11-2020

Determining Ground Elevations Covered by Vegetation on Construction Sites Using Drone-Based Orthoimage and Convolutional Neural Network

Yuhan Jiang

Yong Bai

Sisi Han

Journal of Computing in Civil Engineering, Vol. 34, No. 6 (November 2020). DOI.

Determining Ground Elevations Covered by Vegetation on Construction Sites Using Drone-Based Orthoimage and Convolutional Neural Network

Yuhan Jiang
Department of Civil, Construction, and Environmental Engineering Marquette University, Milwaukee, WI
Yong Bai
Department of Civil, Construction, and Environmental Engineering Marquette University, Milwaukee, WI
Sisi Han
Department of Civil, Construction, and Environmental Engineering Marquette University, Milwaukee, WI

Abstract

Three-dimensional (3D) surveying of a construction site using an image-based method may produce incorrect ground elevation results at vegetation-covered regions, because the light rays are reflected on the surface of vegetation in front of the “truth” ground. This paper presents a convolutional neural

network (CNN) method to identify and locate static vegetation using drone-based high-resolution orthoimages. The developed CNN-based image classification models are supplemented with an overlapping disassembling algorithm to generate 8×8-pixel, 16×16-pixel, 32×32-pixel, or 64×64-pixel small-patches as model inputs. The training datasets are 10 pairs of $1{,}536 \times 1{,}536$-pixel orthoimage and label-image dataset. Experimental results show that cropping a high-resolution image into 9,025 overlapped 32×32-pixel small-patches (with a site size of $17.28 \times 17.28\ \mathrm{cm}^2$) for image classification, and assembling the small-patch label-image predictions to a patch-wise label-image prediction, has the average pixel accuracy of 92.6% in identifying objects on the experimental site. In addition, a vegetation-removing algorithm is designed to divide the label-image prediction into 36,864 nonoverlapping 8×8-pixel patches and traverse them in 192 row-loops and 191 column-loops. The testing results show vegetation in label-images are modified with the "truth" ground elevation and verified with two datasets obtained on different dates. In addition, the measured elevation differentials are close to the measured vegetation heights on the experimental site. This research has advanced the drone-based orthoimaging method in construction site surveying, which can automatically identify the static obstacles and determine the ground elevations more accurately. Furthermore, an approach of using a CNN model to segment a construction site has been proven feasible.

Introduction

Earthmoving is the primary construction activity of any new infrastructure or building project. On a construction site, site preparation works, such as grubbing and clearing, are required to remove the surface materials including trees and plants, stumps, and large roots, and other vegetation (Kim and Russell 2003). After that, the earthwork operations, such as rough grading, excavating, hauling, backfilling, compacting, and finishing works, are conducted. These operations depend on the site elevations (Kim and Russell 2003). Surveying is an important operation to get the elevation data from a construction site at the beginning and during the construction period. Recently, the construction industry has started to use remote surveying methods such as laser scanning (Du and Teng 2007; Kwon et al. 2017), drone photogrammetry (Nassar and Jung 2012; Siebert and Teizer 2014), and stereo vision (Sung and Kim 2016). These methods are highly time efficient and do not interfere with other construction operations. However, the performance of these noncontact surveying methods is affected by the plants and other ground covers on construction sites when determing the ground elevations (Westoby et al. 2012). This is because the light rays are reflected on the surfaces of vegetation instead of the "truth" ground surfaces. In contrast, the contact surveying methods with Total Station, GPS, Level, and Theodolite can obtain the expected ground elevations as all selected target points are measured on the truth ground surface. On the other side, the contact surveying methods have noticeable weaknesses because they follow a time-consuming outdoor procedure and have a high probability of interfering with other construction operations. Therefore, to improve the effectiveness of the remote surveying, automatically detecting and removing the vegetation and other obstacles from their raw surveying results and determining the truth ground elevations are necessary and important for construction professionals who heavily depend on elevation data in earthwork operations and facility layout.

Currently, detecting vegetation points from a photogrammetric point cloud based on vegetation indices and points' spatial geometrical relations (Anders et al. 2019; Cunliffe et al. 2016) has limitations because it only allows a ground point subset and nonground (vegetation) point subset to be classified. In addition, the vegetation index methods are effective in identifying green vegetation, but ineffective with other colors such as the withered vegetation and shaded vegetation, which also results in the issue of treating other green texture objects as the vegetation. Previous research has shown the feasibility of deep learning methods in object detection using image (Schneider et al. 2018), video (Kang et al. 2018), point cloud (Engelcke et al. 2017), and image segmentation (Noh et al. 2015; Badrinarayanan et al. 2017). In general, object detection includes the task of object classification and object localization. The results usually are marked with different colored boxes for identifying different objects' categories and their locations in the original image. The image segmentation is more detailed than object detection and obtains the result of a same-sized pixelwise label-image, which uses different pixel colors to represent the different objects' categories.

The computer vision community has developed several hourglass-like deep learning models for pixelwise image segmentation for road scenes and indoor scenes (Badrinarayanan et al. 2017), and biomedical imagery (Ronneberger et al. 2015). These model architectures include, but are not limited to, DeconvNet (Noh et al. 2015), FCN (Shelhamer et al. 2017), PSPNet (Zhao et al. 2017), RedNet (Mao et al. 2016), SegNet (Badrinarayanan et al. 2017), and U-net (Ronneberger et al. 2015), which are given an input image and returns a pixelwise label-image (see Table 1). These models have three common features: (1) the encoder block starts and repeats with convolution layers and max-pooling layers (except the RedNet) to generate feature-maps from the input image; (2) the decoder block uses up-sampling layers (has the same number as the max-pooling layers in the encoder block) to enlarge the feature-maps' sizes; and (3) the end of the decoder, a convolution layer or a deconvolution layer, is used to translate the feature-maps to the label-image as model output. When training these hourglass-like models, due to the insufficient GPU memory, using small-sized images for model training is required, such as resizing the ImageNet (Deng et al. 2009) down to as small as 256×256-pixel (Zhao et al. 2017). Because the original purpose of these hourglass-like models is for close-range and small-scale image segmentation, downsizing the model training datasets will not impact the model's prediction efficiency in road scenes and indoor scenes segmentation. Moreover, another approach is cropping the large-size image into small-sized patches for model training, after which, in the model prediction stage, due to the required GPU memory being much less than the model training stage, a large-sized input image can be processed by a well-trained hourglass-like model to generate a large-sized label-image production when the GPU memory is sufficient.

Table 1. Deep learning model architectures for pixelwise image segmentation

Models/references	Model training image sizes	Type of model layers
DeconvNet (Noh et al. 2015)	224×224 pixels	Convolution layer/max-pooling layer/fully connected layer/unpooling layer/deconvolution layer
FCN (Shelhamer et al. 2017)	500×500 pixels	Convolution layer/max-pooling layer/ up-sampling layer /deconvolution layer
PSPNet (Zhao et al. 2017)	256×256 pixels	Convolution layer/max-pooling layer/ pyramid pooling layer/up-sampling layer/concatenation layer

Models/references	Model training image sizes	Type of model layers
RedNet (Mao et al. 2016)	243×243 pixels	convolution layer/deconvolution layer
SegNet (Badrinarayanan et al. 2017)	360×480 pixels	Convolution layer/max-poling layer/up-sampling layer
U-net (Ronneberger et al. 2015)	512×512 pixels	convolution layer/max-poling layer/up-sampling layer

Additionally, the remote sensing and geoscience communities have developed some intelligent approaches to use the machine learning method for geospatial object detection in large scale images (Han et al. 2015), and utilize deep learning models, such as the deep convolutional neural network (DCNN) and fully convolutional network (FCN) to assist the large-scale land cover mapping in object classification to replace the traditional state-of-the-art classifier Random Forest and Support Vector Machine (Kussul et al. 2017; Liu et al. 2018). Their research objectives include, but are not limited to, landscape classification (Buscombe and Ritchie 2018), vegetation classification (Liu et al. 2018; Liu and Abd-Elrahman 2018), and crop classification (Kussul et al. 2017). In Table 2, the listed research has two common features: (1) the large-scale top-views were processed using either satellite imagery or aerial imagery, or the bundle adjustment generated orthoimage; and (2) the deep learning model was used for image patch classification, while the spatial information was given by other approaches, such as the conditional random field (Buscombe and Ritchie 2018), object-based image analysis (Liu and Abd-Elrahman 2018) and sliding window scheme (Kussul et al. 2017).

Table 2. Deep learning-based classifier in land cover mapping

Objectives/references	Deep learning models	Object classification/image patch classification	Object categories	Object localization/image segmentation
Landscape classification (Buscombe and Ritchie 2018)	MobileNetV2 DCNN (Sandler et al. 2018)	Classified the selected sparse patches (224×224 pixels) to class-labels	7 in total	Utilized conditional random field to predict pixelwise-label image with the known class-labels from the selected sparse patches
Vegetation classification (Liu et al. 2018; Liu and Abd-Elrahman 2018)	DCNN	Classified each object (corresponding to a 224×224-pixel patch) to a class-label	7 in total	As conducted in the object-based image analysis, the orthoimage was segmented to several objects by Trimble's eCognition software
Vegetation classification (Liu et al. 2018)	FCN	Translated each object corresponding patch (224×224 pixels) to a pixelwise label-image, then assigned the majority pixel label as the object class-label		
Crop classification (Kussul et al. 2017)	DCNN	Classified each window (7×7 pixels) to a class-label	11 in total	Slid the window with 1-pixel step, and assigned the returned class-label to the central pixel of each sliding window

In the proposed research project, the scene scale of the drone-based top-views and object categories in construction site segmentation tasks are different to the road scene segmentation and the land cover mapping. The existing gaps between the proposed research project and the previous project (using developed methods) include the following: (1) objects on a construction site are recorded as their top-views in the drone-based orthoimages (Fig. 1), which have much less texture feature than the side views in the road scenes; (2) for small object classification, such as cat and dog classification, the overall shape and edges are good features (Geirhos et al. 2019; Theodorus et al. 2020), while for a large area object, the texture is a usable feature when the whole object is not enclosed in the image; (3) one frame drone-based orthoimages cover less area and fewer inclosing objects than the satellite imagery and aerial imagery, and the boundaries of adjacent objects such as vegetation and shade are mixed with each other alternative to straight lines; (4) resizing the high-resolution orthoimage to fit the computing capacity of the hourglass-like deep learning models in Table 1 is not a good idea, while disassembling orthoimages into several small-sized patches is necessary to avoid reducing orthoimage size and keep the spatial information, which is referred to as the sliding window scheme (Han et al. 2015; Kussul et al. 2017) or patch-based scheme (Maggiori et al. 2016) in remote sensing and geoscience communities; (5) using small image patches with hourglass-like models to generate a pixelwise label-image, and then assigning the majority pixel label as the object class-label (Liu et al. 2018) is not necessary, because the probability of multiple objects appearing in a single image patch is going down as the patch size goes down; (6) small objects in drone-based orthoimages occupy more pixels than the small objects in the land cover mapping, and thus using the extremely thin patch (7×7 pixel) and 1-pixel step in Kussul et al. (2017) to traverse the drone-based orthoimages is not necessary, which still can cause issues for small objects, like roads and forest stripes, being smoothed and misclassified (Kussul et al. 2017); and (7) classifying a small-sized image patch into only seven types of objects with the 50-layer convolutional layers and one fully connected layer model setup in Liu et al. (2018) and Liu and Abd-Elrahman (2018) is too redundant, because the pixel-to-pixel labeling is not necessary in the patch-based image classification, and therefore adding fully connected layers to increase the model classifying capacity is more efficient than adding convolution layers to generate feature-maps.

Therefore, a patch-wise construction site segmentation and vegetation-removing framework is developed in the proposed research project. At first, the patch-based Convolutional Neural Network (CNN) approach is used to generate the patch-wise label-image for identifying vegetation on a construction site. In detail, the high-resolution orthoimage is proposed to crop into multiple overlapped small patches (50% in row and 50% in column); a CNN model serves as the classifier to identify each small-patch image as a vegetation patch or other categories and mark them with the corresponding pixel label, after which the labeled small-patches are assembled into a high-resolution result in the recorded sequence to restore the geospatial information (see Fig. 1). Because the CNN model is proposed to be trained with small-patch and class-label datasets, where small-patches are cropped from the drone-based orthoimages and manually crafted pixelwise label-images (Fig. 2), class-labels are determined by the majority pixel label in each cropped label-image small-patch. Thus, only the main object will be extracted from each small-patch by the CNN model, and the assembled result is a patch-wise label-image, which has the same size as the drone-based orthoimage. Furthermore, in this research project, the construction site elevations are saved in elevation-map format, which is an 8-bit grayscale image (Fig. 3) and each pixel value represents the elevation data for the corresponding pixel

in the orthoimage (Jiang and Bai 2020a). Thus, the elevation-map has the same pixel coordinate as the patch-wise label-image, and the vegetation removing and ground elevation determination operations could be easily conducted within them. In detail, the vegetation patches are searched from the patch-wise label-image using the pixel class-label; the ground elevation for each vegetation patch is estimated from its neighbor ground pixels' elevations based on the assumption of ground surface being smooth changes around and in the vegetation blocks. Moreover, experiments are conducted to evaluate the effectiveness of the proposed patch-wise construction site segmentation method with high-resolution orthoimage and label-image datasets and also to determine the best patch size. In addition, experiments are conducted to determine the vegetation's heights and the truth ground elevations covered by vegetation from patch-wise label-images and elevation-maps. The rest of this paper presents the research results of dataset acquisition, model training dataset creation, model architecture, and algorithm designs, and also discusses the experimental results of model training and testing, and vegetation identifying and removing on an experimental site.

