8-2020

Estimation of Construction Site Elevations Using Drone-Based Orthoimagery and Deep Learning

Yuhan Jiang
Marquette University

Yong Bai
Marquette University, yong.bai@marquette.edu

Journal of Construction Engineering and Management, Vol. 146, No. 8 (August 2020). DOI.

Estimation of Construction Site Elevations Using Drone-Based Orthoimagery and Deep Learning

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

Abstract

Using deep learning to recover depth information from a single image has been studied in many situations, but there are no published articles related to the determination of construction site elevations. This paper presents the research results of developing and testing a deep learning model for estimating construction site elevations using a drone-based orthoimage. The proposed method includes an orthoimage-based convolutional neural network (CNN) encoder, an elevation map CNN decoder, and an overlapping orthoimage disassembling and elevation map assembling algorithm. In the convolutional encoder-decoder network model, the max pooling and up-sampling layers link the orthoimage pixel and elevation map pixel in the same coordinate. The experiment data sets are eight orthoimage and elevation map pairs ($1{,}536 \times 1{,}536$ pixels), which are cropped into 64,800

patch pairs (128×128 pixels). Experimental results indicated that the 128×128-pixel patch had the best model prediction performance. After 100 training epochs, 21.22% of the selected 2,304 points from the testing data set were exactly matched with their ground truth elevation values; and 52.43% points were accurately matched in ± 5 cm and 66.15% points in ± 10 cm, less than 10% points exceeded ± 25 cm. This research project advanced drone applications in construction, evaluated CNNs' effectiveness in site surveying, and strengthened CNNs to work with large-scale construction site images.

Introduction

The determination of elevations on construction sites is accomplished by contact surveying methods using total station; a global positioning system (GPS); a level, theodolite, and noncontact methods using a laser scanner (Du and Teng 2007; Kwon et al. 2017); drone photogrammetry (Nassar and Jung 2012; Siebert and Teizer 2014); and a stereo camera (Sung and Kim 2016). These methods are either based on the equipment's physical properties or based on the camera model and epipolar geometry as reference information to find the geometry data, so the models can achieve state-of-the-art results. However, their weaknesses are noticeable because contact methods have a time-consuming outdoor procedure and a high probability of interfering with other construction operations. Noncontact methods produce a huge amount of unfiltered target points and need more time to process the raw data. A previous study stated that the duration for estimating on-site soil volume after drone photogrammetry is one processing day under the following conditions: the point cloud is generated by Agisoft PhotoScan, the geometry model is created by Autodesk ReCap with the point cloud, and the soil volume is estimated using Autodesk Civil 3D (Haur et al. 2018). Quickly and accurately estimating elevations of a construction site in real time is still a challenge.

Previous research has shown the feasibility of using deep learning methods to recover the relative depth information for each pixel of an image of indoor scenes (Eigen et al 2014; Liu et al. 2015; Laina et al. 2016), outdoor scenes (Chen et al. 2016; Li and Snavely 2018), and scenes from automatic driving applications (Garg et al. 2016). In addition, convolutional neural networks (CNNs) have been verified as effective and reliable in microscale scenes, such as estimating the surface height map from a single image of a foam mat and mouse pad (Zhou et al. 2017). However, a challenge of training a deep learning model is acquiring a data set. In the automatic drive application, the CNN model is trained to determine the object's distance from a single forward-facing view of a car; the data set, depth, and front-view image pair are created using a stereo camera system that is installed on the front of the car (Garg et al. 2016). Another interesting approach to create a labeled data set is using artificial images, such as generating different view images from a photogrammetry-based, concrete mixer truck, and three-dimensional (3D) model, and using these images to train a construction equipment object detector (Kim and Kim 2018).

The researchers' recent work (Jiang and Bai 2020) presents a two-frame, image-based 3D-reconstruction method that utilizes drone technology to capture a low-high orthoimage pair for assembling a vertical-baseline stereo vision model. The developed method can be used to calculate the construction site elevation values from the stereo vision model and saves them as grayscale values in an elevation map. The elevation map is similar to a depth map (Eigen et al 2014; Garg et al. 2016), which stores the elevation value in each pixel of a grayscale image as its grayscale value. The drone-based orthoimage, also called a top/plan view or orthophoto (Siebert and Teizer 2014), is captured when the camera's principal ray is perpendicular to the construction site surface. In the case of automatic driving, the forward-facing view has the camera's principal ray perpendicular to the objects in front of the car. So, the images captured in front of automatic driving cars and above construction site surfaces have the common characteristic in that the objects in the same depth level/elevation level have common texture features as in the forward-facing view/orthoimage (Fig. 1). Therefore, capturing an orthoimage

over a construction site by drone, then using this image to estimate the site elevations is a feasible approach that will reduce drone flying time and avoid hazards of drone crashes on the construction site.

In this paper, a deep learning–based method, convolutional encoder-decoder network model is proposed to estimate elevations from the orthoimages of a construction site, which links each pixel of the orthoimage with the same coordinate pixel in the elevation map (see Fig. 1). This research evaluates the effectiveness of the single image-based 3D-reconstruction method, which requires much fewer images in estimating elevation than the multiple images-based methods, such as the structure from motion (SfM) method (Nassar and Jung 2012). To explain how to estimate site elevations from a drone-based orthoimage, the rest of this paper presents the data set acquisition, model designs, training and testing, field experiment, and result discussions.

