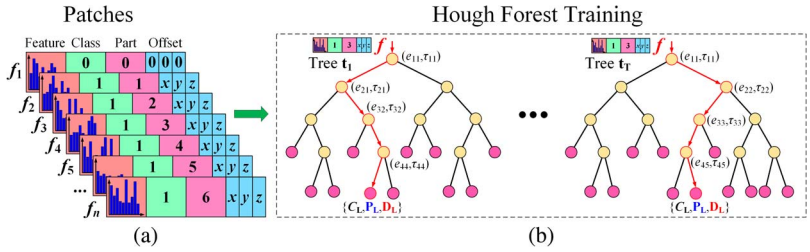


Fig. 3. (a) Training patches. (b) Hough forest training model.

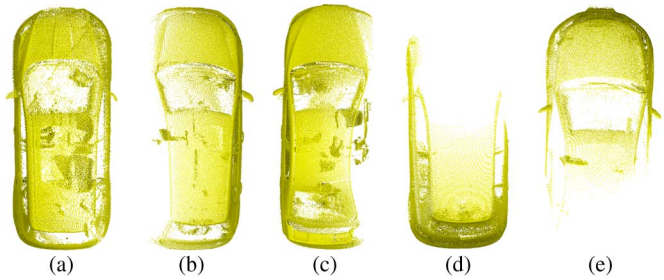


Fig. 4. Illustration of the incompleteness of a car in the real data. (a) Complete coverage and partial coverage with missing (b) left part, (c) right part, (d) front part, and (e) back part.

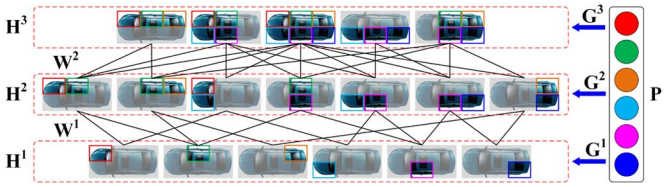


Fig. 5. Hierarchical visibility estimation model.

where C_L^t and P_L^t, respectively, are the label proportion and part proportion lists stored at the leaf node in the tth tree and T is the number of trees in the Hough forest.

In the visibility estimation model, the bottom hidden layer represents the visibilities of the six parts of a car. The top hidden layer represents the possible visibilities of a car. Given such a hierarchical model, the energy of the state $\{\mathbf{P}, \mathbf{H}^1, \mathbf{H}^2, \mathbf{H}^3\}$ is defined as follows:

$$\begin{aligned} &E(\mathbf{P}, \mathbf{H}^1, \mathbf{H}^2, \mathbf{H}^3; \phi) \\ &= -\mathbf{P}^T\mathbf{G}^1\mathbf{H}^1 - \mathbf{P}^T\mathbf{G}^2\mathbf{H}^2 - \mathbf{P}^T\mathbf{G}^3\mathbf{H}^3 - (\mathbf{H}^1)^T\mathbf{W}^1\mathbf{H}^2 \\ &\quad -(\mathbf{H}^2)^T\mathbf{W}^2\mathbf{H}^3 - \boldsymbol{b}^T\mathbf{P} - (\boldsymbol{a}^1)^T\mathbf{H}^1 - (\boldsymbol{a}^2)^T\mathbf{H}^2 - (\boldsymbol{a}^3)^T\mathbf{H}^3 \end{aligned} \tag{8}$$

where $\phi = \{\mathbf{G}^1, \mathbf{G}^2, \mathbf{G}^3, \mathbf{W}^1, \mathbf{W}^2, \boldsymbol{b}, \boldsymbol{a}^1, \boldsymbol{a}^2, \boldsymbol{a}^3\}$ are the model parameters. $\mathbf{G}^1$, $\mathbf{G}^2$, and $\mathbf{G}^3$ are the visible-to-hidden symmetric interaction terms, $\mathbf{W}^1$ and $\mathbf{W}^2$ are the hidden-to-hidden interaction terms, $\boldsymbol{b}$ is the visible bias, and $\boldsymbol{a}^1$, $\boldsymbol{a}^2$, and $\boldsymbol{a}^3$ are the hidden biases [31]. The conditional distributions over the hidden variables are given by

$$p\left(h_j^1 = 1|\mathbf{P}, \mathbf{H}^2\right) = \sigma\left(\sum_i g_{ij}^1 p_i + \sum_m w_{jm}^1 h_m^2 + a_j^1\right) \tag{9}$$

$$p\left(h_m^2 = 1|\mathbf{P}, \mathbf{H}^3\right) = \sigma\left(\sum_i g_{im}^2 p_i + \sum_k w_{mk}^2 h_k^3 + a_m^2\right) \tag{10}$$

$$p\left(h_k^3 = 1|\mathbf{P}, \mathbf{H}^2\right) = \sigma\left(\sum_i g_{ik}^3 p_i + \sum_m w_{mk}^2 h_m^2 + a_k^3\right). \tag{11}$$

This hierarchical visibility estimation model can be efficiently learned using a layerwise training strategy [31] with a stack of RBMs. Then, the output of the top hidden layer produces a probability estimation about the visibility or completeness of a car

$$\text{vis} = \max_{k=1}^{5} h_k^3. \tag{12}$$

III. Three-Dimensional Car Detection Framework

The workflow of the proposed 3-D car detection algorithm is shown in Fig. 6. First, ground points are removed based on a voxel-based upward growing method. Then, off-ground points are voxelized to generate 3-D local patches, which are characterized using the multilayer feature generation model. Next, by combining Hough voting and visibility estimation, a Hough voting space is constructed for reflecting the probabilities of the existence of cars. Finally, cars are detected via a traditional nonmaximum suppression process.

A. Ground Removal

To reduce the quantity of the data to be handled, we develop a voxel-based upward growing method, which can rapidly and effectively remove ground points from point cloud scenes. Considering ground fluctuations in a large scene, first, we vertically partition a raw point cloud into a group of data blocks with a width of w_b in the XY plane. These data blocks are processed separately to filter out ground points. Then, we organize each data block into an octree structure with a voxel size of w_x [see Fig. 7(a)]. As shown in Fig. 7(b), through octree partition, a voxel is connected with 26 neighbors. Based on such an octree partition structure, the proposed voxel-based upward growing method is carried out as follows.