Method Development

In this section, the scheme of construction site high-resolution orthoimage, label-image, and elevation-map datasets acquisition and the scheme of small-patch dataset creation are presented at first. Then, the CNN-based image classification model architecture and the overlapping small-patch disassembling and assembling algorithm are discussed. Finally, the design of a vegetation-removing algorithm is presented, which uses the pixel class-label information in the label-image to remove vegetation blocks in the elevation-map.

Dataset Acquisition

Orthoimage and Elevation-Map Acquisition

A drone-based orthoimage of a construction site can be captured by yielding the camera gimble to negative 90 degrees and stably keeping the camera lens facing the ground. The authors' recent work (Jiang and Bai 2020a) discussed a two-frame-image-based construction site elevations determination method, which utilizes a small-sized drone system to capture a low-high orthoimage pair for assembling a vertical-baseline stereo vision model; then the distances from the low-camera to the ground surface can be determined from the stereo vision model and can easily be translated to elevation data with a known control point. In addition, the elevation values are stored in an 8-bit grayscale image, referred to as an elevation-map, which has equal image size and site size as the generated orthoimage.

In this research project, the drone system, DJI Phantom 4 Pro V2, is designed to fly at $H = 10$ meters over the takeoff location to capture the construction site top-views, which have an image size of $4{,}864 \times 3{,}648$ pixels, a ground sample distance (GSD) of 0.27 cm/pixel, and a site size of 13.13×9.85 m^2 (Fig. 1). These images (including their corresponding 20 meters images) are proposed to resize down to half-size ($2{,}432 \times 1{,}824$ pixels) and cut to a square shape ($1{,}824 \times 1{,}824$ pixels) as the inputs for generating the elevation-map by the elevations determination method in Jiang and Bai (2020a). At that point, the generated high-resolution orthoimages and elevation-maps have image size of $1{,}568 \times 1{,}568$ pixels, a GSD of 0.54 cm/pixel, and a site size of 8.47×8.47 m^2, because the 128-pixel blank margins (no elevation data) are removed from

the $1{,}824 \times 1{,}824$-pixel square shape. With the 8-bit grayscale elevation-map format, the grayscale pixel value can be easily converted from range [0, 255] to its corresponding elevation value range [−5−5, 5] meters by $gray_{v,u} = 255 \times (Ele\_map_{v,u} + 5)/10$; and each 32×32-pixel patch in the elevation-map shares the same elevation value, i.e., all pixels in the patch $Ele\_map[16{:}\,48,16{:}\,48]$ have the same grayscale value/elevation value as the central pixel $Ele\_map[32,32]$.

Label-Image Creation

Fig. 2 shows the graphical user interface of the Label-App, which is designed for labeling an orthoimage with 8-bit values [0, 255] and programmed using Python 3.6.8 and matplotlib 3.1.1 library. The label-image is shown in terrain colormap for better visualization. During the label-image creation, the researchers fully mark the label-image with value 255 by default at first, and then use the cursor to point out vertices on the orthoimage for identifying each object and the keyboard to create a new class-label/value or select a predefined class-label/value such as "shade /240." Like the orthoimage, the generated label-image also has an image size of $1{,}568 \times 1{,}568$ pixels, a GSD of 0.54 cm/pixel, and a site size of 8.47×8.47 m^2. The crafted high-resolution label-images are saved in two file-formats including a grayscale image file for visualization and a 1,568-row and 1,568-column spread sheet file for training the deep learning model. Saving as a spread sheet file is necessary because the interpolation value appears on the boundaries of different objects in the image file.

Small-Patch Dataset Creation

Based on the discussion in the introduction section, the collected high-resolution datasets (the $1{,}568 \times 1{,}568$ pixel dataset is much larger than the 256×256-pixel set) cannot be directly used for training a deep learning model. In the research project, the researchers proposed to disassemble the high-resolution orthoimage and label-image dataset into four small-patch orthoimage and label-image datasets, which have the image sizes of 8×8 pixels, 16×16 pixels, 32×32 pixels, and 64×64 pixels, and site sizes of 4.32×4.32 cm^2, 8.64×8.64 cm^2, 17.28×17.28 cm^2, and 34.56×34.56 cm^2, respectively. Fig. 3 shows a high-resolution orthoimage, a label-image, and an elevation-map dataset. These images have a resolution of $1{,}536 \times 1{,}536$ pixels, which is generated by removing 16 pixels on each margin of the $1{,}568 \times 1{,}568$-pixel images, after which they can be cropped into integer numbers of 8×8-pixel, 16×16-pixel, 32×32-pixel, or 64×64-pixel small-patch side by side. Fig. 4 shows the example of these four different small-patches of orthoimages and label-images. The smallest patch (8×8 pixels) is close to the thin patch (7×7 pixels) used in Kussul et al. (2017), while the thin stripe objects on the satellite imagery do not appear in the high-resolution datasets of this research project.

Additionally, when cropping these small-patches, the strides are set as 4, 8, 16, and 32 pixels, respectively (half of the patch size), to achieve the 50% overlap in row and 50% overlap in column. The number of small-patches can be calculated by Eq. (1) for a single high-resolution orthoimage and label-image dataset. Moreover, in order to make the proposed CNN model more robust in different image orientations, the high-resolution orthoimages and label-images are planned to rotate 90, 180, and 270 degrees to augment datasets by four times. Table 3 listed the number of small-patch datasets from a $1{,}536 \times 1{,}536$-pixel orthoimage and label-image pair

(1)

$$Num.\ of\ Small\_Patches = \left(2 \times \frac{Image\ Height}{Patch\ Size} - 1\right) \times \left(2 \times \frac{Image\ Width}{Patch\ Size} - 1\right)$$

Table 3. Dataset parameters

Patch sizes	Strides	Rows	Columns	Num.	Num. after 4-rotation
8×8	4	383	383	146,689	586,756
16×16	8	191	191	36,481	145,924
32×32	16	95	95	9,025	36,100
64×64	32	47	47	2,209	8,836

Patch-Wise Construction Site Segmentation

Patch-Based Scheme

Generally, a CNN model starts with a convolution layer and ends with a fully connected layer (Fig. 5). Then for a given image input, the model output is a binary class vector (*Output_0*), which contains the probability values of the predefined class-labels only. This is different to FCN models, which can generate out the pixelwise segmentation result. Therefore, three post-processes need to be conducted to get a high-resolution segmented label-image result using CNN model predictions. First, the Argmax function is used to return the index of the maximum probability value of the binary class vector; this index is the class-label/value prediction (*Output_1*) for the input orthoimage patch. For example, the veg is the class-label prediction for the input orthoimage patch in Fig. 5, because it has the maximum value of 95% among the 256 class-labels. Second, the class-label/value prediction is assigned to each pixel of the small-patch as the label-image patch prediction (*Output_2*) for the corresponding input orthoimage patch. Third, the small-patch label-image is used to assemble the high-resolution patch-wise label-image prediction result (*Output_3*).

In this research project, the patch-based scheme is implemented with the high-resolution orthoimage overlapping disassembling and high-resolution label-image assembling algorithm in Fig. 5, which makes the CNN model work with the high-resolution image to generate the patch-wise segmentation results. On the one hand, before the CNN model, this algorithm disassembles the orthoimage into several overlapped small-patches and records their locations in their sequence ID. The number of small-patches is determined by Eq. (1). On the other hand, after the CNN model and the first-two postprocess, when using *Output_2* to assemble the high-resolution label-image prediction (*Output_3*), these small-patches are considered as corner patches, edge patches, or regular patches, and only the selected region (marked as filled rectangles) of each patch will be used in the high-resolution label-image prediction (Fig. 5). For example, 9,025 small-patches with a size of 32×32 pixels (95-row and 95-column) will be produced from a $1{,}536 \times 1{,}536$-pixel orthoimage; the CNN model outputs the same number of 32×32-pixel label-image patch predictions; then, the specific regions of these label-image patches are used to assemble a high-resolution $1{,}536 \times 1{,}536$-pixel label-image prediction, where for each regular 32×32-pixel label-image patch, the used region is only a quarter of the regular patch (16×16 pixels). Thus, in this example, each 16×16-pixel orthoimage patch is linked with a 16×16-pixel label-image patch prediction through a class-label prediction.

Therefore, the expected result is a 4×4-pixel, 8×8-pixel, 16×16-pixel, or 32×32-pixel patch-wise image segmentation result, after running the overlapping disassembling and assembling algorithm paralleled with the CNN-based image classification model with 8×8-pixel, 16×16-pixel, 32×32-pixel, or 64×64-pixel patches, respectively. This is similar to resizing a $1{,}536 \times 1{,}536$-pixel image down to a 384×384-pixel, 192×192-pixel, 96×96-pixel, or 48×48-pixel image for pixelwise image segmentation, where each pixel is a useful
4×4-pixel, 8×8-pixel, 16×16-pixel, or 32×32-pixel region in each regular patch, respectively.

Small-Patch Dataset Shape

Considering that texture is the only usable feature for classifying different objects on a construction site when the whole object is not inclosing in the small-patch, the proposed CNN model uses the RGB color orthoimage patches as model input data. Based on Table 3, a $1{,}536 \times 1{,}536$-pixel orthoimage can produce the model training datasets
with $\boldsymbol{shape}$ $(586756,8,8,3)$, $\boldsymbol{shape}$ $(145924,16,16,3)$, $\boldsymbol{shape}$ $(36100,32,32,3)$,
or $\boldsymbol{shape}$ $(8836,64,64,3)$, where the first number is the quantity of the small-patches, the second and third numbers are the size of the small-patches, and the fourth number indicates these small-patches have RGB 3-channel.

A label-image generated from the Label-App only has one channel. Disassembling a $1{,}536 \times 1{,}536$-pixel label-image can produce small-patch datasets
with $\boldsymbol{shape}$ $(586756,8,8,1)$, $\boldsymbol{shape}$ $(145924,16,16,1)$, $\boldsymbol{shape}$ $(36100,32,32,1)$,
or $\boldsymbol{shape}$ $(8836,64,64,1)$. Then, the majority (maximum frequency) pixel class-label/value in each small-patch is determined and set as the class-label/value for each label-image patch. For example, in Fig. 4 the "darker" region is larger than the "lighter" region of the 64×64-pixel label-image patch, and thus the class-label "sand"/value 80 is assigned for that small-patch. In doing so, the small-patch datasets are translated into class vector (integers), such
as $[\mathbf{130}, \mathbf{95}, \dots, \mathbf{130}]$ with $\boldsymbol{shape}$ $(586756,1)$, $\boldsymbol{shape}$ $(145924,1)$, $\boldsymbol{shape}$ $(36100,256,1)$,
or $\boldsymbol{shape}$ $(8836,256,1)$. Additionally, the class vector needs to be converted to binary class matrix with $\boldsymbol{shape}$ $(586756,256,1)$, $\boldsymbol{shape}$ $(145924,256,1)$, $\boldsymbol{shape}$ $(36100,256,1)$,
or $\boldsymbol{shape}$ $(8836,256,1)$ as the model training datasets (Chollet 2015). For example, an integer of 130 is translated to a binary class vector $[\mathbf{0.0_0}, \mathbf{0.0_2}, \dots, \mathbf{1.0_{130}}, \dots, \mathbf{0.0_{255}}]$ with $\boldsymbol{shape}$ $(256,1)$; and, then a class vector is translated to a binary class matrix with $\boldsymbol{shape}$ $(Num.\,of\ Small\_Patches, 256,1)$.

CNN-Based Image Classification Model

The CNN-based image classification model architecture is presented in Fig. 5, which includes a feature learning block and a classification block. The detailed model layers for the four different patch sizes are shown in Table 4, where the type of layers is described in the Keras 2.3 style (Chollet 2015). In the feature learning block, three convolution layers learn the orthoimage patches (model input) as feature-maps (layer outputs). Three max pooling layers reduce the feature-maps' (layer inputs) size to its half-size as their layer outputs without losing important features. For example, the 8×8-pixel, 16×16-pixel, 32×32-pixel, and 64×64-pixel patches are resized down to 1×1-pixel, 2×2-pixel, 4×4-pixel, and 8×8-pixel patches, respectively, after the 3rd max pooling layer. The flatten layer transforms the feature-map (layer input) into a feature-vector (layer output), which can be used in the classification block. Three fully connected layers (also known as dense layers) translate feature-vectors

(layer inputs) to a binary class vector $[\mathbf{0.0_0}, \mathbf{0.0_2}, \ldots, \boldsymbol{p_i}, \ldots, \mathbf{0.0_{255}}]$ as the CNN model output for each orthoimage patch input.