Proposed Method for Estimating Construction Site Elevations

Drone-Based Orthoimage Acquisition

An orthoimage of a construction site can be captured by yielding the camera gimbal to negative 90°, keeping the camera lens facing the ground. In this paper, the drone, DJI Phantom 4 Pro V2.0, is designed to fly at 10 and 20 m over the takeoff location. As the focal length is fixed in the camera pinhole model (Fig. 2), the size of an object in the orthoimage (*Image@H*) has a negative relationship with the drone flight height (H). The captured orthoimages have the following measurements: image sizes $= 3{,}648 \times 4{,}864$ pixels; ground sample distance $(\mathrm{GSD}) = 0.27$ cm/pixel; site size $= 9.85 \times 13.13$ m^2 m2 with $H = 10$ m; and $\mathrm{GSD} = 0.54$ cm/pixel, site size $= 19.70 \times 26.26$ m^2 with $H = 20$ m.

Elevation Data Acquisition

The elevation data can be acquired by either contact or noncontact methods listed in the “Introduction” section, but the challenge is linking the orthoimage’s pixels with the elevation data. A possible approach is to store the elevation data in an equal-size, 8-bit, grayscale image, referred to as an elevation map, which uses 0 as the elevation lower boundary (−5 m) and 255 as the elevation upper boundary (5 m). In an elevation map, the grayscale value of each pixel can be easily converted from range [0, 255] to its corresponding real elevation value. In Fig. 3, the elevation map is represented in viridis colormap for better visualization, and the X/Y profiles show the elevation changes at the selected point (in meters). Acquiring elevation data for each pixel of the orthoimage is unreasonable. To save time, the simplified approach is to share the same grayscale value/elevation value for a patch, such as a 32×32-pixel patch. For example, the first selected pixel (16,16) shares its grayscale value/elevation value with the patch Elevation map [0:31,0:31].

Data Set Creation

An RGB orthoimage acquired by the drone has three channels. Considering that the texture color is important in distinguishing between different objects on the construction site, the texture information is kept, rather than using a grayscale image. Thus, the proposed deep learning model has different input and output data channels, such as input shape (128,128,3) and output shape (128,128,1). Considering the computing capacity of the workstation system, in Fig. 4 the first through fifth columns list the possible model input and output data set examples of 32×32-pixel, 64×64-pixel, 128×128-pixel, 256×256-pixel, and 512×512-pixel patches, which are cropped from [0:31,0:31], [0:63,0:63], [0:127,0:127], [0:255,0:255], and [0:511,0:511] of the $1{,}536 \times 1{,}536$-pixel orthoimage and elevation map (the sixth column), respectively. For the elevation map patches, each larger patch contains four times more elevation values than the smaller patch, such as the 64×64-pixel patch contains elevation values from pixel (16,16), pixel (16,48), pixel (48,16), and pixel (48,48), while the 32×32-pixel patch only contains the elevation value from pixel (16,16). Thus, a smaller patch size is better for the deep learning model to learn the local features from the input and output data set. Conversely, a larger patch size is better for learning the global features from the input and output data sets.

Therefore, for this paper, experiments were conducted to evaluate the performances of the different patch size configurations, which include 32×32-pixel, 64×64-pixel, 128×128-pixel, 256×256-pixel, and 512×512-pixel patches. When creating these patches, the stride was set as 16, 32, 64, or 96 pixels for moving these square boxes on the orthoimage and elevation map (larger strides were used to avoid workstation system memory shortages), and the number of patches were determined using Eq. (1), where "[ ]" is the floor function. Moreover, to make the deep learning model robust in different image orientations, the orthoimage and elevation map were planned to rotate 90°, 180°, and 270° to increase the data set by four times. Table 1 lists the detailed parameters in creating data sets from an orthoimage and elevation map pair with $1{,}536 \times 1{,}536$ pixels

(1)

$$\text{Num. of Data Set} = \left\lfloor \frac{\text{Image Height} - \text{Patch Size}}{\text{Stride}} + 1 \right\rfloor \times \left\lfloor \frac{\text{Image Width} - \text{Patch Size}}{\text{Stride}} + 1 \right\rfloor \times 4$$

Table 1. Data set parameters

Patch sizes	Strides	Rows	Columns	Number	Number after four rotations
32×32	16	95	95	9,025	36,100
64×64	32	47	47	2,209	8,836
128×128	32	45	45	2,025	8,100
256×256	64	21	21	441	1,764
512×512	96	11	11	121	484

Network Architecture Setup

The proposed deep learning model is a convolutional encoder-decoder network model (Fig. 5). In the encoder, the five convolution layers learn the model input orthoimage patch as feature maps; each convolution layer contains a 2D convolution operation with zero-padding [see Fig. 6(a)], and the layer output has the same size as the layer input (Chollet 2015); each max pooling layer next to the convolution layer is a max pooling operation [Fig. 6(b)], which reduces the layer input (convolution layer output) to one-half the size as the layer output. In the decoder, the five convolution layers interpret the feature maps to model an elevation map output; each convolution layer contains a 2D convolution operation with zero padding as well. The up-sampling layers are the reverse operations of max pooling operations, which enlarge the layer input to its double size as the layer output [Fig. 6(b)]. In detail, Fig. 6(a) shows an example of a zero-padded convolution operation. The original input is 5×5 in size, which has been padded to 7×7; the 3×3 kernel convolution has a 5×5 output, which has the same as the original input. If the original input is not padded with zero, the convolution output is the filled 3×3 region only. Fig. 6(b) shows an example of how max pooling (2×2 filter and $\text{strides} = 2$) and up-sampling work in the model.