For each voxel v in a data block, it grows upward to its nine neighbors located above this voxel [see Fig. 7(b)]. Then, the growing process continues from the neighbors to grow upward to their corresponding neighbors. This upward growing process stops when no more voxels can be reached. Finally, the voxel v_h with the highest elevation within the grown region is ascertained. If the elevation of v_h is smaller than a predefined ground threshold h_g, voxel v is regarded as ground; then, all of the points in voxel v are removed. Otherwise, if the elevation of v_h lies above h_g, voxel v is treated as off-ground; then, all of the points in voxel v are preserved. The voxel-based upward growing method has the following properties: 1) fit for processing large scenes with strong ground fluctuations and

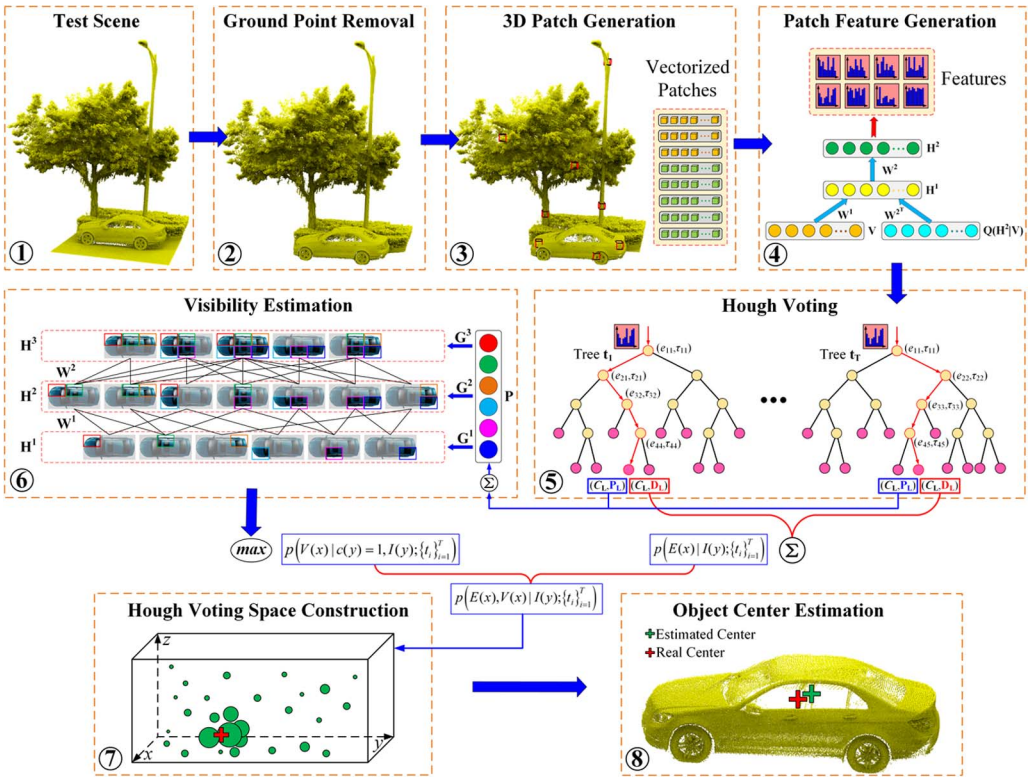


Fig. 6. Overview of the proposed 3-D car detection workflow.

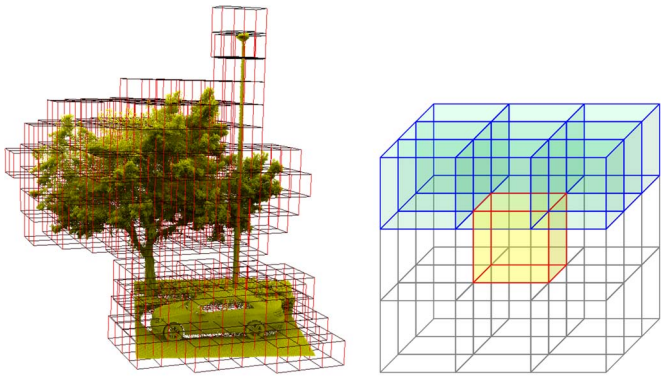


Fig. 7. (a) Octree partition structure. (b) Voxel-based upward growing scheme.

Fig. 8. (a) Raw point cloud. (b) Off-ground points after ground removal.

2) remove ground points rapidly and effectively. Fig. 8 shows an example of the obtained off-ground points after ground removal.

B. Car Detection

Before detecting cars, to improve computational efficiency, the isolated off-ground points are first grouped into separated clusters. In this paper, we adopt a Euclidean distance clustering approach which clusters discrete points based on their Euclidean distances. Theoretically, an unclustered point is included into a specific cluster if and only if its shortest distance to the points in this cluster lies below a predefined distance threshold (e.g., 0.15 m). Fig. 9(a) shows an example of the clustered off-ground points based on Euclidean distance clustering. These clusters are processed separately to estimate car centroids. To this end, first, each cluster is organized into an octree structure with a voxel size of w_v. Then, a set of local patches is generated with a size of $n \times n \times n$ voxels and an overlapping size of n_o voxels. Finally, each local patch is characterized using the multilayer feature generation model.

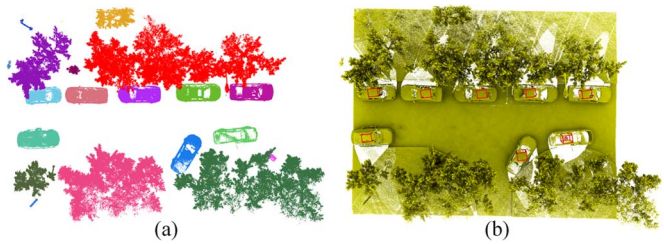


Fig. 9. (a) Clustered off-ground points. (b) Estimated 3-D car centroids.

At the detection stage, each local patch is passed to the Hough forest, where each tree receives a copy of the patch. At each internal node that the patch arrives at, the binary test function is used to determine the route that the patch follows according to the patch's feature. When the patch reaches a leaf node, the information (label proportion, part proportion list, and offset list) stored at the leaf node is used to estimate visibilities and cast votes.

Consider a patch $f(y) = (I(y), c(y), p(y), d(y))$, where $I(y)$ denotes the feature representation of the patch centered

at position y and $c(y)$, $p(y)$, and $d(y)$ are the hidden class label, part index, and offset of the patch, respectively. Denote $E(x)$ as the random event corresponding to the existence of a car centered at position x. Denote $V(x)$ as the random event corresponding to the visibility or completeness of a car centered at position x. By our definition, the existence of a car centered at position x inevitably indicates $c(y) = 1$. Then, the joint probability $p(E(x), V(x)|I(y))$ that the feature representation $I(y)$ predicts a car centered at position x and with a good completeness is expressed as

$$\begin{aligned}
&p\left(E(x), V(x)|I(y)\right)\\
&= p(E(x), V(x), c(y) = 1|I(y))\\
&= p(E(x)|V(x), c(y) = 1, I(y))\, p\left(V(x), c(y) = 1|I(y)\right)\\
&= p(E(x)|c(y)=1, I(y))\, p(V(x)|c(y)=1, I(y))\, p(c(y)=1|I(y))\\
&= p(E(x)|c(y)=1, I(y))\, p(c(y)=1|I(y))\, p(V(x)|c(y)=1, I(y))\\
&= E_{I(y)}(x) V_{I(y)}(x) \qquad (13)
\end{aligned}$$