Table 4. Model layer parameters

Model architecture for 8×8, 16×16, 32×32, and 64×64-pixel patches					Output shapes for each patch				
					Row×column				
Blocks	Layer (type and filter size)	Stride	Padding	Activation	8×8	16×16	32×32	64×64	Channels
Input	input_1 (Input Layer)	—	—	—	8×8	16×16	32×32	64×64	3
Feature learning block	conv2d_1 (64,Conv2D 3×3)	1	same	ReLU	8×8	16×16	32×32	64×64	64
	max_pooling2d_1 (Max Pooling 2×2)	2	—	—	4×4	8×8	16×16	32×32	64
	conv2d_2 (128,Conv2D 3×3)	1	same	ReLU	4×4	8×8	16×16	32×32	128
	max_pooling2d_2 (Max Pooling 2×2)	2	—	—	2×2	4×4	8×8	16×16	128
	conv2d_3 (256,Conv2D 3×3)	1	same	ReLU	2×2	4×4	8×8	16×16	256
	max_pooling2d_3 (Max Pooling 2×2)	2	—	—	1×1	2×2	4×4	8×8	256
	dropout_1 (Dropout 0.5)	—	—	—	1×1	2×2	4×4	8×8	256
Classification block	flatten_1 (Flatten)	—	—	—	256	1,024	4,096	16,384	—
	dense_1 (Dense)	—	—	ReLU	256	1,024	2,048	4,096	—
	dropout_2 (Dropout 0.5)	—	—	—	256	1,024	2,048	4,096	—
	dense_2 (Dense)	—	—	ReLU	256	512	1,024	1,024	—
	dropout_3 (Dropout 0.5)	—	—	—	256	512	1,024	1,024	—
Output	dense_3(Dense)	—	—	SoftMax	256				—

Furthermore, after each convolutional layer and dense layer, there is an activation function (layer), which performs the nonlinear transformation of the input features from the previous convolutional layers or dense layers (Dettmers 2015). Because the model input datasets will be normalized from value range [0,255] to [0.0,1.0] by dividing them by 255, the activation function should progressively change from 0.0 to 1.0 with no discontinuity. Therefore, the rectified linear unit activation function (ReLU), $f(x) = \max(0, x)$, is used in hidden layers. Because the ReLU function does not always output a nonzero value, which results in less neurons being utilized and less dependence between features (Nair and Hinton 2010), it is faster than the Sigmoid activation functions. In addition, the SoftMax activation function is used in the 3rd dense layer to calculate the probabilities of the 256 class-labels in the binary class vector $[\mathbf{0.0_0}, \mathbf{0.0_2}, \ldots, \boldsymbol{p_i}, \ldots, \mathbf{0.0_{255}}]$. Finally, the dropout layers are used to prevent model overfitting, which randomly sets half of the input units to 0 during the model training (Chollet 2015).

Additionally, for compiling the proposed CNN-based model, the researchers use “adam” as the optimizer, “categorical_crossentropy” as the loss function, and use “accuracy” as the metric. The “validation_split” is set to 0.05, which means that 95% of small-patch datasets are used for training the model and 5% of small-patch datasets are used for model validation. The “early stopping” configuration is set as “EarlyStopping(monitor=‘val_accuracy’, patience=5),” which means the model training will be stopped because the monitored quantity of validation accuracy had stopped improving for the past five epochs (Chollet 2015).

Patch-Wise Vegetation Removing

There are two approaches for removing the vegetations’ heights from the raw surveying result (elevation-map) and determining the truth ground elevations using the identified vegetation blocks in the patch-wise label-image. The first approach is measuring an average height of vegetation blocks on the construction site, and then directly subtracting this value in the elevation-map for the vegetation blocks. This may cause irregularity elevation changes on vegetation blocks’ boundaries, where the vegetation has a lower height than the central region. However, this approach has the advantage in dense vegetation areas, such as the bottom of Fig. 8, where have no ground surface shows in the top-view. The second approach is estimating an average elevation of neighbor ground surfaces in the elevation-map, and then uses this value for updating the vegetation blocks’ elevations. This approach is similar to the iterations of interpolation method in removing vegetation points from a point cloud, which classifies points above the interpolated surface as vegetation, and interpolates again with a new selection of potential ground points (Anders et al. 2019). The second approach works for sparse vegetation areas or isolated plants, such as the marked vegetation blocks in Fig. 8, where the ground surfaces or neighboring grounds appear in the orthoimages and elevation-maps, and searching neighbor ground blocks with the label-image and interpolating these surroundings’ elevation values as the estimated truth ground elevation under the vegetation is possible.

In this research project, the patch-wise vegetation removal focuses on the isolated and sparse vegetation blocks on the construction sites (Fig. 8). The proposed vegetation-removing algorithm in Fig. 6 is based on Approach 2, which is more convenient for automatically estimating the ground elevation without any manual participation, and the result is more smooth at the boundaries than Approach 1. In detail, the proposed ***VEG_REMOVING_IN_ROW_THEN_COL_TRAVERSE*** algorithm

traverse the patch-wise label-image in the row-column-row-loop shown in Fig. 7, which ends with a row-loop. In each row-loop, the ***SEARCH_VEG_REPLACE_GROUND*** algorithm uses an adjustable window, which can be extended in the row direction only to search the minimum required number of ground pixels using the pixel class-label in the label-image. Similarly, in each column-loop, the adjustable window is changed in column direction only, to search the minimum required number of ground pixels as well. When sufficient ground pixels appear in the search window, the ***SEARCH_VEG_REPLACE_GROUND*** algorithm replaces the current vegetation patch's elevation value $Ele\_map[row_{index}: row_{index} + stride, col_{index}: col_{index} + stride]$ in the elevation-map with the average elevation value *ground_ele* from the searched neighboring ground pixels. An alternative option is only replacing any pixel $Ele\_map[v, u]$ in the elevation-map when its elevation value is higher than *ground_ele*. This will assist in keeping the sparse truth ground elevation in the elevation-map, which classifies the lower pixels as the ground pixels, alternative to the vegetation pixels labeled in the patch-wise label image. In addition, the removed vegetation patches will be marked with a new ground class-label in the label-image, and drawn with a specific color in the orthoimage as well (see Fig. 7).

Experiments

In this section, the patch-wise construction site segmentation method is compared in the 8×8-pixel, 16×16-pixel, 32×32-pixel, or 64×64-pixel patch-based CNN-based image classification models (Fig. 5 and Table 4) at first. Then, the vegetation-removing and truth ground elevation determination experiment is evaluated with the best patch-wise segmentation result of the experimental site (i.e., a lake beach site, as in Fig. 8). In this research project, the configuration of the software environment is Python 3.6.8, OpenCV 3.4.2, Keras 2.3.1, TensorFlow-GPU 1.14, CUDA 10.0, and cuDNN 7.6.4.38 on a workstation system with 2×Xeon Gold 5122@3.6GHz CPUs, 96GB (8GB×12) DDR4 2666 MHz memory, and 4×11GB memory GeForce RTX 2080 Ti GPUs.

Construction Site Segmentation

Training Dataset

Followed by the high-resolution dataset acquisition method, 10 $1{,}536 \times 1{,}536$-pixel orthoimages were collected during 2019 (Fig. 9), and the corresponding label-images were labeled with the 10 categories of objects and surfaces in Table 5. For the vegetation blocks, in data A and B, the vegetation had not recovered yet; in data C and D, the vegetation was growing; and in data G, O, AD, AL, AM, and CG, the vegetation was fully grown, and their heights were around 2–3 ft (0.6096–0.9144 m0.6096–0.9144 m, Fig. 8).

Table 5. Class-label definitions

Class-label	8-bit grayscale value	Definitions
n	255	Default value/other undefined objects
Shade	240	Shades on ground
Umbrella	220	Red umbrella surface
Can	180	Garbage cans
Shrub	150	Shrub surface
Veg	130	Vegetation surface
Withered	110	Withered vegetation surface

Class-label	8-bit grayscale value	Definitions
Sand	80	Ground surface, includes sand and soil
wood	30	Wooden surface, includes platform and path
takeoff	0	Drone takeoff and landing pad

Four small-patch orthoimage datasets were generated from 10 orthoimages, which have the following: $\boldsymbol{shape}$ $(5867560,8,8,3)$, $\boldsymbol{shape}$ $(1459240,16,16,3)$, $\boldsymbol{shape}$ $(361000,32,32,3)$, and $\boldsymbol{shape}$ $(88360,64,64,3)$, respectively. Furthermore, four binary class matrixes with $\boldsymbol{shape}$ $(5867560,256,1)$, $\boldsymbol{shape}$ $(1459240,256,1)$, $\boldsymbol{shape}$ $(361000,256,1)$, and $\boldsymbol{shape}$ $(88360,256,1)$ were produced for the 8×8-pixel, 16×16-pixel, 32×32-pixel, and 64×64-pixel small-patch label-image dataset, respectively. Therefore, the four small-patch orthoimage datasets and the four binary class matrixes were assembled as the four model training datasets for the four CNN models in Table 4, respectively.

Training and Validation

The CNN model training parameters including dataset numbers, batch sizes, and epochs are listed in Table 6. The results of training loss, training accuracy, validation loss, and validation accuracy with early stopping for the four different patch sizes are shown in Fig. 10, which were stopped at different epochs (see Table 6). The 64×64-pixel and 8×8-pixel patch trials stopped at the 13th epoch and were the earliest trials, and the 32×32-pixel patch stopped at the 14th epoch. The 16×16-pixel patch took the most epochs for the validation accuracy to reach stable.

Table 6. Model training parameters and results

Patch size trials					Training epoch trials	
Patch sizes	Datasets (validation split=0.05split=0.05)			Batch sizes	Early stopping (monitor='val_accuracy', patience=5), epochs=50	
	Total No.	Training	Validation		w/ early stopping	w/o early stopping
8	5,867,560	5,574,182	293,378	256	13	50
16	1,459,240	1,386,278	72,962	256	24	50
32	361,000	342,950	18,050	256	14	50
64	88,360	83,942	4,418	256	13	50

Several small-patch validation samples are shown in Fig. 11, where the model training datasets of label-image patches (class-labels, or binary class vectors in the CNN model) are compared with the model predictions. These samples show the larger patches, i.e., the 32×32-pixel and 64×64-pixel, were more accurate than the smaller patches, i.e., the 8×8-pixel and 16×16-pixel. Although the large patches formed complex label-image patches with multiple objects (class-labels) in a single label-image patch, the CNN model–generated class-labels were the same as the corresponding class-labels in the training dataset. The overall validation accuracy of the randomly selected 5% small-patch datasets also confirmed that the 32×32-pixel and 64×64-pixel patches were more accurate than the 8×8-pixel and 16×16-pixel patches (Fig. 10). However, it is hard to conclude that either the 32×32-pixel or 64×64-pixel batch has the best performance in the small-patch classification task based on these early stopping trials.

In addition, the additional model trainings were conducted without early stopping to 50 epochs. In Fig. 12, the 64×64-pixel patch model has the largest model training accuracy of 0.9908 at the 50th epoch, but it is an overfit model because its validation accuracy of 0.9219 at the 50th epoch did not improve as the training accuracy did. The designed three dropout layers showed the limited function in avoiding the model overfitting issue in the 8×8-pixel, 16×16-pixel, and 32×32-pixel patch models. In Fig. 12, the extra training shows a slight negative impact on the 8×8-pixel patch model's training accuracy and training loss, whereas it shows a slight improvement in the 32×32-pixel patch model training accuracy and training loss. However, the extra training has neither significantly good nor bad impact on the model validation accuracy and validation loss. For example, the 32×32-pixel model has the best validation accuracy of 0.9304 at the 50th epoch, which is not much different from the 0.9288 at the early stopping trial. The cause of overfitting can be visualized in the assembled patch-wise validation results as well. In Fig. 13, compared to the early stopping, the 50-epoch has the noise predictions on the wooden platform of data AM and G, but it has better model predictions for the "withered" class-label in data A and CG, so the overall model validation accuracy was maintained around 93% for the 32×32-pixel patch. Therefore, considering the 32×32-pixel patch had the smallest model validation loss and the best model validation accuracy in the small-patch image classification, the researchers conclude that the 32×32-pixel patch (with a site size of $17.28 \times 17.28\ \text{cm}^2$) has the best performance in construction site patch-wise segmentation, followed by the 64×64-pixel patch and 16×16-pixel patch. The smallest 8×8-pixel patch, however, has the worst performance.