The proposed convolutional encoder-decoder network model has an equal number of max pooling layers and up-sampling layers. This type of model is referred to as an hourglass-like model, which has been widely used in image segmentation, such as SegNet (Badrinarayanan et al. 2017). Another hourglass-like model uses deconvolution network in the decoder, such as DeconvNet (Noh et al. 2015), where each deconvolution (also known as transposed convolution) layer is the opposite operation of normal convolution (Chollet 2015). In this research study, the convolutional decoder and deconvolutional decoder were compared and their generated results do not have any significant difference. Additionally, the proposed model is different from SegNet, in

which the up-sampling layer is the first layer in the decoder, but the proposed model uses a convolution layer first (Fig. 5).

To make this encoder-decoder model able to interpret an orthoimage patch and predict an equivalent-sized elevation map patch, the intersection part of the encoder-decoder is proposed as a 512-channel feature map, which is generated from the “max_pooling2d_5” layer (Table 2). For example, the encoder generates a $4 \times 4 \times 512$ feature map for the $128 \times 128 \times 3$ input (Fig. 5). This intermedia feature map is required by the model output. Based on the data set creation, an elevation map shares a common integer value from 0 to 255 in each 32 × 32 small-patch; thus, a 128 × 128 elevation map patch contains 16 (4 × 4) elevation values; if each feature-map just represents the probability of the integer value from 0 to 255, then at least a 4 × 4 × 256 feature map is required for the decoder. The proposed “conv2d_5” layer uses 512 filters (Table 2), which doubles the required channel number. The 512-channel feature map can be understood as each channel is the probability of the element in the list [0.0, 0.5, 1.0, …, 245.5, 255.0]. In this research study, adding the interaction feature map to 1,024 channels was no different from the 512-channel. In addition, the five max-pooling layer is the maximum number for the encoder because the smallest model input 32×32-pixel patch is transformed to a 1×1-pixel feature map after five max pooling operations.

Table 2. Model layers parameters

	Model architecture for 32 × 32, 64 × 64, 128 × 128, 256 × 256, and 512 × 512-pixel patches						Output shapes for each patch					
							32	64	128	256	512	
Blocks	Layers (type and kernel size)	Strides	Padding	Activations	Filters/ channels	Parameters number	Rows/columns					Channels
Input	input_1 (input layer)	—	—	—	3	0	32	64	128	256	512	3
Encoder	conv2d_1 (conv2D 3 × 3)	1	Same	ReLU	64	1,792	32	64	128	256	512	64
	max_pooling2d_1 (max pooling 2 × 2)	2	Same	—	—	0	16	32	64	128	256	64
	conv2d_2 (conv2D 3 × 3)	1	Same	ReLU	128	73,856	16	32	64	128	256	128
	max_pooling2d_2 (max pooling 2 × 2)	2	Same	—	—	0	8	16	32	64	128	128
	conv2d_3 (conv2D 3 × 3)	1	Same	ReLU	256	295,168	8	16	32	64	128	256
	max_pooling2d_3 (max pooling 2 × 2)	2	Same	—	—	0,	4	8	16	32	64	256
	conv2d_4 (conv2D 3 × 3)	1	Same	ReLU	512	1180160	4	8	16	32	64	512
	max_pooling2d_4 (max pooling 2 × 2)	2	Same	—	—	0	2	4	8	16	32	512
	conv2d_5 (conv2D 3 × 3)	1	Same	ReLU	512	235,9808	2	4	8	16	32	512
	max_pooling2d_5 (max pooling 2 × 2)	2	Same	—	—	0	1	2	4	8	16	512
Decoder	conv2d_6 (conv2D 3 × 3)	1	Same	ReLU	512	235,9808	1	2	4	8	16	512
	up_sampling2d_1 (up sampling 2 × 2)	1	—	—	—	0	2	4	8	16	32	512
	conv2d_7 (conv2D 3 × 3)	1	Same	ReLU	512	235,9808	2	4	8	16	32	512
	up_sampling2d_2 (up sampling 2 × 2)	1	—	—	—	0	4	8	16	32	64	512
	conv2d_8 (conv2D 3 × 3)	1	Same	ReLU	256	117,9904	4	8	16	32	64	256
	up_sampling2d_3 (up sampling 2 × 2)	1	—	—	—	0	8	16	32	64	128	256
	conv2d_9 (conv2D 3 × 3)	1	Same	ReLU	128	295,040	8	16	32	64	128	128
	up_sampling2d_4 (up sampling 2 × 2)	1	—	—	—	0	16	32	64	128	256	128
	conv2d_10 (conv2D 3 × 3)	1	Same	ReLU	64	73,792	16	32	64	128	256	64
	up_sampling2d_5 (up sampling 2 × 2)	1	—	—	—	0	32	64	128	256	512	64
Output	conv2d_11 (conv2D 3 × 3)	1	Same	Sigmoid	1	577	32	64	128	256	512	1
	Total parameters: 10,179,713						Layer output shape					
	Trainable parameters: 10,179,713						(Rows, columns, channels)					
	Nontrainable parameters: 0											