where

$$E_{I(y)}(x) = p\left(E(x)|c(y) = 1, I(y)\right) p\left(c(y) = 1|I(y)\right) \qquad (14)$$

$$V_{I(y)}(x) = p\left(V(x)|c(y) = 1, I(y)\right). \qquad (15)$$

$E_{I(y)}(x)$ describes the probability of the existence of a car centered at position x given the patch feature $I(y)$, and $V_{I(y)}(x)$ depicts the probability of the visibility of a car centered at position x given a positive patch feature $I(y)$. $E_{I(y)}(x)$ can be computed based on the label proportion and offset list information stored at the leaf nodes that patch $f(y)$ reaches in all trees in the Hough forest. For a single tree $\boldsymbol{t}$, the probability estimation is given by

$$\begin{aligned}
E_{I(y)}(x; \boldsymbol{t}) &= p\left(d(y) = y - x|c(y) = 1, I(y)\right) p(c(y) = 1|I(y))\\
&= \frac{C_L}{|D_L|} \sum_{d \in D_L} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\|(y - x) - d\|_2^2}{2\sigma^2}\right) \qquad (16)
\end{aligned}$$

where σ^2 is the variance of the Gaussian Parzen window. For the entire Hough forest $\{\boldsymbol{t}_i\}_{i=1}^T$, we average the probabilities from all trees to form a forest-based probability estimation

$$E_{I(y)}\left(x; \{\boldsymbol{t}_i\}_{i=1}^T\right) = \frac{1}{T} \sum_{i=1}^{T} E_{I(y)}(x; \boldsymbol{t}_i). \qquad (17)$$

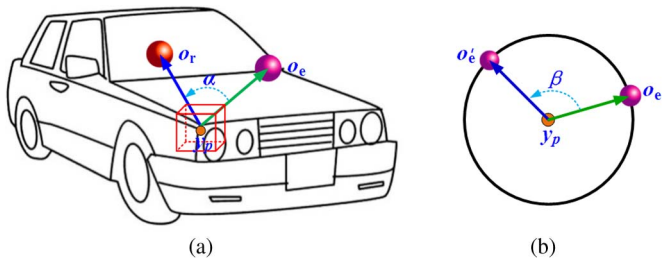
Furthermore, $V_{I(y)}(x)$ can be estimated by the visibility estimation model. The part proportion lists stored at the leaf nodes that patch $f(y)$ arrives at are weighted aggregated by the corresponding label proportions to create the input to the visibility estimation model

$$\mathbf{P}_{I(y)} = \sum_{t=1}^{T} C_L^t P_L^t. \qquad (18)$$

Then, the output of this model forms a hierarchical estimation for $V_{I(y)}(x)$

$$V_{I(y)}\left(x; \{\boldsymbol{t}_i\}_{i=1}^T\right) = \max_{k=1}^{5} \left(h_k^3; \mathbf{P}_{I(y)}\right). \qquad (19)$$

Integrating (17) and (19), we obtain the probabilistic vote cast by a single patch $f(y)$ about the existence and visibility of a car in its vicinity. Then, by aggregating the votes cast from different patches, we construct a 3-D Hough voting space



(a) (b)

Fig. 10. Illustration of (a) an incorrect voting; o_r is the real centroid, and o_e is the estimated wrong centroid. (b) Rotational voting strategy.

$HS(x)$, where each location x contains the accumulated votes about the existence and visibility of a car at this location

$$HS(x) = \sum_{f(y)} E_{I(y)}\left(x; \{\boldsymbol{t}_i\}_{i=1}^T\right) \cdot V_{I(y)}\left(x; \{\boldsymbol{t}_i\}_{i=1}^T\right). \qquad (20)$$

Finally, based on the constructed Hough voting space, cars' centroids are estimated by ascertaining the locations with local maxima through a nonmaximum suppression process.

Rotational Voting: In real scenes, cars are usually parked with various orientations. Thus, to effectively detect cars, the votes cast by a local patch should correctly estimate a car's centroid in a rotation-invariant manner. However, as shown in Fig. 10(a), if the offset stored at a leaf node is directly used to cast votes, a wrong centroid might be estimated for a rotated car. To handle orientation variations, we develop a rotational voting strategy. As shown in Fig. 10(b), the offset is rotated a full circle to proceed Hough voting. Then, (16) becomes

$$E_{I(y)}(x; \boldsymbol{t}) = \int_{\beta=0}^{2\pi} \frac{C_L}{|D_L|} \sum_{d \in D_L} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\|(y - x) - R(\beta) \cdot d\|_2^2}{2\sigma^2}\right) \qquad (21)$$

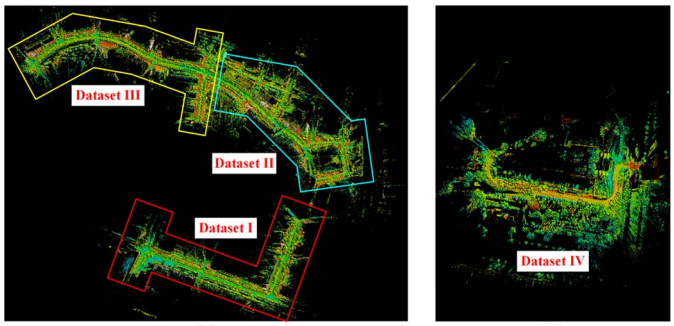
where

$$R(\beta) = \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \qquad (22)$$

Multiscale Hough Forests: To handle scale variations of cars in real scenes, we construct a set of multiscale Hough forests at the detection stage. To this end, first, we predefine a set of scale factors $\{s_1, s_2, \ldots, s_M\}$, where M is the number of scale factors. Each Hough forest is dominated by a scale factor. Thus, we construct M Hough forests, which provide different voting scales. Then, the offset in each Hough forest is scaled by its corresponding scale factor to rotationally cast votes. Therefore, for a Hough forest with a scale factor s, (21) becomes

$$\begin{aligned}
E_{I(y)}(x; \boldsymbol{t}, s) = &\int_{\beta=0}^{2\pi} \frac{C_L}{|D_L|} \sum_{d \in D_L} \frac{1}{2\pi\sigma^2}\\
&\times \exp\left(-\frac{\|(y - x) - s \cdot R(\beta) \cdot d\|_2^2}{2\sigma^2}\right). \qquad (23)
\end{aligned}$$

Such multiscale Hough forests result in a set of multiscale Hough voting spaces $\{HS(x; s_1), HS(x; s_2), \ldots, HS(x; s_M)\}$. These Hough voting spaces are processed separately to estimate cars' centroids. Fig. 9(b) shows the estimated 3-D cars' centroids.