Testing and Evaluation

The trained early stopping and 50-epoch models were tested with the data AO in Fig. 3. The orthoimage and label-image were rotated and repeatedly disassembled into four small-patch orthoimage and binary class vector datasets, which have the numbers listed in the last column of Table 3. For example, the created testing dataset for the 32×32-pixel patch was 36,100 pairs of a small-patch orthoimage and binary class vector dataset. The best image classification testing accuracy of 0.9435 is the 32×32-pixel patch with 50-epoch (overfitting), the second-best testing accuracy of 0.9433 is the 64×64-pixel patch with 50-epoch (overfitting), and the third-best testing accuracy of 0.9423 is the 32×32-pixel patch with early stopping. Thus, about 94% of the small-patch orthoimages were assigned the correct class-labels by the CNN model. For the assembled patch-wise label-image in Fig. 14, the 32×32-pixel patch with early stopping shows the best segmentation result, followed by

the 32×32-pixel patch with 50-epoch (overfitting). As for the results of model overfitting, the worse "wood" and "can" prediction performance and better "withered" prediction performance appeared after the early stopping point, which are the same as the CNN model validation results. Thus, the researchers conducted additional testing with the CNN-based image classification model of the 32×32-pixel patch with early stopping only, where orthoimage data AO, K, and Z were tested without rotations, and each of the patch-wise segmentation results were assembled from 9,025 overlapped small-patch label image predictions.

Fig. 15(a) mapped the unmatched pixels between the manually crafted pixelwise label-image (left) and the patch-wise segmentation results (right), where the testing data AO, K, and Z had a pixel accuracy of 93.57% (2,207,641 of 2,359,296 pixel), 93.61%, and 90.64%, respectively. There are noticeable unmatched pixels on the boundaries of different objects, which are reasonable results because the comparisons are between a pixel and a 16×16-pixel patch (a quarter of 32×32-pxiel). Excluding the boundaries, the majority of unmatched pixels were between withered and veg, withered and sand, and shade and veg, where the CNN-based image classification results were more accurate than human eyes [Fig. 15(b)]. In this research project, the withered class-label was defined as a ground surface category between the sand and sparse veg; the shrub class-label was defined as dense plants other than the sparse veg; and the shade class-label was defined as the shade on the ground surface. Although the researchers tried hard to distinguish the different objects from the orthoimages, errors had mixed in with the manually crafted label-images somewhere. The small veg on the wooden path of data K was mislabeled but successfully identified by the CNN model [Fig. 15(b)]. However, the mislabeled boundaries of shrub and veg in the model training dataset resulted in the "well" trained CNN model identifying the veg patches with highlighted leaves and dark background as the wrong shrub patches [Fig. 15(c)]. This explains why the Intersection over Union (IoU) for shrub, shade, and withered were worse than the other class-labels in Table 7. Moreover, in the early stage of this research project, the researchers obtained a 0.9646 validation accuracy and 0.9673 testing accuracy in image patch classification without adding the withered class-label. Thus, the performance of patch-wise segmentation can be improved by considering the withered and sand as one ground surface category, and considering the mixed veg, shrub, and shade as one vegetation category. Furthermore, the pixel accuracy of 93.57% of data AO is not significantly different to its small-patch classification testing accuracy of 94.23%. Thus, the developed overlapping small-patch disassembling and assembling algorithm was efficient in the patch-wise segmentation task with an average pixel accuracy of 92.6%, which has the good performance for the large area objects, such as the IoU 0.9827 for wood and 0.8666 for veg in the three testing datasets. The detailed IoU for each class-label of the model training and testing datasets are summarized in Table 7.

Table 7. Model training and testing IoU

Class-label	Value	Model validation IoU (intersection over union)										Model Testing IoU			
		A	B	C	D	G	O	AD	AL	AM	CG	Average	AO	K	Z	Average
n[a]	255	—	0.0000[a]	0.0000[a]	—	—	—	0.0000[a]	—	0.0000[a]	—	—	—	—	—	—
Shade	240	—	—	0.7711	0.8853	0.7782	0.7588	0.5133	0.4902	0.6808	—	0.6968	0.4950	0.1557	0.1621	0.2709
Umbrella[b]	220	—	—	—	—	—	—	0.0000	—	—	0.9682	0.4841	—	—	—	—
Can[c]	180	0.0000	0.9097	—	0.8739	—	—	0.8406	0.8807	0.8594	—	0.7274	0.8483	—	—	0.8483
Shrub[d]	150	—	—	0.8904	0.0000	0.0000	0.9384	0.0000	—	—	0.9401	0.4615	0.0000	0.0000	0.0000	0.0000
Veg	130	0.8783	0.8426	0.9635	0.8869	0.9050	0.9501	0.9333	0.8672	0.8787	0.8663	0.8972	0.8360	0.9137	0.8502	0.8666
Withered[e]	110	0.5003	0.6243	0.3860	0.5040	0.4793	0.4164	0.0000	0.1989	0.1925	0.4697	0.3772	0.1840	0.0928	0.2789	0.1852
Sand[f,g]	80	0.7476	0.8107	0.8681	0.8980	0.8545	0.0000[f]	0.8353	0.8845	0.7726	0.8415	0.8348	0.6832	0.0000	0.7446	0.4759
Wood	30	0.9803	0.9939	0.9762	0.9932	0.9794	0.9544	0.9907	0.9772	0.9915	0.9874	0.9824	0.9886	0.9720	0.9875	0.9827
Takeoff	0	—	—	—	0.8890	—	—	0.8662	0.8917	0.8748	—	0.8804	0.8757	—	0.5258	0.7007
Mean IoU		0.6213	0.6969	0.6936	0.7413	0.6661	0.6697	0.4979	0.7415	0.6563	0.8455	0.6830	0.6138	0.3557	0.5070	0.4922
Corrected mean IoU[a,f]			0.8362[a]	0.8092[a]			0.8036[f]	0.5533[a]		0.7501[a]		0.7368[a,f]				
Pixel accuracy		0.9123	0.9477	0.9722	0.9620	0.9397	0.9662	0.9599	0.9245	0.9483	0.9680	0.9501	0.9357	0.9361	0.9064	0.9261

Note: The above two errors were excluded to get the corrected Mean IoU should refer to the errors of a and f.

[a] When manually crafting a label-image, all pixels were set to the default value 255 at first, which resulted in 141, 4, 8, and 58 pixels nn on Data B, C, AD, and AM.

[b] The small corner (337 pixels) of the umbrella in Data AD (left-bottom) was not identified by the CNN model, while the umbrella had a good performance in Data CG.

[c] This error occurred in Data A (left-upper) with 256 pixels, where the pixels are the ground surface on the experimental site.

[d] Shrub was labeled in the bottom of Data C, O, and CG and well identified, but 4,864, 1,664, 256, 256, 72,192, and 14,080-pixel errors occurred in Data D, G, AD, AO, K, and Z, respectively.

[e] No pixel was labeled as withered in Data AD, but the CNN model prediction was correct.

[f] The isolated withered block in Data O was labeled with sand in the past, which resulted in 37 pixels not being completely covered by the correct withered class-label.

[g] The prediction error occurred on the wooden path in Data K (left), and 103 pixels in the sand path were not covered by the correction withered while creating the label-image.

Furthermore, the vegetation index $ExG = 2G - R - B$ (Anders et al. 2019) was applied to identify vegetation ($ExG > 0.1$) in the model training and testing dataset. Fig. 16 shows this vegetation index method was only sensitive to a range of green colors and also resulted in the error at the garbage cans in data AO. However, vegetation in data A were not identified; the sparse vegetation with dark color were skipped in data G, AO, K, and Z; and some of the dense vegetation in data C and CG were missed as well. Thus, the vegetation index method is not suitable for the detailed vegetation detection on a complicatedly textured construction site with other green textured objects. Therefore, the researchers conclude that the developed CNN-based image classification model with the 32×32-pixel ($17.28 \times 17.28\ \mathrm{cm}^2$) patch has good accuracy in identifying objects on the construction site using the drone-based high-resolution orthoimage.

Vegetation-Removing Testing

Algorithm Configuration

In this research project, the patch-wise vegetation-removing experiments were conducted with the 32×32-pixel early stopping patch-wise segmentation predictions. The proposed algorithm in Fig. 6 was programmed using Python 3.6.8, with the following parameter settings. The shade, shrub, and veg were set as the ***veg_label_list*** (in Fig. 6), which were considered as vegetation blocks in the label-image, and needed to be removed and replaced with class-label ground/ value 95.

The $\boldsymbol{ground_{label}}$ was set as sand, withered, and the relabeled ground, which were considered as ground blocks to provide the elevation values for vegetation blocks. The initial search window was set with size $(qszie \times 2ratio_q + stride) \times (qszie \times ratio_q + stride)$, where $qsize$ is the small-patch (32-pixel) used in the CNN-based image classification model. The stridestride is the step used for traversing label-images, which was set as $qsize/4 = 8$-pixel; thus, a $1{,}536 \times 1{,}536$-pixel patch-wise label-image was disassembled into 36,864 small-patches (192-row, 192-column) with a size of 8×8 pixels, and traversed by 192 row-loop and 191 column-loop. The $ratio_q$ is the parameter used to control the initial size of the search window and the required number of ground pixels in the search window, which impacts on the smooth degree of estimated truth ground elevations on the vegetation removed elevation-map. This research project used a large $ratio_q = 8$ to get smooth elevation changes on the boundaries of vegetation and ground surfaces, and then, the minimum required number of ground pixels in the search window was $qsize \times qsize \times ratio_q = 32 \times 32 \times 8$. The maximum search windows size depends on the parameter win_{size_max}, which was set as half of the image $\mathrm{width} = 768$-pixel to handle the extreme condition that the ground surface only appeared in corners or edges, such as data CG. In this case, the first row-column-row-loop was not enough to remove the vegetation on bottom-left corner, and then, the additional row-column-row-loop successfully removed all vegetation and marked them with the pink color in the orthoimage [Fig. 17(a) and Table 8].

Table 8. Pixel class-label summary

Class-label	Value	Label-image prediction								Vegetation removed label-image prediction				
		D		AO		G		CG		D	AO	G	CG-1[a]	CG-2[a]
n	255		—	—	—	—	—	—	—	—	—	—	—	—
Shade	240	21,760	0.92%	8,192	0.35%	229,632	9.73%	—	—	—	—	—	—	—
Umbrella	220	—	—	—	—	—	—	436,224	18.49%	—	—	—	436,224	436,224
Can	180	18,688	0.79%	73,856	3.13%	—	—	—	—	18,688	73,856	—	—	—
Shrub	150	4,864	0.21%	256	0.01%	1,664	0.07%	455,424	19.30%	—	—	—	59,776	—
Veg	130	444,480	18.84%	427,456	18.12%	1,003,520	42.53%	403,840	17.12%	—	—	384[b]	103,744	—
Withered	110	75,136	3.18%	26,624	1.13%	41,216	1.75%	25,600	1.09%	—	—	—	—	—
Ground[c]	95	—	—	—	—	—	—	—	—	925,440	728,832	1,842,176	797,568	961,088
Sand	80	379,200	16.07%	266,304	11.29%	566,528	24.01%	76,224	3.23%	—	—	—	—	—
Wood	30	1,396,480	59.19%	1,539,456	65.25%	516,736	21.90%	961,984	40.77%	1,396,480	1,539,456	516,736	961,984	961,984
Takeoff	0	18,688	0.79%	17,152	0.73%	—	—	—	—	18,688	17,152	—	—	—
Sum	—	2,359,296	100.00%	2,359,296	100.00%	2,359,296	100.00%	2,359,296	100.00%	2,359,296	2,359,296	2,359,296	2,359,296	2,359,296

[a] Two row-column-row-loops were conducted to remove all vegetation.

[b] 384-pixel of unremoved veg at the top-left corner of data G.

[c] Ground = shade + shrub + veg + withered + sand.

Furthermore, a shade, shrub, or veg pixel in the label-images was updated its elevation with the average elevation value (*ground_ele*) of the searched neighboring ground pixels when this vegetation pixel's elevation $Ele\_map[v, u]$ in the elevation-map was higher than the *ground_ele*. Moreover, to get the smooth elevation changes on the boundaries of ground blocks and vegetation blocks, during the row-column-row-loop, the sand and withered pixels in the label-image were given updated elevations *ground_ele* and remarked with the class-label ground/ value 95 as well. Thus, after all vegetation pixels are removed by the developed algorithm, the total number of shade, shrub, veg, sand, and withered pixels in the patch-wise label-image prediction should be equal to the number of ground pixels in the vegetation removed label-image (Table 8).