Furthermore, each convolutional layer also includes an activation function, which performs the nonlinear transformation of the features generated from the convolution operation (Dettmers 2015). In the proposed model, the input and output data sets, 24-bit RGB orthoimage, and 8-bit grayscale elevation map pairs with the value range [0, 255] are normalized to the range [0, 1] by dividing them by 255. Thus, the activation function should progressively change from 0 to 1 with no discontinuity for generating the output. The rectified linear unit activation function (ReLU), $f(x) = \max(0, x)$, is a very popular choice for use in hidden layers; it is faster than many activation functions, such as sigmoid. The ReLU function does not always output a nonzero, so it results in fewer neurons being utilized and less dependence between features (Nair and Hinton 2010). In addition, the sigmoid activation function (also known as logistic), $f(x) = 1/(1 + exp(-x))$ is used in the output layer to generate the continuous values for the elevation map, instead of using the SoftMax function to classify the objects in SegNet (Badrinarayanan et al. 2017). The detailed model layers and each layer output shape for each patch size trial are shown in Table 2, where the types of layers are described in the Keras 2.3 style (Chollet 2015).

Additionally, this paper uses the "Sequential model API" in Keras to set up the convolutional encoder-decoder network model. When compiling the model, it uses "rmsprop" as the optimizer, and "mean_squared_error" as the loss function (Chollet 2015); "validation_split" is set to 0.05, which means that 95% of the data sets is used for training the model and 5% of the data set is used for validation. In this paper, the efficiency of early stopping compared to nonstopping has been evaluated. The early stopping technique stopped model training when the monitored quantity had stopped improving (Chollet 2015), such that the training loss or validation loss had not decreased for 10 epochs. This paper uses "EarlyStopping(monitor='val_loss', patience = 10)." What's more, this paper uses the "same" padding for max pooling layers. Because the model input sizes are 32, 64, 128, 256, and 512, which can be divided by 32 (2525), the padding setting in max pooling should have no impact on the result, because in each max pooling layer, the layer input size is halved, while the layer output size is still an integer that can be divided by 2. However, the model results varied on this setting. Using the "same" padding generated a better result than "valid" padding.

Model Prediction and Postprocess

The input layer and output layer of the proposed model (Fig. 5) indicate that the trained model predicts an elevation map patch from an input orthoimage patch. A model prediction example is shown in Fig. 7(a), while the edged area of each prediction patch is different from the center area. This is because the zero padding is used in convolution operations. The normal convolution operation shrinks the input image size down to the filled center region in Fig. 6(a). In this paper, the added padding operation enlarges the image size with "0" before the convolution operation [Fig. 6(a)]. Then, the zero-padding convolution ensures that the output maintains the same size as the input. However, the added "0" produces unwanted features in the edge of the prediction patches. The side-by-side assembly of predictions in Fig. 7(a) shows the unexpected gridlines.

Fig. 7(b) shows the workflow of the orthoimage disassembling and elevation map assembling algorithm, which generates the elevation map without unexpected gridlines. This algorithm needs to disassemble the orthoimage into several overlapping patches. The required number of patches is determined by Eq. (2). When assembling the elevation map, only selected parts of each patch will be used. Compared with the side-by-side approach, the proposed overlapping algorithm replaces the patch edges with other predictions' center regions. Additionally, the assembly of the elevation map has the same GSD as the orthoimage. Then, the 3D geometry data can be reconstructed using Eq. (3)

(2)

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

(3)

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x \cdot GSD \\ y \cdot GSD \\ \text{Elevation} \end{bmatrix} = \begin{bmatrix} \left(u - \frac{\text{Image Width}}{2}\right) \cdot GSD \\ \left(v - \frac{\text{Image Height}}{2}\right) \cdot GSD \\ \text{Elevation\_map}[u, v] \times \frac{\text{Elevation}_{\text{Range}}}{255} - \left|\text{Elevation}_{\text{LowBoundary}}\right| \end{bmatrix}$$

Experiment

System Configuration

Experiments in this paper were conducted using the proposed deep learning model in Fig. 5 and Table 2. The configuration of the system environment was 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×128GB×12) DDR4 2666 MHz memory, and $4 \times 11\text{GB}$ memory GeForce RTX 2080 Ti GPUs.

Experiment Data Set

In this paper, the experiment data sets, the orthoimage, and the elevation map pairs were generated using the method in Jiang and Bai (2020). In this method, an original orthoimage taken by a drone will be preprocessed as presented in Table 3. Additionally, in this paper, the edges of the orthoimages and elevation maps are removed to make their width (1,536 pixels) and height (1,536 pixels), which are exactly divisible by 32, 64, 128, 256, and 512. This is because the various patch size configurations will be compared.