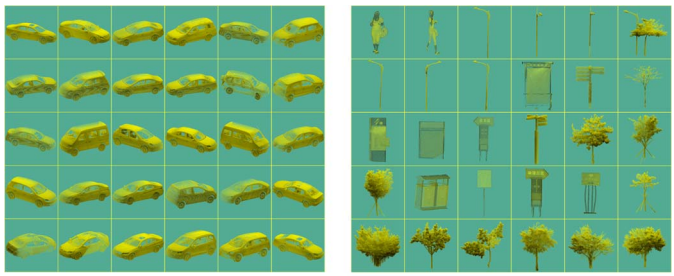


(a) (b)

Fig. 11. Illustration of the 3-D point cloud data collected in (a) Software Park Phase II and (b) Xiamen University Haiyun campus.

TABLE I
DESCRIPTION OF THE FOUR SELECTED POINT CLOUD DATA SETS

Dataset	Points	Area (m^2)	Length (m)
I	110,737,020	53,451	776
II	306,109,823	83,248	1263
III	229,967,171	51,451	845
IV	69,955,307	10,591	221



(a) (b)

Fig. 12. (a) Positive training samples. (b) Negative training samples.

IV. RESULTS AND DISCUSSION

A. Point Cloud Data Sets

The 3-D point cloud data used in this study were acquired by a RIEGL VMX-450 MLS system in Xiamen, China. Fig. 11 presents the point cloud data collected in Software Park Phase II and Xiamen University Haiyun campus, respectively. We partitioned the point cloud in Software Park Phase II into three separated data sets [see Fig. 11(a)]. The entire point cloud in Haiyun campus was treated as a single data set [see Fig. 11(b)]. A detailed description of these four data sets is listed in Table I.

From the collected point clouds, we selected a group of training samples for constructing the multilayer feature generation model, Hough forests, and hierarchical visibility estimation model, respectively. Fig. 12 presents a subset of the positive and negative samples used in this study. At the training stage, to augment the training samples to cover different orientations, each training sample was rotated eight times to generate eight samples with equal angle intervals along its central axis in the horizontal plane.

B. Parameter Sensitivity Analysis

At both training and detection stages, the configuration of the following three parameters has a significant impact on the performance of the proposed algorithm: size of the local patch (n), overlapping size (n_o) between two adjacent patches, and dimension of the high-order patch feature (D_2). To determine the optimal configurations for these three parameters, we conducted a set of experiments to examine the performance of each parameter configuration on the detection of 3-D cars. The testing results were presented and analyzed using receiver operating characteristic (ROC) curves (see Fig. 13). As shown in Fig. 13(a), when the patch size increases, the performance improves accordingly. This is because a small-sized local patch lacks distinctive representations, while a large-sized local patch is more powerful to exhibit salient informative features. However, when the patch size exceeds 10 voxels, the performance just improves slightly. In addition, a large-sized patch can bring computational burdens to train the DBM model. Thus, considering both computational complexity and performance, the patch size was set at 10 voxels. As shown in Fig. 13(b), the performance improves as the overlapping size increases. However, when the overlapping size exceeds 3 voxels, the performance almost stays unchanged. When the overlapping size increases, the number of generated patches increases dramatically at both training and detection stages. This will increase the training time and slow down the detection speed. Therefore, to obtain a good performance and a low time complexity, we set the overlapping size at 3 voxels. As shown in Fig. 13(c), the feature dimension of a local patch has a slight influence on the detection performance. This is because we adopted a deep learning model to generate a high-order feature representation for each local patch. The high-order feature representation is actually a combination and high-level abstraction of a set of low-order features; therefore, it is more powerful and distinctive. In our implementation, we set the feature dimension at 50.

C. Parameter Sensitivity Analysis for Ground Removal

In the proposed voxel-based upward growing method, the block size w_b and voxel size w_x determine the performance of the filtering result and time cost on ground removal. In this section, we tested a group of parameter configurations to evaluate their impact on ground removal performance. Two test scenes were selected in this experiment. Test scene one (a 1-D road segment with dense vegetation) contains about 6.8 million points and has a road length of about 49 m [see Fig. 14(a)]. Test scene two (a 2-D road segment with a median) contains about 12.1 million points and has a road distance of about 94 m [see Fig. 14(b)]. The ground removal time for each test scene was also recorded and illustrated in Fig. 14(c) and (d), respectively. As reflected by the testing results, with a fixed voxel size, the ground removal time increases as the block size increases, while with a fixed block size, the ground removal time decreases as the voxel size increases. Thus, to rapidly remove ground points, a relatively small block size and a relatively big voxel size should be selected. However, a big voxel might decrease the ground removal accuracy. In conclusion, considering both processing time and accuracy, the parameter configurations of $w_b = 3$ m and $w_x = 5$ cm are the optimal choices, which can provide good accuracy and low processing time (1.71 s for test scene one and 3.26 s for test scene two) in ground removal. Fig. 14(e) and (f) presents the ground removal results on these two test scenes.

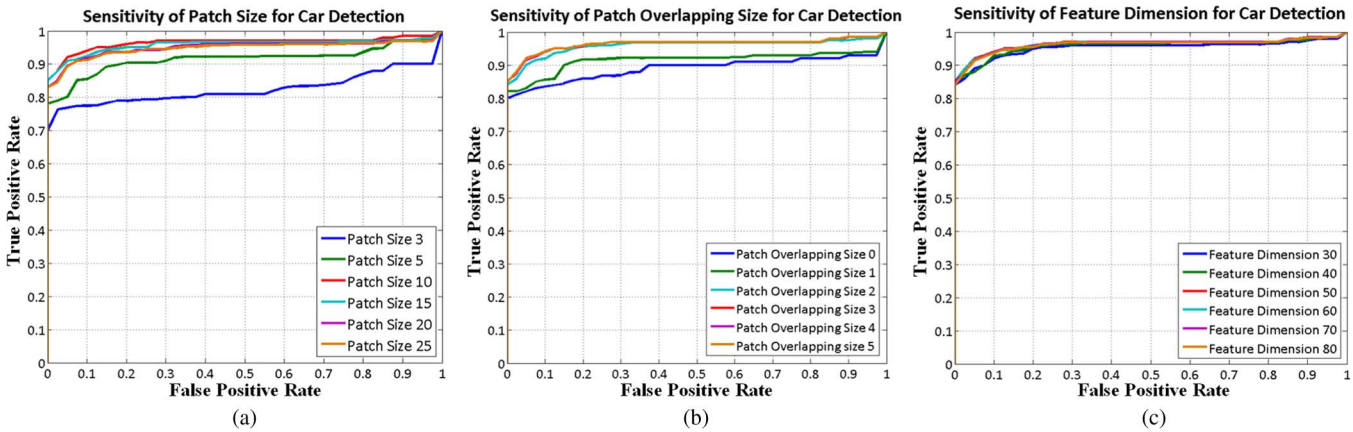


Fig. 13. Illustration of ROC curves for different parameter configurations: (a) patch size, (b) patch overlapping size, and (c) patch feature dimension.