Testing and Evaluation

The testing data AO in Fig. 3, and CNN model training data D and data G were used to evaluate the vegetation removing algorithm. Table 8 shows the sum number of shade, shrub, veg, sand, and withered pixels in the label-image prediction is equal to the number of ground pixels in the vegetation removed label-image, which confirms that the developed algorithm had successfully traversed the high-resolution patch-wise label-image in a single row-column-row loop [except the 384 veg pixels in data G, see Fig. 17(a)].

Fig. 17(b) shows the elevation differentials between the original elevation-maps and vegetation removed elevation-maps on ground blocks and vegetation blocks. There are larger elevation changes appearing on the edges of the wooden platform and garbage cans, where the updated elevations fixed the errors in the elevation-map. That is because in the elevation-map, each 32×32-pixel patch shared the same elevation value even if this patch contains different objects, while each 8×8-pixel small-patch in the vegetation removed elevation-map shared the same elevation value based on the object's class-label. Excluding these elevation corrections, the elevation differentials represent the vegetation heights on the experimental site. In Fig. 17(b), data AO has the larger elevation differential than data D for the same vegetation blocks, which correctly reflects the growing of the vegetation blocks from 6/5/2019 to 9/5/2019. The elevation differential in data G is larger than data AO, which also reflects the vegetation heights on the experimental site (Fig. 8). The detailed vegetation heights (elevation differentials) and the peak values of data D, AO, and G are shown in Fig. 18(a), where the data D has the peak value of 0.4706 m, data AO has the peak value of 0.6275 m, and data G has the peak value of 0.9412 m (considering the single value 0.9804 m as noise point). These peak values are close to the measured vegetation heights 0.6096 and 0.9144 m on the experimental site. Thus, any elevation differentials larger than these three peaks in these three data were considered as the noise points alternative to vegetation heights.

In Fig. 18(a), the point clouds were generated using the selected central points of each 8×8-pixel patch of the orthoimage (textures) and the elevation-differential-map (vegetation heights). Thus, 36,864 points were generated for each data, but only 6,945 veg points, 6,679 veg points, and 15,680 veg points were retained for analysis in data D, AO, and G, respectively [see the statistics summary in Fig. 18(b)]. That is the same as the percentage of veg pixels of the label-images in Table 8, where 18.84%, 18.12%, and 42.53% represent veg pixels in data D, AO, and G, respectively. In addition, three subset point clouds, named $\mathrm{D} < 0.4706$, $\mathrm{AO} < 0.6275$, and $\mathrm{G} < 0.9804$ were created by removing the points that have the elevation differential larger than the peak value of each data. Thus,

for outliers, most are the elevation corrections on the edges of the wooden platforms, which were excluded from the subsets [see the boxplot in Fig. 18(b)]. In data D and AO, their differential of peak value is 0.1569 m, the 2-sample t-Test was conducted for these two subsets, which 95% confidently concluded that the mean of $\mathrm{AO} < 0.6275$ is 0.0805 m greater than $\mathrm{D} < 0.4706$. Similarly, the mean of $\mathrm{G} < 0.9804$ is 0.1402 m greater than $\mathrm{AO} < 0.6275$ at the 0.05 level of significance. In Fig. 18(b), three contour plots were crafted based on the three subset point clouds. The results indicate the largest vegetation height in data AO was 0.6 m for the right vegetation block, and 0.42 m for the left vegetation block; the largest vegetation height in data D was 0.45 m for the right vegetation block, and 0.25 for the left vegetation block; the largest vegetation height in data G was 0.9 m for the two isolated vegetation blocks. Therefore, the researchers conclude that the measured elevation differentials successfully reflect the vegetation heights on the experimental site.

In Fig. 19, the vegetation-removed point clouds were generated using the selected central points of each 8×8-pixel patch of the vegetation removed orthoimage (textures) and the vegetation removed elevation-map (elevation values), where 14,460 ground points and 11,388 ground points appeared in data D and data AO, and account for 39.23%, 30.89% of the total generated 36,864 points, which are the same percentages of ground pixels in the vegetation-removed label-images. Because the generated point clouds have the similar shape visually, the contour plots confirm the ground region are similar to each other, and the histogram shows the ground elevations have the similar distribution, the researcher conducted the additional 2-sample tt-Test for verifying these two ground point clouds are similar or not. The hypothesis test results indicated the difference between these two ground point clouds (*Difference=AO_GROUND–D_GROUND*) has the 99% CICI of (-0.010172, 0.00072689), and the standard deviations do not differ at the 0.01 level of significance for these two ground point clouds. Thus, the estimated ground elevations in data AO and data D are close to each other, and the measured vegetation heights were fully removed from the raw surveying results. Therefore, the researchers conclude that the developed vegetation-removing algorithm is stable in estimating the truth ground elevations, and the performance is robust under the different conditions of the covered vegetation.

Furthermore, Fig. 20(a) shows a stitched point cloud demo of the data AO (mid), data G(right), and data AD (left), where the vegetation were removed in three separated orthoimages and elevation-maps. These ground points have small elevation gaps on the joints between data AD and AO, and data AO and G. The smooth ground elevation changes on the joints could be achieved by stitching the three orthoimages and elevation-maps at first, and then, patch-wise segmenting the stitched orthoimages with the patch-based CNN model, and followed by the patch-wise vegetation removing.
Fig. 20(b) shows a large-scale demo, where the $4{,}864 \times 3{,}648$-pixel orthoimage is the original image captured by the drone at 20m, which had a $\mathrm{GSD\ of} = 0.54\ \mathrm{cm/pixel}$ and site size of $26.26 \times 19.70\ \mathrm{m}^2$. The patch-wise label-image was generated by the 32×32-pixel early stopping model. The elevation data was generated by the elevation estimation method in Jiang and Bai (2020b). 456-row loops and 455-column loops were conducted to remove the vegetation and estimate the ground elevations. Because the developed patch-wise construction site segmentation and vegetation-removing framework can be extended in these two cases, which is not limited to work with the model training and testing dataset, the researchers conclude that the developed framework in this paper

(Fig. 1) can automatically identify the vegetation and determine the truth ground elevation covered by vegetation on a construction site.

Conclusion and Future Research

This paper presents a deep learning-based method to identify vegetation objects on a construction site using drone-based orthoimage and determine the truth ground surface elevations from the raw surveying results. The keypoints of the method are outlined in Fig. 1, which includes: (1) using a drone to acquire construction site orthoimages; (2) disassembling the high-resolution orthoimage into overlapping small-patches; (3) using the CNN-based image classification model to generate the class-label for each orthoimage patch; (4) assigning the class-label to each pixel of the small-patches to generate the label-image predictions for each orthoimage patch; (5) assembling small-patch label-image predictions to a high-resolution patch-wise label-image prediction; (6) searching and identifying vegetation blocks in the patch-wise label-image; (7) updating vegetation block elevation values with the surrounding grounds' elevations in the same coordinate elevation-map; and (8) converting the vegetation removed elevation-map to elevation data or three-dimensional (3D) point cloud of the construction site.

The CNN-based image classification models with 8×8-pixel, 16×16-pixel, 32×32-pixel, and 64×64-pixel patches were tested and compared. The testing results showed the 32×32-pixel patch (size $= 17.28 \times 17.28$ cm^2) had the best performance of 94% accuracy in identifying the main objects' class-label from each small-patch orthoimage on the construction site. The developed overlapping disassembling and assembling algorithm, which runs in parallel with the CNN model, contributes to making the workstation system more convenient to train the CNN model with high-resolution images instead of shrinking images and losing image details. By cropping the datasets into multiple overlapped small-patches, the model training datasets were augmented as well. The testing results show that the developed patch-wise segmentation method, which disassembling the $1{,}536 \times 1{,}536$-pixel high-resolution image into 9,025 overlapping small-patches for image classification and assembling the label-image small-patch predictions to the $1{,}536 \times 1{,}536$-pixel patch-wise label-image prediction is an effective image segmentation method with an average pixel accuracy of 92.6% and high IoU for large area objects. In addition, with this suitable small-patch size, the edges of different objects were well determined and applied to fix the elevation errors occurred on the edges of different objects in the elevation-maps.

Additionally, after the objects on a construction site were identified in a $1{,}536 \times 1{,}536$-pixel patch-wise label-image prediction, a vegetation-removing algorithm was used to divide this high-resolution label-image into 36,864 nonoverlapping 8×8-pixel patches and traverse them into 192 row-loops and 191 column-loops. In each row-loop and column-loop, the developed algorithm extended search windows in the row and column direction, respectively. It searched the sufficient surrounding ground pixels first. Then, elevation values of the current vegetation patch in the corresponding elevation-map were replaced with the average elevation from the searched neighbor ground pixels. The testing results showed that vegetation blocks on the high-resolution label-image were removed and the truth ground elevations were determined, and the measured elevation differentials reflected the vegetations' heights measured on the experimental site.

To fully remove the static vegetation blocks on the experimental site, the CNN-based image classification model was trained with the datasets collected from the still experimental site, which only contain the static objects such as the static vegetation block and static structures. The experimental results confirm that the developed patch-wise construction segmentation and truth ground elevation determination framework works on the experimental site, while applying it on an active excavation construction site, and further work is required in model training dataset expansion, such as including the top-views of all potential static and dynamic objects on construction sites. This is because an active construction site is much more complex than the still construction site. The dynamic objects, such as excavators, dozers, trucks, and workers on the construction site have impacts on accurately determining the ground elevations using the remote surveying methods. For example, a dozer may be included in an elevation-map or a drone photogrammetric point cloud to compute automatically and correctly determine the truth ground elevation under the dozer; this equipment should be identified at first, and then its height should be removed from the surveying result.

The success of this research project contributes to the advancement of drone application and deep learning methods in construction site surveying. The researchers provided and verified a feasible approach of using a CNN model to patch-wise segment a high-resolution drone-based orthoimage of construction sites with a high pixel accuracy and acceptable IoU. The developed model can be used for automatically identifying and locating multiple categories of static objects from the raw surveying results, which is more than identifying only vegetation or nonvegetation categories by the vegetation index method or iterations of interpolation method. In addition, this model can be extended to removing dynamic objects from the high-resolution orthoimaging videos. As a result, the research project proved that it is possible to use drone technologies to make the image-based construction surveying and measurement of ground elevations much more accurate and convenient.

Data Availability Statement

The model training and testing datasets (orthoimages and label-images appear in Fig. 3 and Fig. 9) are available from the corresponding author upon request. The Python code of the CNN-based image classification model (in Fig. 5 and Table 4) and vegetation-removing algorithm (in Fig. 6) are also available from the corresponding author upon request.

Acknowledgments

This research project was financially supported by the McShane Endowment Fund at Marquette University. The authors are thankful for the reviewers' valuable comments.

References

Anders, N., J. Valente, R. Masselink, and S. Keesstra. 2019. "Comparing filtering techniques for removing vegetation from UAV-based photogrammetric point clouds." *Drones* 3 (3): 61. https://0-doi-org.libus.csd.mu.edu/10.3390/drones3030061.

Badrinarayanan, V., A. Kendall, and R. Cipolla. 2017. "SegNet: A deep convolutional encoder-decoder architecture for image segmentation." *IEEE Trans. Pattern Anal. Mach. Intell.* 39 (12): 2481–2495. https://0-doi-org.libus.csd.mu.edu/10.1109/TPAMI.2016.2644615.

Buscombe, D., and A. C. Ritchie. 2018. “Landscape classification with deep neural networks.” *Geosciences* 8 (7): 244. https://0-doi-org.libus.csd.mu.edu/10.3390/geosciences8070244.

Chollet, F. 2015. “Keras: The python deep learning library.” Accessed August 7, 2019. https://keras.io/.

Cunliffe, A. M., R. E. Brazier, and K. Anderson. 2016. “Ultra-fine grain landscape-scale quantification of dryland vegetation structure with drone-acquired structure-from-motion photogrammetry.” *Remote Sens. Environ.* 183 (Sep): 129–143. https://0-doi-org.libus.csd.mu.edu/10.1016/j.rse.2016.05.019.

Deng, J., W. Dong, R. Socher, L. Li, K. Li, and F. Li. 2009. “ImageNet: A large-scale hierarchical image database.” In Proc., 2009 IEEE Conf. on Computer Vision and Pattern Recognition, 248–255. New York: IEEE.

Dettmers, T. 2015. “Deep learning in a nutshell: Core concepts.” Accessed August 7, 2019. https://devblogs.nvidia.com/deep-learning-nutshell-core-concepts/.

Du, J. C., and H. C. Teng. 2007. “3D laser scanning and GPS technology for landslide earthwork volume estimation.” *Autom. Constr.* 16 (5): 657–663. https://0-doi-org.libus.csd.mu.edu/10.1016/j.autcon.2006.11.002.

Engelcke, M., D. Rao, D. Z. Wang, T. Chi Hay, and I. Posner. 2017. “Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks.” In Proc., 2017 IEEE Int. Conf. on Robotics and Automation (ICRA), 1355–1361. New York: IEEE.