Table 3. Image processing parameters

Processing step		Image size (pixels)	GSD (cm/pixel)	Site size (m2)
Image preprocessing	Original	$3{,}648 \times 4{,}864$	0.27	9.85×13.13
	Cutting to square shape	$3{,}648 \times 3{,}648$	0.27	9.85×9.85
	Scaling, 0.5	$1{,}824 \times 1{,}824$	0.54	9.85×9.85
	Removing margin, 128 pixels	$1{,}568 \times 1{,}568$	0.54	8.47×8.47
Additional processing in this paper	Removing each side 16 pixels	$1{,}536 \times 1{,}536$	0.54	8.29×8.29

Fig. 8(a) shows the model training and validation data sets. The first and second column orthoimages were taken in Atwater Park (Shorewood, Wisconsin) during different seasons. Data A and B were taken on March 24, 2019, when the vegetation had not recovered yet. Data C and D were taken on June 5, 2019, when the vegetation was growing. Additional Data AC, CA, CG, and CI were taken September 2019, when the vegetation was fully grown. Data AC is the same wooden platform as B and D; Data CA is another wooden platform on this site; and Data CG and CI detail the umbrella and stairways. In addition, Fig. 8(b) includes an additional orthoimage and elevation map pair, which is proposed to be used for quantitatively evaluating the trained model. Furthermore, the elevation maps were aligned by picking a point on the wooden platform/path and setting its elevation as ±0.00±0.00. Fig. 8(c) shows the spatial relation among these nine data sets. In addition, the other orthoimages captured from different drone flight heights on the experiment site are proposed to be used to evaluate the trained model and will be shown in the "Discussion" section.

Training and Validation

The model training parameters including batch sizes, epochs, and data set numbers are listed in Table 4. The 100 epochs and early stopping were shared for the five different patch size trials. Eight orthoimage and elevation map pairs [Fig. 8(a)] and their four rotations were used to train the model. Thus, the total number of data sets is eight times the number listed in the last column of Table 1. The data set numbers varied for the five different patch size trials, because the system memory limitation resulted in different strides being used for creating data sets. Moreover, in this paper, when training the model, the "batch size $= 32$" was used in the 512×512-pixel patch trial, and "batch size $= 128$" was used in the other trials. This is because of the single GPU's memory limitation (11 GB or 10.24 GiB); an additional 3.38 GiB memory and 2.29 GiB memory are needed for each GPU with batch sizes of 128 and 64, respectively. Fortunately, the small batch size in the 512×512-pixel patch trial only results in more model training times in each epoch.

Table 4. Model training parameters and results

Patch size trials					Training epoch trials	
Patch sizes	Data sets (validation split = 0.05)			Batch sizes	Early stopping (monitor = 'val_loss', patience = 10), epochs = 100	
	Total number	Training	Validation		With early stop	Without early stop
32×32	288,800	274,360	14,440	128	29	100
64×64	70,688	67,153	3,535	128	27	100
128×128	64,800	61,560	3,240	128	18	100
256×256	14,112	13,406	706	128	35	100
512×512	3,872	3,678	194	32	32	100

The loss results of model training for each trial are shown in Fig. 9(a), and the loss results of model validation (also known as model testing) for each trial are shown in Fig. 9(b). The five different patch size trials were stopped at different epochs (Table 4). The 128×128-pixel patch trial stopped at the 18th epoch is the earliest trial, and the 256×256 patch stopped at the 35th epoch. The 256×256-pixel patch took the most epochs for the validation loss to reach stable for 10 epochs. Furthermore, the validation results of each patch size are shown in Fig. 10, where the ground truths are the elevation map patches used in training the model and the predictions are the model outputs generated from the trained model with the corresponding inputs. The ground truths and model predictions are shown in the same viridis colormap range—the more similar the color, the more accurate the predictions. Visually, the model predictions are not a constant color (grayscale value) for a 32×32-pixel patch as the elevation map patches. The developed model decodes the elevation values for

each pixel of the input patch instead of a single elevation value for the whole patch. The model output results show that the trained model can distinguish different objects, such as the wooden paths that are distinguished from the ground in the 256×256 and 512×512 trials. The trained model also shows the ability to correct elevation value errors that occur in the wooden path of 256×256 and 512×512 trials. In detail, the wooden paths of 256×256 and 512×512 in the ground truth have incorrect elevation values, while the predictions for the wooden paths show corrected elevations.

For the five early stopping different patch size trials, the minimum model training loss occurred on the 128×128-pixel patch trial at its 18th epoch [Fig. 9(a)]. The 128×128-pixel patch also has the smaller model validation loss, while the minimum model validation loss occurred on 64×64-pixel patch trial at its 27th epoch. The Data A predictions in Fig. 10 indicates that the 128×128-pixel patch trial has better performance than other patch sizes in the early stopping trials, and the overlapping assembled predictions in Fig. 11 confirms that the 128×128-pixel patch has the best performance in the early stopping trial for Data CI as well. That may be because the 128×128-pixel patch balances the local features of each 32×32 patch and contains global features to connect each single 32×32 patch as well. The detailed comparisons of the different patch sizes will be stated in the "Discussion" section.