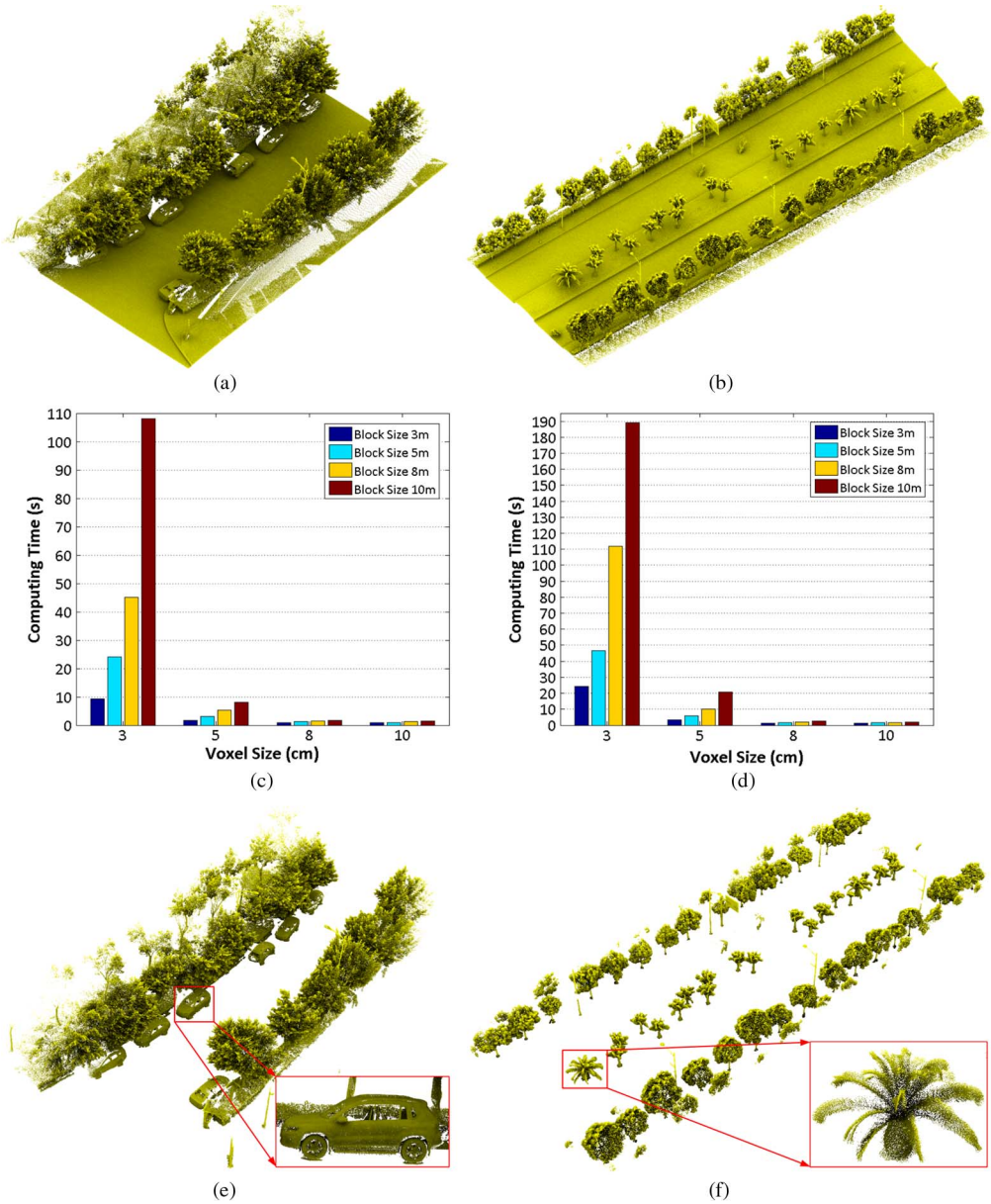


Fig. 14. (a) Test scene one, (b) test scene two, (c) ground removal time of different parameter configurations on test scene one, (d) ground removal time of different parameter configurations on test scene two, (e) ground removal result of test scene one, and (f) ground removal result of test scene two.

TABLE II
PARAMETER DESCRIPTIONS AND SETTINGS IN THE PROPOSED ALGORITHM

Notation	Description	Setting	Location
w_v	Voxel size for 3-D local patch generation	3 cm	Section II-A
$n \times n \times n$	3-D local patch size	10×10×10	Section II-A
n_o	Overlapping size of two adjacent 3-D local patches	3	Section II-A
D_1	Number of hidden units in the lower hidden layer of the DBM	500	Section II-A
D_2	Dimension of the high-order patch feature	50	Section II-A
d_{max}	Maximal depth of a tree in the Hough forest	16	Section II-B
N_{min}	Minimum number of patches in a node of a tree	20	Section II-B
T	Number of trees in the Hough forest	15	Section II-C
w_b	Block size for ground point removal	3 m	Section III-A
w_x	Voxel size for ground point removal	5 cm	Section III-A
h_g	Ground elevation threshold for ground point removal	0.4 m	Section III-A
d_c	Distance threshold for clustering off-ground points	0.15 m	Section III-B
σ^2	Variance of the Gaussian Parzen window	16	Section III-B

TABLE III
GROUND TRUTH, CAR DETECTION RESULTS, AND QUANTITATIVE EVALUATION RESULTS

Dataset	Ground Truth	Detection Result		Quantitative Evaluation			
	Car	Car	False Positive	Completeness	Correctness	Quality	F_1-Measure
I	189	178	8	0.94	0.96	0.90	0.95
II	384	369	11	0.96	0.97	0.93	0.96
III	207	195	13	0.94	0.94	0.89	0.94
IV	72	67	3	0.93	0.96	0.89	0.94
Average	-	-	-	0.94	0.96	0.90	0.95

D. Car Detection

To evaluate the performance of the proposed 3-D car detection algorithm, we applied it to the four selected data sets. After parameter sensitivity analysis, the optimal parameter settings used in the proposed algorithm are detailed in Table II. Considering scale variations of cars in these data sets, a set of multiscale Hough forests with scale factors of {0.8, 0.9, 1.0, 1.1, 1.2} was constructed. The 3-D car detection results along with the ground truth are detailed in Table III. As reflected in Table III, most of the cars were correctly detected, and the number of false positives was very low in each data set. Fig. 15 shows parts of the detection results in these four data sets. To present a good view of the detected cars, the dense vegetation over the cars was cut off in Fig. 15(b), (d), (f), and (h). As shown in Fig. 15(b) and (f), some cars failed to be detected. This is because some moving cars were distorted seriously in the collected point clouds due to the Doppler effect. In addition, caused by the occlusions of the dense vegetation, some cars parked behind trees were scanned with very bad incompleteness. Therefore, such cars were undetected using the proposed algorithm. Moreover, as shown in Fig. 15(h), two objects were falsely detected as cars due to their high similarities to cars. On the whole, the proposed algorithm obtained very good performance in detecting 3-D cars directly from large-volume 3-D point cloud data.