Geirhos, R., P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. 2019. “ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness.” In Proc., 7th Int. Conf. on Learning Representations (ICLR 2019). Cambridge, MA: International Conference on Learning Representations. https://openreview.net/forum?id=Bygh9j09KX.

Han, J., D. Zhang, G. Cheng, L. Guo, and J. Ren. 2015. “Object detection in optical remote sensing images based on weakly supervised learning and high-level feature learning.” *IEEE Trans. Geosci. Remote Sens.* 53 (6): 3325–3337. https://0-doi-org.libus.csd.mu.edu/10.1109/TGRS.2014.2374218.

Jiang, Y., and Y. Bai. 2020a. “Determination of construction site elevations using drone technology.” In Proc., Construction Research Congress 2020. Reston, VA: ASCE.

Jiang, Y., and Y. Bai. 2020b. “Estimation of construction site elevations using drone-based orthoimagery and deep learning.” *J. Constr. Eng. Manage.* 146 (8): 04020086. https://0-doi-org.libus.csd.mu.edu/10.1061/(ASCE)CO.1943-7862.0001869.

Kang, K., H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, and W. Ouyang. 2018. “T-CNN: Tubelets with convolutional neural networks for object detection from videos.” *IEEE Trans. Circuits Syst. Video Technol.* 28 (10): 2896–2907. https://0-doi-org.libus.csd.mu.edu/10.1109/TCSVT.2017.2736553.

Kim, S., and J. S. Russell. 2003. “Framework for an intelligent earthwork system: Part I. System architecture.” *Autom. Constr.* 12 (1): 1–13. https://0-doi-org.libus.csd.mu.edu/10.1016/S0926-5805(02)00034-1.

Kussul, N., M. Lavreniuk, S. Skakun, and A. Shelestov. 2017. “Deep learning classification of land cover and crop types using remote sensing data.” *IEEE Geosci. Remote Sens. Lett.* 14 (5): 778–782. https://0-doi-org.libus.csd.mu.edu/10.1109/LGRS.2017.2681128.

Kwon, S., J. W. Park, D. Moon, S. Jung, and H. Park. 2017. “Smart merging method for hybrid point cloud data using UAV and LIDAR in earthwork construction.” *Procedia Eng.* 196 (Jan): 21–28. https://0-doi-org.libus.csd.mu.edu/10.1016/j.proeng.2017.07.168.

Liu, T., and A. Abd-Elrahman. 2018. “Deep convolutional neural network training enrichment using multi-view object-based analysis of Unmanned Aerial systems imagery for wetlands classification.” *ISPRS J. Photogramm. Remote Sens.* 139 (May): 154–170. https://0-doi-org.libus.csd.mu.edu/10.1016/j.isprsjprs.2018.03.006.

Liu, T., A. Abd-Elrahman, J. Morton, and V. L. Wilhelm. 2018. “Comparing fully convolutional networks, random forest, support vector machine, and patch-based deep convolutional neural networks for object-based wetland mapping using images from small unmanned aircraft system.” *GISci. Remote Sens.* 55 (2): 243–264. https://0-doi-org.libus.csd.mu.edu/10.1080/15481603.2018.1426091.

Maggiori, E., Y. Tarabalka, G. Charpiat, and P. Alliez. 2016. “Fully convolutional neural networks for remote sensing image classification.” In Proc., 2016 IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS), 5071–5074. New York: IEEE.

Mao, X. J., C. Shen, and Y. B. Yang. 2016. “Image restoration using convolutional auto-encoders with symmetric skip connections.” Preprint, submitted August 7, 2019. http://arxiv.org/abs/1606.08921.

Nair, V., and G. Hinton. 2010. “Rectified linear units improve restricted boltzmann machines.” In Proc., 27th Int. Conf. on Machine Learning (ICML 2010), 807–814. Haifa, Israel: International Machine Learning Society.

Nassar, K., and Y. Jung. 2012. “Structure-from-motion approach to the reconstruction of surfaces for earthwork planning.” *J. Constr. Eng. Project Manage.* 2 (3): 1–7. https://0-doi-org.libus.csd.mu.edu/10.6106/JCEPM.2012.2.3.001.

Noh, H., S. Hong, and B. Han. 2015. “Learning deconvolution network for semantic segmentation.” In Proc., 2015 IEEE Int. Conf. on Computer Vision (ICCV 2015), 1520–1528. New York: IEEE.

Ronneberger, O., P. Fischer, and T. Brox. 2015. “U-Net: Convolutional networks for biomedical image segmentation.” In Proc., Medical Image Computing and Computer-Assisted Intervention (MICCAI 2015), 234–241. Cham, Switzerland: Springer.

Sandler, M., A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen. 2018. “Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation.” Prepint, submitted April 7, 2020. http://arxiv.org/abs/1801.04381.

Schneider, S., G. W. Taylor, and S. Kremer. 2018. “Deep learning object detection methods for ecological camera trap data.” In Proc., 2018 15th Conf. on Computer and Robot Vision (CRV), 321–328. New York: IEEE.

Shelhamer, E., J. Long, and T. Darrell. 2017. “Fully convolutional networks for semantic segmentation.” *IEEE Trans. Pattern Anal. Mach. Intell.* 39 (4): 640–651. https://0-doi-org.libus.csd.mu.edu/10.1109/TPAMI.2016.2572683.

Siebert, S., and J. Teizer. 2014. “Mobile 3D mapping for surveying earthwork projects using an Unmanned Aerial Vehicle (UAV) system.” *Autom. Constr.* 41 (May): 1–14. https://0-doi-org.libus.csd.mu.edu/10.1016/j.autcon.2014.01.004.

Sung, C., and P. Y. Kim. 2016. “3D terrain reconstruction of construction sites using a stereo camera.” *Autom. Constr.* 64 (Apr): 65–77. https://0-doi-org.libus.csd.mu.edu/10.1016/j.autcon.2015.12.022.

Theodorus, A., M. Nauta, and C. Seifert. 2020. “Evaluating CNN interpretability on sketch classification.” In Proc., 12th Int. Conf. on Machine Vision (ICMV 2019). Bellingham, WA: Society of Photo-Optical Instrumentation Engineers. https://0-doi-org.libus.csd.mu.edu/10.1117/12.2559536.

Westoby, M. J., J. Brasington, N. F. Glasser, M. J. Hambrey, and J. M. Reynolds. 2012. “‘Structure-from-Motion’ photogrammetry: A low-cost, effective tool for geoscience applications.” *Geomorphology* 179 (Dec): 300–314. https://0-doi-org.libus.csd.mu.edu/10.1016/j.geomorph.2012.08.021.

Zhao, H., J. Shi, X. Qi, X. Wang, and J. Jia. 2017. “Pyramid scene parsing network.” In Proc., 2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 6230–6239. New York: IEEE.

Figures

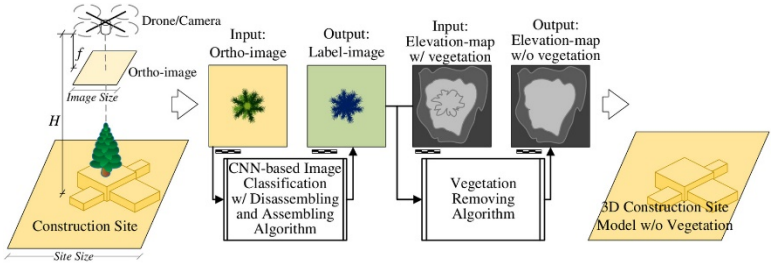

Fig. 1. Overall workflow of the proposed framework.

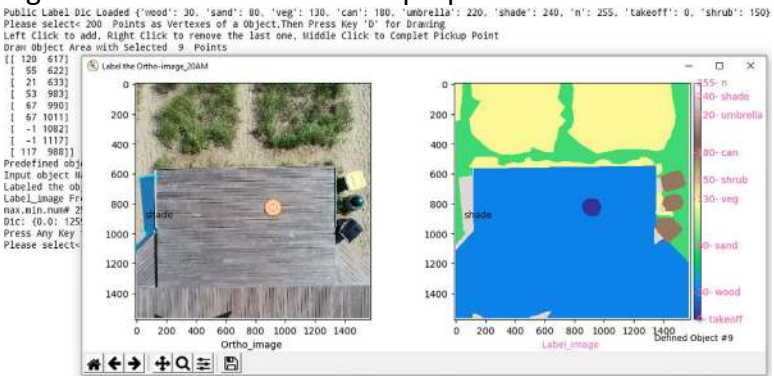

Fig. 2. Label-image crafting example. (Image by Yuhan Jiang.)

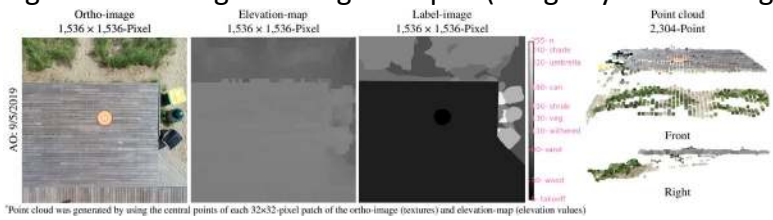

Fig. 3. Pair of orthoimage, label-image, and elevation-map dataset.

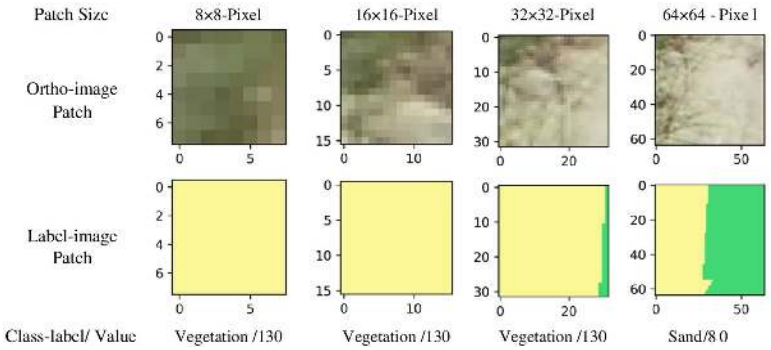

Fig. 4. Small-patch examples.

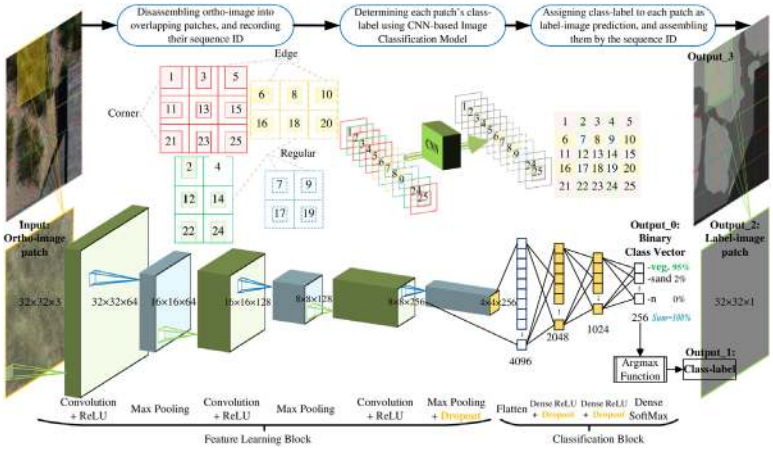

Fig. 5. CNN-based image classification model with $32 \times 32 - pixel$ patch.

```
SEARCH_VEG_REPLACE_GROUND(row_index, col_index, ele_map, label_image, ortho_img, stride, qsize, in_row_bool)
Initial  win_size_max = MIN(image_Width, image_Height)/2, ratio_q = 8, counter_min = qsize × qsize × ratio_q
if label_image[row_index, col_index] in veg_label_list:
  for win_size in range(0, win_size_max, stride) #search in adjustable window [row_low_b: row_up_b, col_low_b: col_up_b]
    row_low_b = row_index − qsize × ratio_q/(2 − in_row_bool) − win_size × (1 − in_row_bool)# in Col. -loop, extend the windows in the Col. direction only
    col_low_b = col_index − qsize × ratio_q/(1 + in_row_bool) − win_size × (in_row_bool)      # in Row - loop, extend the windows in the Row direction only
    row_up_b  = row_index + stride + qsize × ratio_q/(2 − in_row_bool) + win_size × (1 − in_row_bool)# ratio_q/2 narrows initial window in image_Height/Col. direction
    col_up_b  = col_index + stride + qsize × ratio_q/(1 + in_row_bool) + win_size × (in_row_bool)     # ratio_q/2 narrows initial window in image_Width/Row direction
    label_image_patch = label_image[row_low_b: row_up_b, col_low_b: col_up_b]
    ele_map_patch     =     ele_map[row_low_b: row_up_b, col_low_b: col_up_b]
    ground_index = (label_image_patch == ground_label)#return the indexes of ground_label in the label_image_patch
    if SUM(ground_index) >= counter_min#return the num. of all True elements in ground_index
      # if the num. of ground_label in the windows is enough, then remove current vegetation patch [row_index: row_index + stride, col_index: col_index + stride]
      ground_ele = MEAN(ele_map_patch[ground_index])#return the average value of neigbor ground blocks' elevations
      ele_map[row_index: row_index + stride, col_index: col_index + stride] = ground_ele # alternative option: update elevation only when ele_map[i, s] > ground_ele
      label_image[row_index: row_index + stride, col_index: col_index + stride] = ground_label
      ortho_img[row_index: row_index + stride, col_index: col_index + stride] = (255,125,255)# mark the ortho_image
      break
```

```
VEG_REMOVING_IN_ROW_THEN_COL_TRAVERSE(ele_map, label_image, ortho_img, qszie)
Initial  row, col = 0,0; image_Width, image_Height = ele_map. shape; H, W = 0,0; i = 0 ; stride = qsize/4 # traversal stride
while i < MIN (image_Height/stride + image_Width/stride) # max num. of Row & Col − loops
  while col + W <= image_Width:
    if col + W < image_Width#process Row − loop
      row_index = row + H;  col_index = col + W
      SEARCH_VEG_REPLACE_GROUND(row_index, col_index, ele_map, label_image, ortho_img, stride, qsize, in_row_bool = True)# remove in Row
      W = W + stride # move to next element in the Row
    else#completed the Row, move to Col − loop
      W = 0;   H = stride # skip the already processed 1st element in the Row − loop
      while row + H < image_Height: #process Col − loop
        row_index = row + H;  col_index = col + W
        SEARCH_VEG_REPLACE_GROUND(row_index, col_index, ele_map, label_image, ortho_img, stride, qsize, in_row_bool = False)# remove in Col.
        H = H + stride # to next element in the Col
      row = row + stride # move to the next row
      H = 0
      break# complete the current col
  col = col + stride # move to the next col
  i = i + 1
```