Another model training was conducted without early stopping, the 18 to 100 epochs model training loss and validation loss of the five different patch size trials are shown in Figs. 9(c and d). The 128×128-pixel patch has the minimum model training loss of 8.74E-06 at 100 epochs, which is smaller than 1.82E-04 at the early stopping trial. The 64×64-pixel patch and 256×256-pixel patch trials have a more stable decreasing trend and smaller values for training loss compared to the extreme size patches 32×32 and 512×512. Therefore, using the 128×128-pixel patch for the developed convolutional encoder-decoder network model has the best model training and validation performance, followed by the 64×64-pixel patch and 256×256-pixel patch.

Testing

The Testing data AO in Fig. 8(b) is different from the Training data AC in Fig. 8(a). The data were captured on the same day but in different flight paths and sequences; the drone landed after it captured the AC low-high orthoimage pair and took off again to capture the AO pair. For the AC pair, the 10-m orthoimage was captured first followed by the 20-m orthoimage; but for the AO pair, the 20-m orthoimage was captured first followed by the 10-m orthoimage.

Fig. 12 contains the model predictions for the Testing data AO. Visually, the 128×128-pixel patch has the best result in the early stopping trial, and the 64×64-pixel patch is better than others. The patches 128×128 and 256×256 are better than others in the 100 epochs trials. The 100 epochs results are more detailed than the early stopping ones. These 2D predictions can easily be converted to 3D point clouds using Eq. (3) with the selected 2,304 (48×48) points ($\text{strides} = 32$ pixels in column and row directions). Fig. 13 overlaps the 128×128 and 256×256 prediction point clouds (one pixel is one point) with the ground truth, which is converted from the elevation map and plotted with RGB cubes. The model predictions have the similar shape as the ground truth and are more accurate than the ground truth for the wooden platform surface and its edges. The quantitative evaluations will be stated in the "Discussion" section.

Discussion

Patch Size Comparison and Discussion

As the model training and testing results show, the 128×128-pixel patch and the 64×64-pixel patch are better than the other patch sizes in the early stopping trials. Fig. 14 shows the overlapping assembly of model predictions with the ground truth elevation map of the eight-model training data sets between these two patch sizes. In addition, several interesting points were selected to show their X/Y-profile elevation (m) changes.

Each data set in Fig. 14 has ground surface, large objects or structures, and small objects. For the ground surface 3D reconstruction, the 128×128-pixel patch has the best performance, as seen with the selected points in Data A and the Y-profiles of Data C, D, and AC. Additionally, the sparse grass on the ground shows no impact to the 3D reconstruction of the ground surface shape, such as the X-profiles of Data B and D (Fig. 14). The trained model with 128×128-pixel patch correctly identifies that these regions are ground surface and not vegetation. For large object 3D reconstruction, the 128×128-pixel patch also has the best performance, seen in the Y-profile of the umbrella in Data CG, the Y-profile of the stairways in Data CI, and the wooden platforms and wooden paths in all of the training data sets. For small objects, both the 128×128-pixel and 64×64-pixel patches have good performance in the 3D reconstruction of the small objects' shapes, such as the X/Y-profiles of the garbage can in Data B and D.

In general, the 128×128-pixel patch has a better performance with the early stopping setting at the 18th epoch than the 64×64-pixel patch trial with 27 epochs. However, training the developed model with the smallest 32×32-pixel patch has given a potential function to correct the elevation errors in the ground truth, such as the wooden path edge in the center region of Data A, the wooden platform corner in Data B, and the gap between the platform and the garbage can in Data B (Fig. 15). However, the large patch size 256×256 and 512×512 trials retained these errors. Therefore, the median size 128×128-pixel patch is the best option for balancing the local features and global features, each elevation value in the 32×32-pixel patch, and the connections between 32×32-pixel patches.

Texture Comparison and Discussion

Aside from the ground surfaces, the vegetation surfaces and wooden surfaces are the two major textures in the experiment site [Fig. 8(c)]. The vegetation surfaces were captured during different seasons; the vegetation blocks show different colors in Data A, B, C, D, AC, and CG (Fig. 14). In Fig. 14, the selected point in Data AC is on the ground surface. The neighboring vegetation blocks were 3D-reconstructed well in the X-profile of Data AC (128×128-pixel patch), in which the real vegetation blocks' surface heights ranged from 0.6 to 0.9 m on September 5, 2019. The X-profiles of Data A and B also crossed the withered vegetation blocks, in which the 128×128-pixel patch results are matched with the ground truth. In additional, Data CG and CI contain denser foliage in the shrub blocks, which are different from the vegetation blocks. The Y-profile of Data CG and X-profile of Data CI are matched with the ground truth. The wooden surfaces and ground surfaces were captured in different brightness environments, and their colors are varied in Fig. 14. When creating the experiment data sets, all wooden surfaces (except the stairways) were set as elevation ± 0.00 m. They were all 3D-reconstructed well in the model predictions. Furthermore, the Y-profile of Data CI shows the 3D-reconstructed stairways are matched with the ground truth, and the selected point in Data CA has the correct elevation differential to the wooden platform as well. Thus, the developed model that trained with 3-channel RGB orthoimages is robust in complex textured regions for the early stopping 128×128-pixel patch trial.