To quantitatively evaluate the accuracy and correctness of the 3-D car detection results, we adopted the following four measures: *completeness*, *correctness*, *quality*, and F_1-*measure* [32], [33]. *Completeness* (cpt) depicts the proportion of true positives in the ground truth, *correctness* (crt) describes the proportion of true positives in the detected components, and *quality* (qlt) and F_1-*measure* (fmr) provide two overall performance measures. They are defined as follows: $cpt = TP/(TP + FN)$, $crt = TP/(TP + FP)$, $qlt = TP/(TP + FP + FN)$, and $fmr = 2 \times cpt \times crt/(cpt + crt)$, where TP, FN, and FP are the numbers of true positives, false negatives, and false positives, respectively. The quantitative evaluation results are detailed in Table III. The proposed algorithm achieved average completeness, correctness, quality, and F_1-measure of 0.94, 0.96, 0.90, and 0.95, respectively, in detecting cars on these four data sets. Thus, the proposed algorithm is feasible and promising for detecting cars directly from 3-D point clouds.

To further demonstrate the performance and effectiveness of the proposed algorithm in detecting 3-D cars with scale variations and different levels of incompleteness, we conducted a set of comparative experiments using the point cloud data shown in Fig. 16. This point cloud data contain cars of varying scales and different levels of incompleteness. We, respectively, applied the proposed algorithm, the proposed algorithm without multiscale voting, the proposed algorithm without visibility estimation, and the proposed algorithm without multiscale voting and visibility estimation to the selected point cloud data to detect cars. Fig. 16 shows the detected cars using different algorithms. As reflected by the detection results, without multiscale voting and visibility estimation, the cars

Fig. 15. Parts of the point cloud data and the corresponding 3-D car detection results from (a) and (b) data set I, (c) and (d) data set II, (e) and (f) data set III, and (g) and (h) data set IV.

of big scales or bad incompleteness failed to be detected [see Figs. 16(b)–(d)], while such cars were correctly detected by using the proposed algorithm [see Fig. 16(a)]. Therefore, benefiting from multiscale voting and visibility estimation, the proposed algorithm obtains a promising performance in detecting 3-D cars of different scales and different levels of incompleteness.

The proposed algorithm was implemented using C++ and executed on an HP Z820 8-core-16-thread workstation. The computing time in each processing step was recorded for evaluating computational performance (see Table IV). In practice, first, each data set was vertically divided into a group of data blocks with a block size of about 50 m. Then, all of the data blocks were distributed to a multithread computing environment with 16 parallel threads. Such a parallel computing strategy can dramatically improve the computational performance and reduce the time complexity of the proposed algorithm. As shown in Table IV, all of these four large-scale point cloud data sets can be rapidly processed in a short time. For example, for the largest data set (data set II), the total computing time was only about 1926 s (about 32 min). In conclusion, the proposed algorithm is feasible and promising in rapidly handling

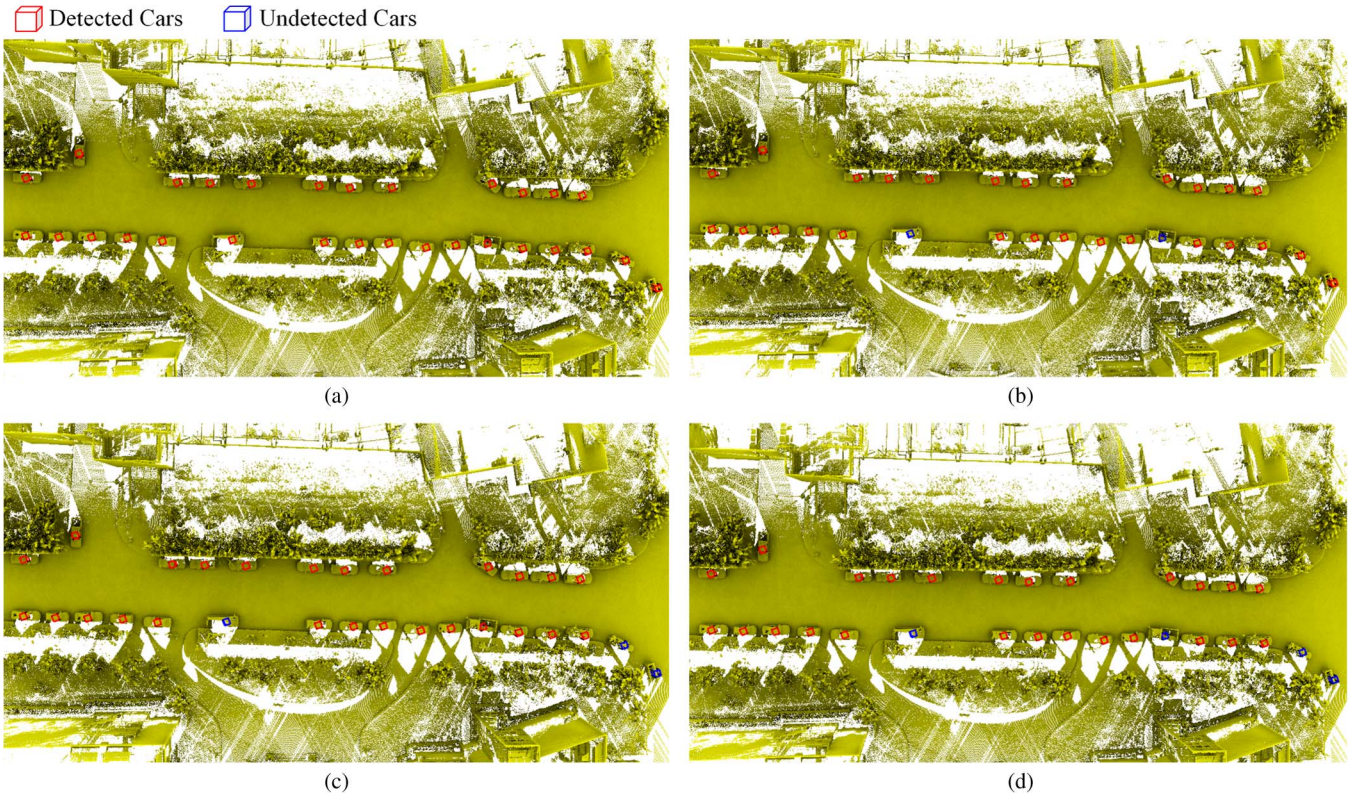


Fig. 16. Illustrations of detected 3-D cars using (a) the proposed algorithm, (b) the proposed algorithm without multiscale voting, (c) the proposed algorithm without visibility estimation, and (d) the proposed algorithm without multiscale voting and visibility estimation.