Fig. 6. Vegetation-removing algorithm.

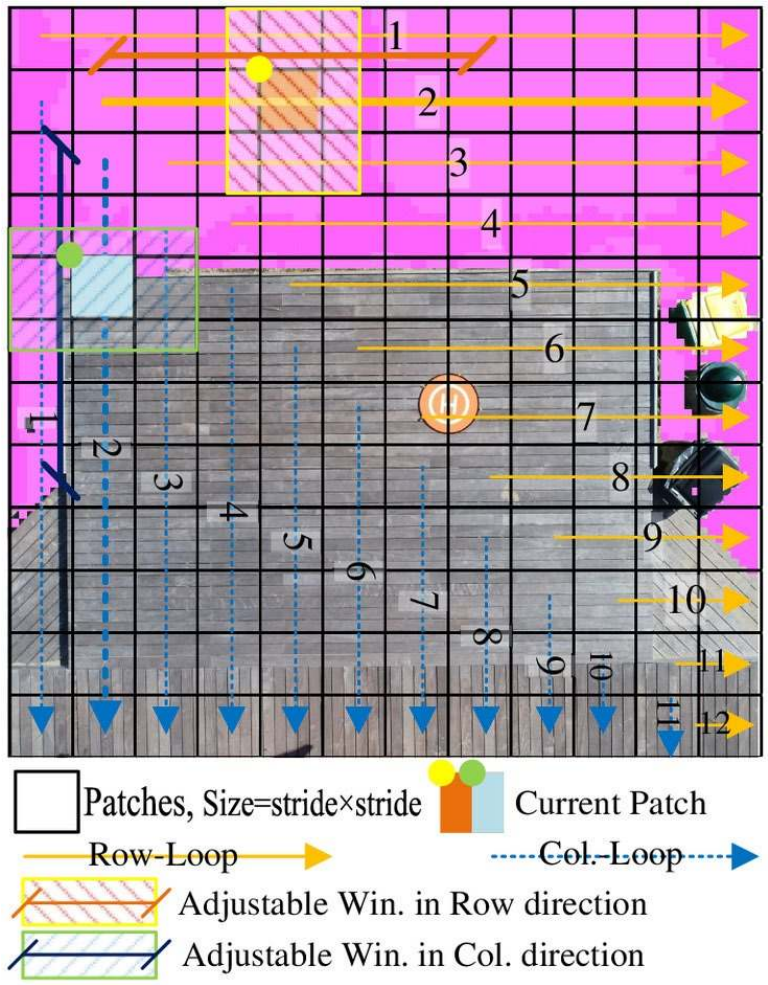

Fig. 7. Row-column-row-loop for traversing a label-image.

Fig. 8. Experimental site. (Image by Yuhan Jiang.)

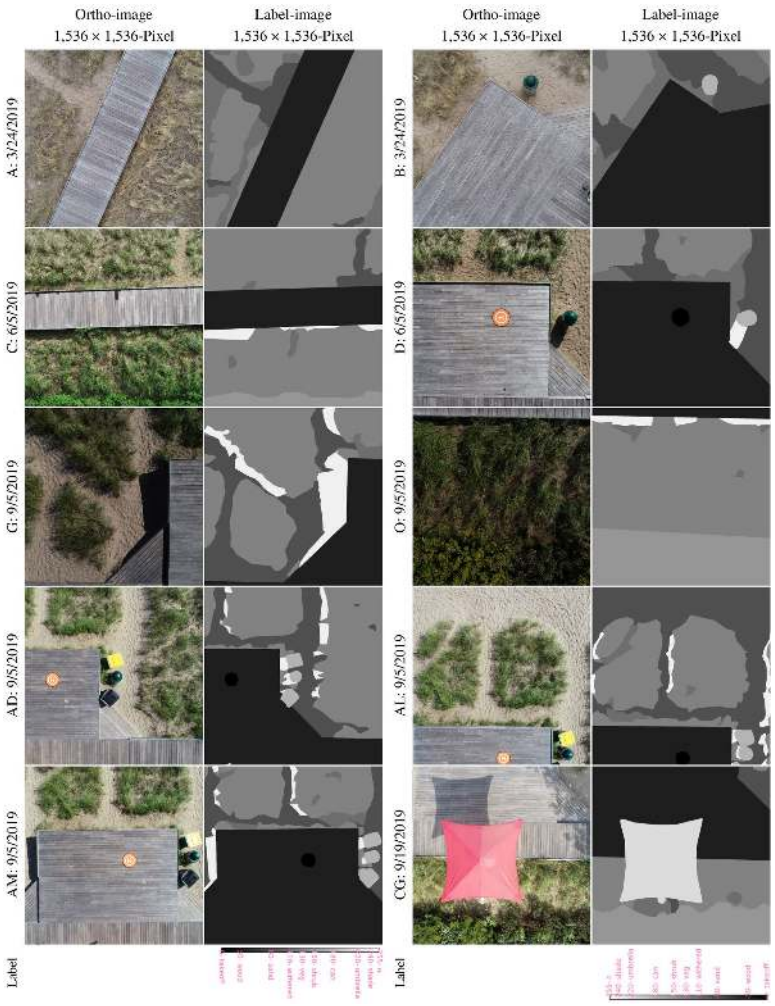

Fig. 9. Training and validation datasets.

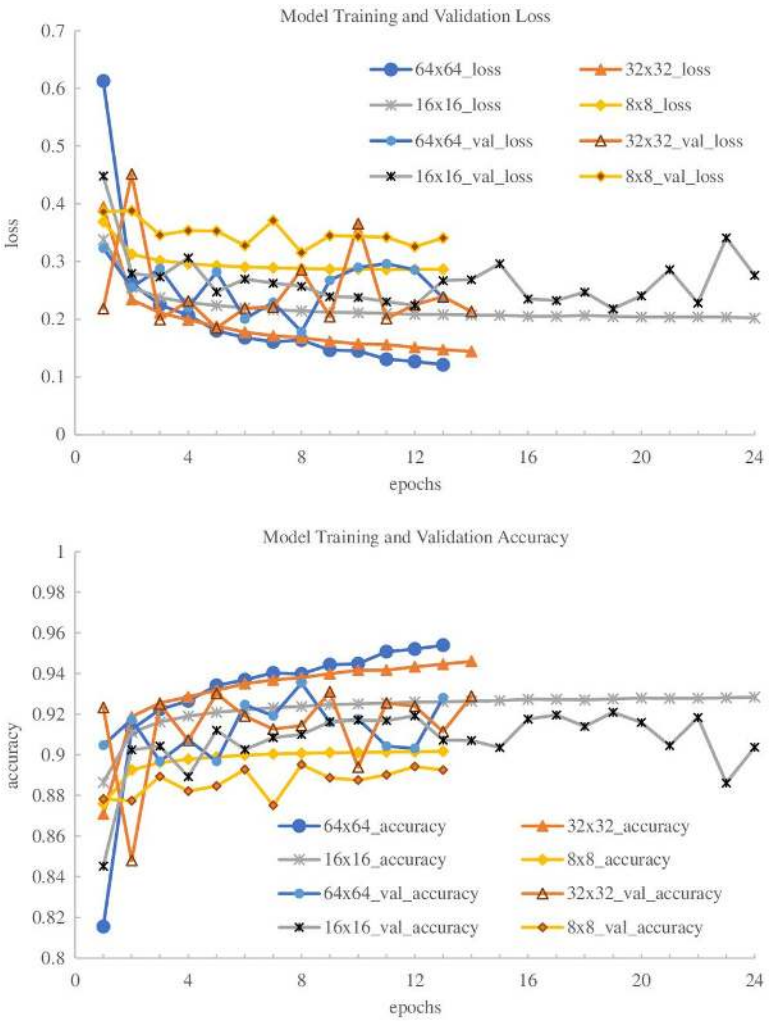

Fig. 10. Training and validation loss and accuracy of early stopping trials.

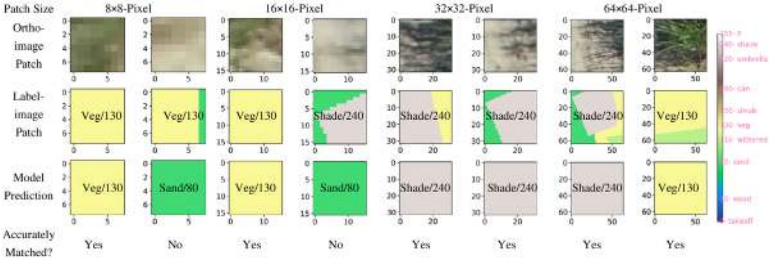

Fig. 11. Small-patch predictions of early stopping trials.

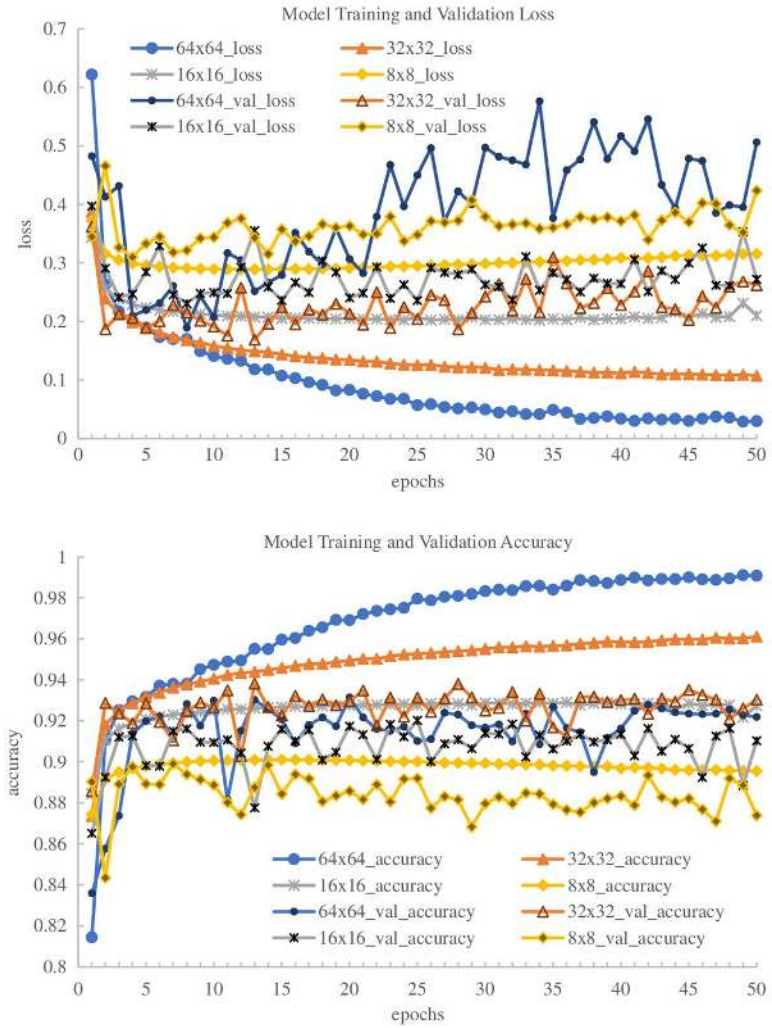

Fig. 12. Training and validation loss and accuracy of 50-epoch trials.