In addition, there are three kinds of poorly textured regions in the experiment data set, including shaded spots, shaded strips, and shaded blocks. For small spots of shade, such as the garbage can's shade in Data D (Fig. 14), the 128×128-pixel patch generated the correct predictions. For large shade blocks, such as the tree's shade and umbrella's shade on the wooden platform in Data CA and CG, respectively (Fig. 14), the 128×128-pixel patch has the correct predictions. The early stopping 128×128-pixel patch trial has inconsistent performance for the shaded strips. The selected point in Data AC is on the shade of the vegetation block. The ground surface was identified as vegetation in the 64×64-pixel patch trial but was correctly identified using the 128×128-pixel patch. However, the 64×64-pixel patch trials are more aligned with the "ground truth" than the 128×128-pixel patch trials, such as the Y-profiles of the shaded ground surface close to the wooden platform in Data CA and the shaded area next to the bottom stairs in Data CI (Fig. 14). Fortunately, adding model training epochs can improve the prediction accuracy (Fig. 16), which will be discussed in the next section.

Therefore, using the $128 \times 128 \times 3$ RGB orthoimage input patch and $128 \times 128 \times 1$ grayscale elevation map pair data sets to train the developed convolutional encoder-decoder network model has a good performance both in complex textured and poorly textured regions.

Epoch Comparison and Discussion

The validations of Data CA and CI (Fig. 16) indicate that it is worth continuing to train the model after the early stopping point to improve the performance of the 128×128-pixel patch. This is due to the model not training well enough at the 18th epoch, though it still has the potential to narrow down the variations of the validation loss [Fig. 17(a)]. In addition, the comparison of testing results (Fig. 12) shows that the 128×128-pixel and 256×256-pixel patches are better and smoother than other patch sizes in the 100 epochs trials, and the comparison of the two 100 epochs validation loss curves in Figs. 17(a and b) confirmed that the 128×128-pixel patch is more stable and can reach a stable trend earlier than the 256×256-pixel patch. Thus, the well-trained 128×128-pixel patch has both the best model training and prediction performance for the developed convolutional encoder-decoder network model.

Furthermore, the quantitative evaluation of the model validation accuracy and testing accuracy were conducted by measuring the point cloud (Fig. 13). In detail, for each validation result of the 128×128-pixel patch early stopping trial and 128×128-pixel patch 100 epochs trial, 2,304 (48×48) points (the centers of each 32×32-pixel patch) are selected from the corresponding orthoimage and the assembled elevation map predications ($1{,}536 \times 1{,}536$ pixels). Then, the 3D point clouds were generated using Eq. (3) with the selected 2,304 points. For each model training and validation data from A to CI, the variable "ELE-DIFF-EARLY" was created as the elevation differential between ground truth and 128×128-pixel patch early stopping, and variable "ELE-DIFF-100" was created as the elevation differential between ground truth and the 128×128-pixel patch 100 epochs. Both variables have 18,432 ($2{,}304 \times 8$) samples. For the Testing data AO, the same variables were created and named "AO-DIFF-EARLY" and "AO-DIFF-100" with 2,304 samples. The descriptive statistics for the four variables are listed in Table 5. For the model training and validation results, the 99% confidence interval (CI) of elevation differential is reduced from (1.6, 2.1) to (0.6, 0.8) cm by adding the model training epochs. In addition, the trained model has a good result in predicting the elevations for the Testing data AO; the elevation differential has a 99% CI of (2.14, 4.08) cm. Therefore, the model training epochs have a positive effect in improving the model accuracy; after 100 epochs the developed model is a well-trained model.

Table 5. Elevation differential results

Sample	N	Mean (m)	StDev	SE mean	99% CI for μ	Minimum	Q1	Median	Q3	Maximum
ELE-DIFF-EARLY	18,432	0.018495	0.133267	0.000982	(0.015966, 0.021024)	−2.31373	−0.03922	0	0.07843	1.84314
ELE-DIFF-100	18,432	0.0073	0.058502	0.000431	(0.006190, 0.008410)	−1.05882	0	0	0.03922	0.94118
AO-DIFF-EARLY	2,304	0.0424	0.17635	0.00367	(0.03293, 0.05187)	−1.17647	−0.03922	0.03922	0.11765	1.01961
AO-DIFF-100	2,304	0.03111	0.18096	0.00377	(0.02140, 0.04083)	−1.05882	−0.03922	0	0.07843	1.21569

Accuracy Evaluation and Discussion

Fig. 18(a) shows the distributions of the elevation differential between the ground truth, model validation results, and model testing results. The histogram of "ELE-DIFF-100-CM" shows that 94% of points from the model training data sets have an elevation error less than 10 cm in the well-trained model (100 epochs). The two histograms, "AO-DIFF-100-CM" and "AO-DIFF-EARLY-CM," of the Testing data AO show that the well-trained model has a significant improvement over the early stopping model. The well-trained model prediction accuracy is 52.43% compared to the 47.05% on the early stopping model, in which an accurate elevation measurement is defined as measurement error that is equal to or less than 5.0 cm (Takahashi et al. 2017). The worst predictions ($\text{error} > 25\ \text{cm}$ or $\text{error} < -25\ \text{cm}$) account for 9.64% and 12.37% in the well-trained model and early stopping model, respectively. In addition, the prediction contour maps are shown in Fig. 18(b), and the elevation differentials were mapped as well. Most of the worst predictions of the well-trained model are on the edges of the wooden platform and garbage cans. This is because the ground truths on these locations are incorrect; the model predictions have corrected them. Excluding these errors, the model prediction accuracy will increase. Thus, the well-trained model has at least a 52.43% accuracy in estimating the construction site elevations.