TABLE IV
COMPUTING TIME ON THE FOUR POINT CLOUD DATA SETS (UNIT: SECOND)

Dataset	Ground Removal	Clustering	Detection	Total
I	58	227	870	1155
II	94	416	1416	1926
III	63	260	948	1271
IV	17	68	247	332

large-volume 3-D point cloud data for automated detection of 3-D cars.

E. Comparative Study

A comparative study was also conducted to further compare the proposed algorithm with the following four existing 3-D car detection algorithms: Hough forest method (HF) [14], supervoxel-neighborhood-based Hough forest method (SHF) [15], bottom-up and top-down descriptors (BTD) [19], and implicit shape model based method (ISM) [20]. The 3-D car detection results along with their quantitative evaluations by using different algorithms are detailed in Table V. As reflected in Table V, the ISM method achieved a better performance than the other three methods. The HF and SHF methods utilized low-order features to model local patches. In these methods, orientation variations were considered and solved via a circular voting strategy (HF method) and a local reference frame strategy (SHF method). However, scale variations and different levels of incompleteness of cars were not handled. Thus, some cars of big sizes or with bad incompleteness were undetected by using these two methods. The BTD method performed well on the cars with good completeness. However, it also failed to detect the cars scanned with serious incompleteness. Comparatively, the ISM method obtained a similar performance to our proposed algorithm. Therefore, benefiting from the use of high-order features to characterize local patches and the considerations of scale variations and different levels of incompleteness, the proposed algorithm outperformed the other four algorithms in correctly and completely detecting 3-D cars from 3-D point clouds.

V. CONCLUSION

In this paper, we have proposed an automated algorithm for rapidly and effectively detecting 3-D cars directly from large-volume 3-D point cloud data. To obtain high-order feature representations of 3-D local patches, a multilayer feature generation model has been constructed based on a two-layer DBM. To handle cars with different levels of incompleteness, a hierarchical visibility estimation model has been created to augment the probabilities of the existence of cars being present at specific locations. Considering scale and orientation variations of cars, a set of multiscale Hough forests has been constructed to rotationally cast votes to estimate cars' centroids. Quantitative evaluations on four selected point cloud data sets have shown that the proposed algorithm achieves average completeness, correctness, quality, and F_1-measure of 0.94,

TABLE V
CAR DETECTION PERFORMANCE OBTAINED BY DIFFERENT METHODS

Method	Dataset	Detection Result		Quantitative Evaluation			
		Car	False Positive	Completeness	Correctness	Quality	F_1-Measure
HF [14]	I	164	20	0.87	0.89	0.78	0.88
	II	347	23	0.90	0.94	0.85	0.92
	III	181	27	0.87	0.87	0.77	0.87
	IV	59	12	0.82	0.83	0.70	0.82
	Average			0.87	0.88	0.78	0.87
SHF [15]	I	166	18	0.88	0.90	0.80	0.89
	II	350	17	0.91	0.95	0.87	0.93
	III	182	20	0.88	0.90	0.80	0.89
	IV	61	9	0.85	0.87	0.75	0.86
	Average			0.88	0.91	0.81	0.89
BTD [19]	I	168	16	0.89	0.91	0.82	0.90
	II	351	17	0.91	0.95	0.88	0.93
	III	183	19	0.88	0.91	0.81	0.89
	IV	61	8	0.85	0.88	0.76	0.86
	Average			0.88	0.91	0.82	0.90
ISM [20]	I	175	7	0.93	0.96	0.89	0.94
	II	366	13	0.95	0.97	0.92	0.96
	III	189	14	0.91	0.93	0.86	0.92
	IV	68	4	0.94	0.94	0.89	0.94
	Average			0.93	0.95	0.89	0.94

0.96, 0.90, and 0.95, respectively. By adopting a multithread computing strategy, the proposed algorithm can rapidly process huge-volume 3-D point cloud data toward 3-D car detection. Comparative studies have also demonstrated that the proposed algorithm outperforms the other four algorithms in accurately and completely detecting 3-D cars of different scales and different levels of incompleteness. In conclusion, the proposed algorithm is feasible and achieves promising performance and accuracy in 3-D car detection from large-scale 3-D point clouds.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments.

REFERENCES

[1] X. Bai, H. Zhang, and J. Zhou, "VHR object detection based on structural feature extraction and query expansion," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 10, pp. 6508–6520, Oct. 2014.

[2] Z. Chen *et al.*, "Vehicle detection in high-resolution aerial images via sparse representation and superpixels," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 1, pp. 103–116, Jan. 2016.

[3] J. Tang, C. Deng, G. Huang, and B. Zhao, "Compressed-domain ship detection on spaceborne optical image using deep neural network and extreme learning machine," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 3, pp. 1174–1185, Mar. 2015.

[4] K. Williams, M. J. Olsen, G. V. Roe, and C. Glennie, "Synthesis of transportation applications of mobile LiDAR," *Remote Sens.*, vol. 5, no. 9, pp. 4652–4692, Sep. 2013.

[5] B. Yang and Y. Zang, "Automated registration of dense terrestrial laser-scanning point clouds using curves," *ISPRS J. Photogramm. Remote Sens.*, vol. 95, pp. 109–121, Sep. 2014.

[6] W. Y. Yan and A. Shaker, "Radiometric correction and normalization of airborne LiDAR intensity data for improving land-cover classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 12, pp. 7658–7673, Dec. 2014.

[7] A. Börcs and C. Benedek, "A marked point process model for vehicle detection in aerial LiDAR point clouds," in *Proc. ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, Melbourne, VIC, Australia, 2012, vol. 1–3, pp. 93–98.

[8] X. Descombes, R. Minlos, and E. Zhizhina, "Object extraction using a stochastic birth-and-death dynamics in continuum," *J. Math. Imag. Vis.*, vol. 33, no. 3, pp. 347–359, Mar. 2009.

[9] A. Börcs and C. Benedek, "Extraction of vehicle groups in airborne LiDAR point clouds with two-level point processes," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 3, pp. 1475–1489, Mar. 2015.

[10] M. Varela-González, H. González-Jorge, B. Riveiro, and P. Arias, "Automatic filtering of vehicles from mobile LiDAR datasets," *Measurement*, vol. 53, pp. 215–223, Jul. 2014.

[11] W. Yao, S. Hinz, and U. Stilla, "Automatic vehicle extraction from airborne LiDAR data of urban areas aided by geodesic morphology," *Pattern Recognit. Lett.*, vol. 31, no. 10, pp. 1100–1108, Jul. 2010.

[12] W. Yao, S. Hinz, and U. Stilla, "Extraction and motion estimation of vehicles in single-pass airborne LiDAR data towards urban traffic analysis," *ISPRS J. Photogramm. Remote Sens.*, vol. 66, no. 3, pp. 260–271, May 2011.

[13] J. Zhang, M. Duan, Q. Yan, and X. Lin, "Automated vehicle extraction from airborne LiDAR data using an object-based point cloud analysis method," *Remote Sens.*, vol. 6, no. 9, pp. 8405–8423, Sep. 2014.