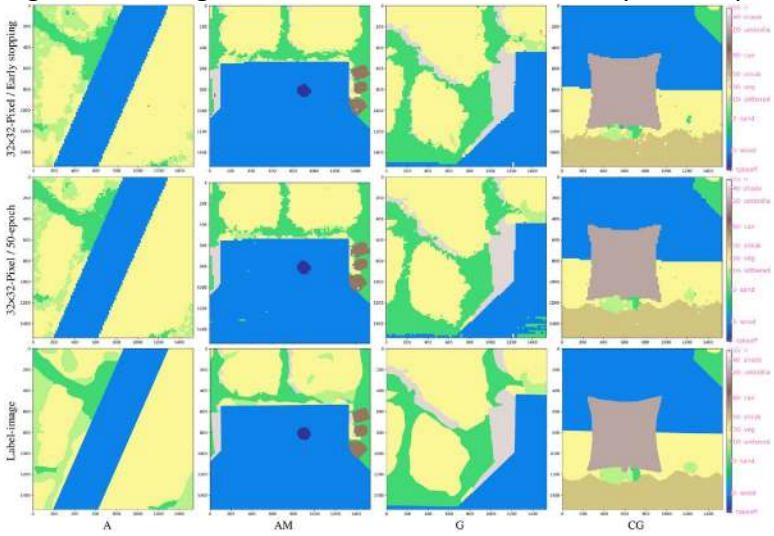

Fig. 13. Patch-wise predictions.

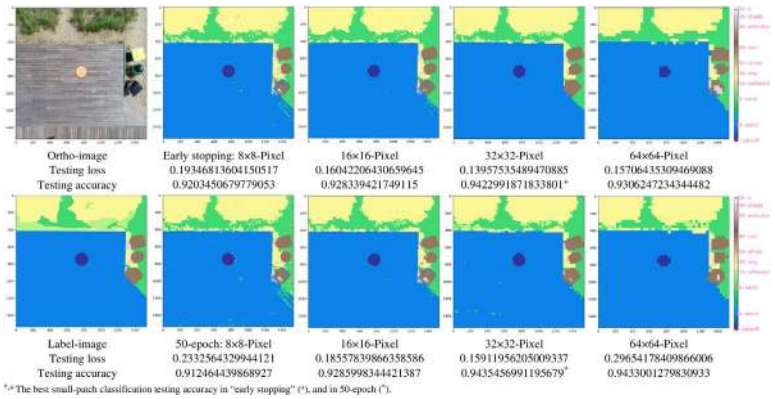

Fig. 14. Testing results.

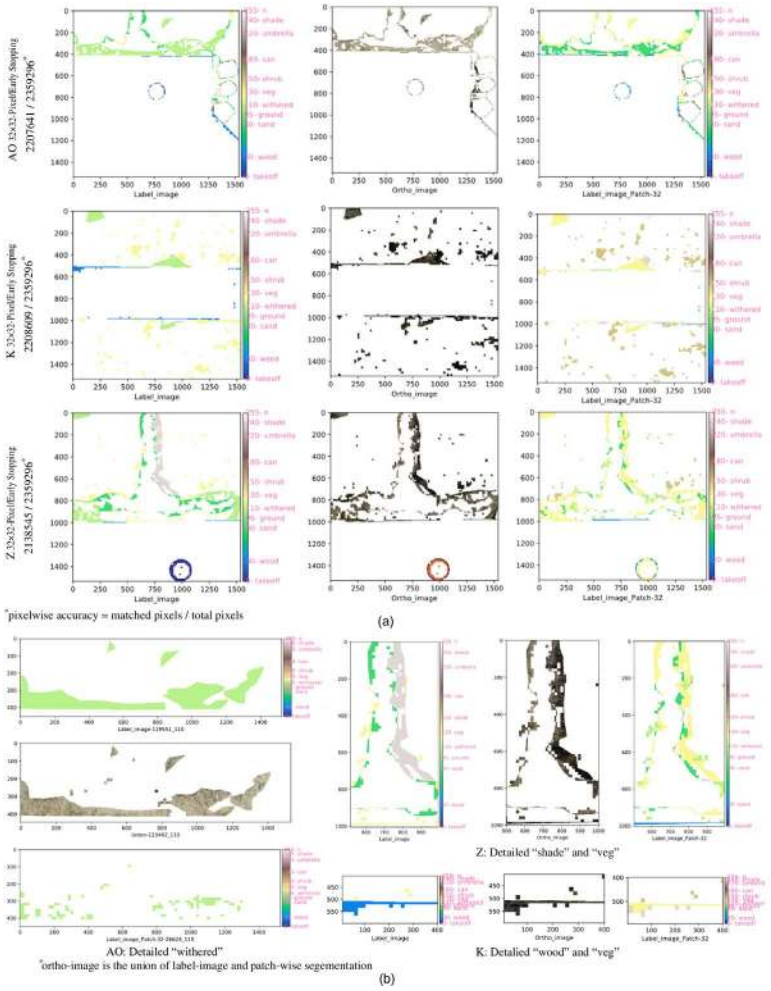

Fig. 15. (a) Mapped pixel prediction errors; (b) detailed prediction comparisons; and (c) detailed label errors.

Fig. 16. Vegetation identifying results.

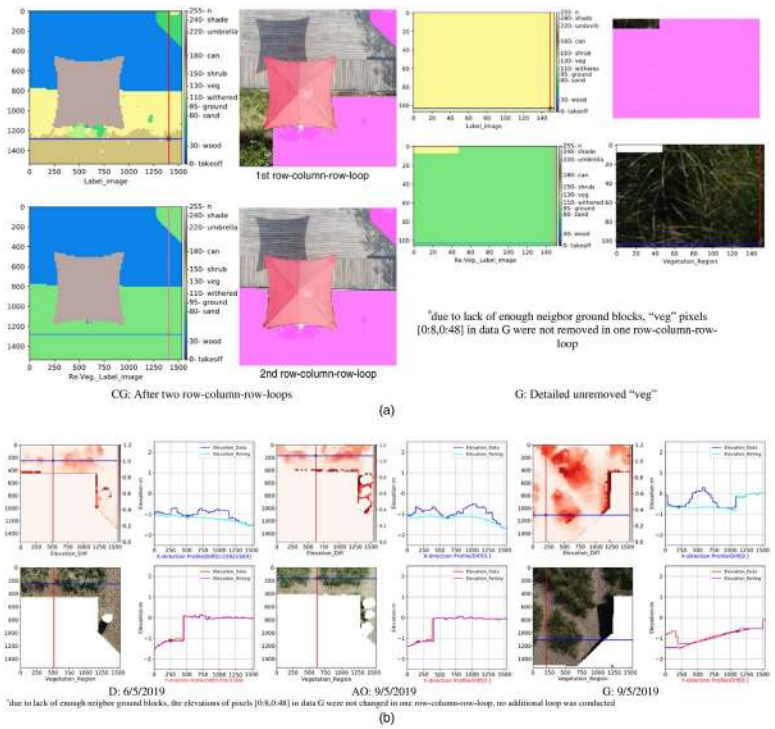

Fig. 17. (a) Vegetation removing in label-image; and (b) vegetation removing in elevation-map.

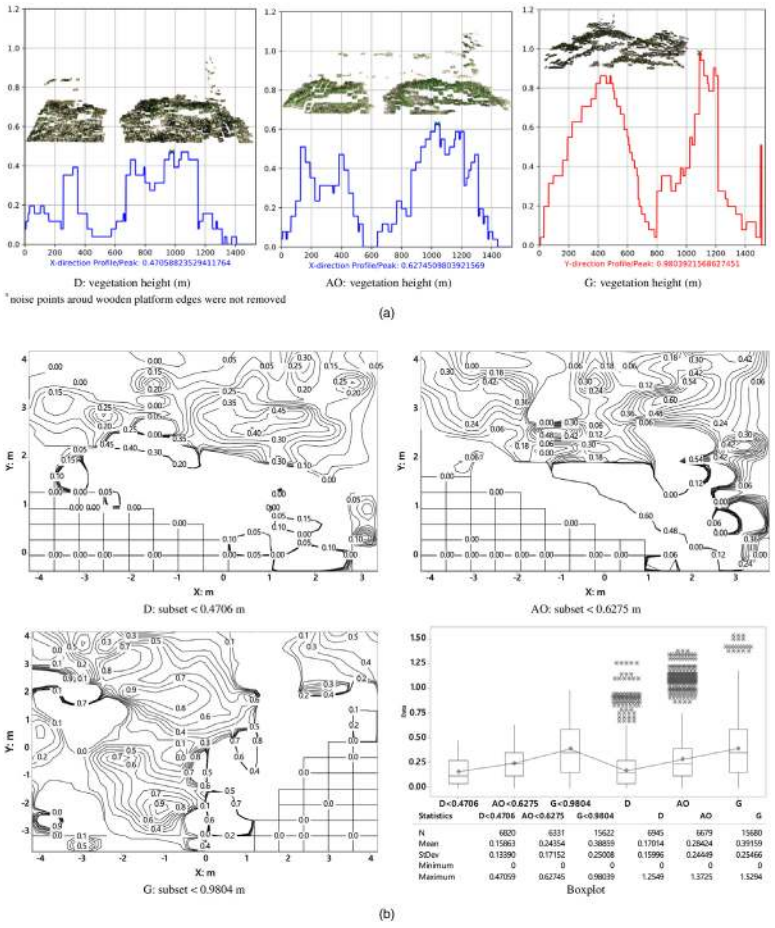

Fig. 18. (a) Vegetation height in profile; and (b) vegetation height in contour plot.

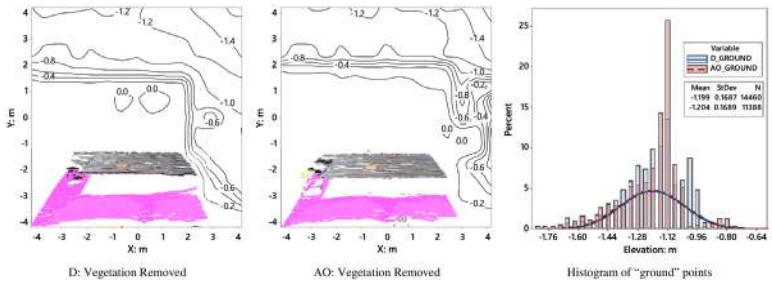

Fig. 19. Estimated ground elevation comparisons.

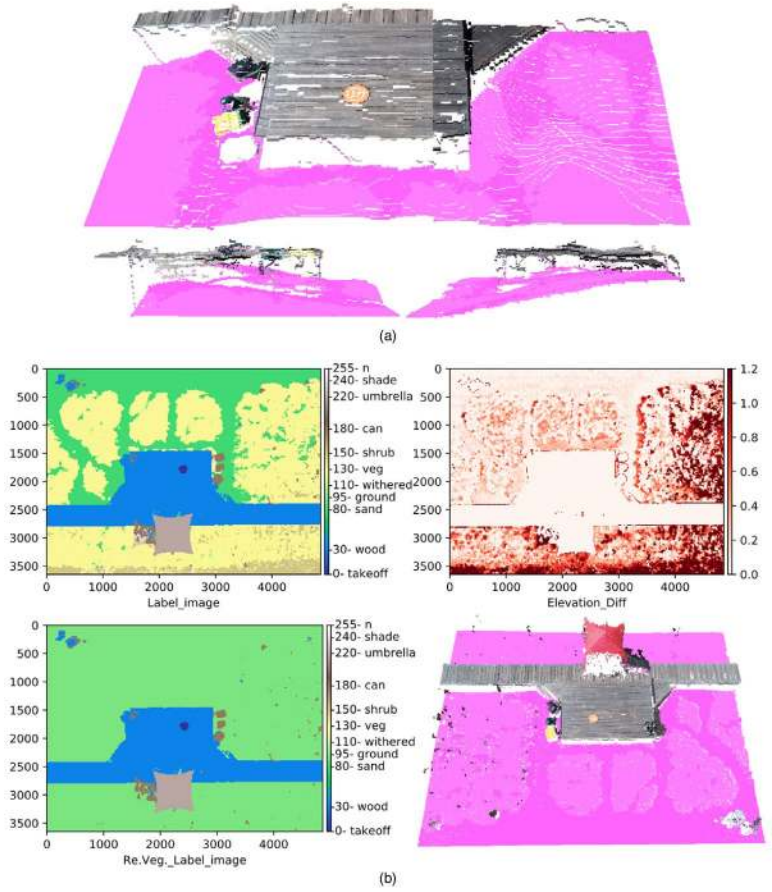

Fig. 20. (a) Results stitching demo; and (b) large-scale demo.