[14] H. Wang *et al.*, "Object detection in terrestrial laser scanning point clouds based on Hough forest," *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 10, pp. 1807–1811, Oct. 2014.

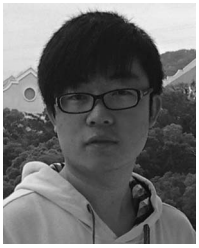
[15] H. Wang *et al.*, "3-D point cloud object detection based on supervoxel neighborhood with Hough forest framework," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 4, pp. 1570–1581, Apr. 2015.

[16] Y. Yu, J. Li, H. Guan, F. Jia, and C. Wang, "Three-dimensional object matching in mobile laser scanning point clouds," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 3, pp. 492–496, Mar. 2015.

[17] B. Fortin, R. Lherbier, and J. C. Noyer, "Feature extraction in scanning laser range data using invariant parameters: Application to vehicle detection," *IEEE Trans. Veh. Technol.*, vol. 61, no. 9, pp. 3838–3850, Nov. 2012.

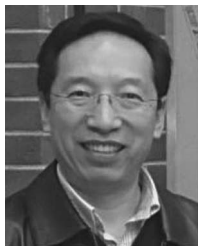
[18] W. Yao and U. Stilla, "Comparison of two methods for vehicle extraction from airborne LiDAR data toward motion analysis," *IEEE Geosci. Remote Sens. Lett.*, vol. 8, no. 4, pp. 607–611, Jul. 2011.

[19] A. Patterson, IV, P. Mordohai, and K. Daniilidis, "Object detection from large-scale 3-D datasets using bottom-up and top-down descriptors," in *Proc. Eur. Conf. Comput. Vis.*, Marseille, France, 2008, vol. 5305, pp. 553–566.
[20] A. Velizhev, R. Shapovalov, and K. Schindler, "Implicit shape models for object detection in 3D point clouds," in *Proc. ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, Melbourne, VIC, Australia, 2012, vol. 1–3, pp. 179–184.
[21] M. Körtgen, G. J. Park, M. Novotni, and R. Klein, "3D shape matching with 3D shape contexts," in *Proc. Central Eur. Semin. Comput. Graph.*, Budmerice, Slovakia, 2003, pp. 5–7.
[22] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nice, France, 2008, pp. 3384–3391.
[23] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Proc. IEEE Int. Conf. Robot. Autom.*, Kobe, Japan, 2009, pp. 3212–3217.
[24] G. Carneiro and J. C. Nascimento, "Combining multiple dynamic models and deep learning architectures for tracking the left ventricle endocardium in ultrasound data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2592–2607, Nov. 2013.
[25] B. Chen *et al.*, "Deep learning with hierarchical convolutional factor analysis," *IEEE Trans. Patten Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1887–1901, Aug. 2013.
[26] R. Salakhutdinov, J. B. Tenenbaum, and A. Torralba, "Learning with hierarchical-deep models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1958–1971, Aug. 2013.
[27] T. Tieleman, "Training restricted Boltzmann machines using approximations to the likelihood gradient," in *Proc. Int. Conf. Mach. Learn.*, Helsinki, Finland, 2008, pp. 1064–1071.
[28] R. Salakhutdinov and G. Hinton, "An efficient learning procedure for deep Boltzmann machines," *Neural Comput.*, vol. 24, no. 8, pp. 1967–2006, Aug. 2012.
[29] J. Gall, A. Yao, N. Razavi, L. V. Gool, and V. Lempitsky, "Hough forests for object detection, tracking, and action recognition," *IEEE Trans. Patten Anal. Mach. Intell.*, vol. 33, no. 11, pp. 2188–2202, Nov. 2011.
[30] Z. Lei, T. Fang, H. Huo, and D. Li, "Rotation-invariant object detection of remotely sensed images based on texton forest and Hough voting," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 4, pp. 1206–1217, Apr. 2012.
[31] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.
[32] Y. Yu, J. Li, H. Guan, C. Wang, and J. Yu, "Semiautomated extraction of street light poles from mobile LiDAR point-clouds," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 3, pp. 1374–1386, Mar. 2015.
[33] Y. Yu, J. Li, H. Guan, and C. Wang, "Automated extraction of urban road facilities using mobile laser scanning data," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 2167–2181, Aug. 2015.



Yongtao Yu received the Ph.D. degree in computer science and technology from Xiamen University, Xiamen, China, in 2015.

He is currently an Assistant Professor with the Faculty of Computer and Software Engineering, Huaiyin Institute of Technology, Huaian, China. He is the coauthor of over 20 research papers published in refereed journals and proceedings. His research interests include pattern recognition, computer vision, machine learning, intelligent interpretation of 3-D point clouds, and remotely sensed imagery.



Jonathan Li (M'00–SM'11) received the Ph.D. degree in geomatics engineering from the University of Cape Town, Cape Town, South Africa, in 2000.

He is currently a Professor with the Department of Geography and Environmental Management, University of Waterloo, Waterloo, ON, Canada, and with the Fujian Key Laboratory of Sensing and Computing for Smart Cities, School of Information Science and Engineering, Xiamen University, Xiamen, China. He is the coauthor of more than 300 publications, over 100 of which are refereed journal papers, including *Remote Sensing of the Environment*, IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING, IEEE GEOSCIENCE AND REMOTE SENSING LETTERS, *International Journal of Remote Sensing, Photogrammetric Engineering and Remote Sensing*, and the *ISPRS Journal of Photogrammetry and Remote Sensing*. His research interests include mobile laser scanning, point cloud processing, and feature extraction.

Dr. Li is the Chair of ISPRS ICWG I/Va on Mobile Scanning and Imaging Systems (2012–2016), the Chair of ICA Commission on Sensor-Driven Mapping (2015–2019), and an Associate Editor of IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS and IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING.

Haiyan Guan (M'15) received the Ph.D. degree in geomatics from the University of Waterloo, Waterloo, ON, Canada, in 2014.

She is currently a Professor with the College of Geography and Remote Sensing, Nanjing University of Information Science and Technology, Nanjing, China. She is the coauthor of over 30 research papers published in refereed journals and proceedings. Her research interests include airborne, terrestrial, and mobile laser scanning data processing algorithms and 3-D spatial modeling and reconstruction of critical infrastructure and landscape.

Cheng Wang (M'12) received the Ph.D. degree in information and communication engineering from the National University of Defense Technology, Changsha, China, in 2002.

He is currently a Professor and the Executive Director of the Fujian Key Laboratory of Sensing and Computing for Smart Cities, School of Information Science and Engineering, Xiamen University, Xiamen, China. He has published almost 100 papers in refereed journals and proceedings. His research interests include remote sensing image processing, mobile laser scanning data analysis, and multisensor fusion.

Dr. Wang is currently the Cochair of ISPRS WG I/3 on Multi-Platform Multi-Sensor System Calibration (2012–2016).