Fig. 19 shows two 20-m orthoimages that have the same $\text{GSD} = 0.54\ \text{cm/pixel}$ as the model training data set. The blue garbage can (17.65 cm lower than the wooden platform) is the new object not used in training the developed model, and the original images were cut to a $3{,}584 \times 4{,}864$-pixel patch without image resizing. The Y-profile at the blue garbage can is $-13.7\ \text{cm}$, which is close to the true value with the error $3.95\ \text{cm} < 5.0\ \text{cm}$. The Y-profile on the top of the umbrella is 3.196 m, which is accurately matched with its true value of 3.20 m. Therefore, training the developed model with the orthoimages captured at height 10 m can be used in 3D reconstruction of orthoimages at height 20 m. The trained model is also able to generate the accurate elevations for the orthoimages at 20 m as well. However, its performance worsens for the orthoimages at 40 m and above.

Furthermore, the top view of the experiment site in Fig. 8(c) was captured at a flight height of 100 m. The elevation prediction results of the well-trained models with 128×128-pixel and 32×32-pixel patches are shown in Fig. 20. The 32×32-pixel patch results show the ground surfaces, wooden surfaces, and shrub blocks are reconstructed well, but the vegetation blocks are assigned with incorrect elevations. The bad prediction of the 128×128-pixel patch occurs around (500, 2,000), where the shaded wooden path was not included in the model training data sets. Thus, to make the model satisfied with complex construction site situations, a comprehensive data set (orthoimage and elevation map pairs) is required. This data set should include different textures of the construction site, because the top-layer materials of the construction sites are not limited to vegetation, water, snow, sand, rock, soil, concrete, asphalt, buildings, and structures. For training a precise deep learning model, the number of data sets should be large enough to cover the various construction site surfaces.

Conclusion and Future Work

Conclusion

This paper presents a single-frame, image-based, 3D-reconstruction method, which only needs a drone-based orthoimage as the input. The overall procedure is summarized in Fig. 1, which includes the following: (1) using a drone to acquire construction site orthoimages, (2) using an overlapping disassembling algorithm to generate the overlapping patches and their sequence number, (3) using the trained convolutional encoder-decoder network model to predict the elevation map for each patch, (4) assembling the prediction patches with the assigned sequence, and (5) converting the elevation map to elevation data or 3D point cloud.

This experiment showed that the 128×128-pixel patch had the best prediction performance when the elevation values were shared in the elevation map with a 32×32-pixel patch. Adding model training epochs

had a positive relationship to the model prediction accuracy. The testing results showed that the well-trained model had a 52.43% accuracy in elevation estimation with ± 5.0 cm error, 66.15% accuracy with ± 10.0 cm error, and less than 10% results with ± 25 cm errors. Compared with the 94% accuracy (error $\pm$ 10.0 cm) in model training, it still has huge potential for improving the deep learning method for single image-based 3D-reconstruction of construction sites. The proposed deep learning model can be used to estimate construction site elevations if the model is well-trained with data sets of similarly textured objects and sites. In this orthoimage-based, 3D-reconstruction method, the model training data sets were the reference information to estimate the construction site elevations. Thus, the performance of the proposed method relied on the quality and quantity of the model training data sets. The quality meant more comprehensive texture features and geometry shape features, while the quantity helped to build the ability to ignore incorrect elevation values in the data set and noise in the model predictions.

Furthermore, the proposed overlapping disassembling and assembling algorithm was needed, which made the workstation system more able to train a deep learning model with larger sized images instead of shrinking images and losing image details. By disassembling the data sets into multiple small patches, the number of data sets was increased significantly. With the suitable patch size, such as the 128×128-pixel patch, the model balanced the global features and local features, and was even well-trained earlier than larger patch sizes. In addition, the smallest 32×32-pixel patch contained the maximum local features, which was important in correcting the incorrect elevations that occurred in the model training data sets, which were produced by other methods. It also showed an advantage in the 3D reconstruction of points, lines, and planes with sharp elevation changes, such as the corner of wooden platforms and the edges of the wooden paths.

Future Works

In this paper, the model training data set was limited to 10-m drone-based orthoimages, which only contained a few objects in a single-image frame. In addition, the formation of the elevation map only contained single elevation values in each 32×32-pixel patch. Therefore, adding the sixth convolution layer or adding the filters in the fifth convolution layer for the developed CNN encoder model had nonsignificant improvement in the model prediction. Future research can assign more elevation values to each 32×32-pixel patch, and the proposed model may need additional CNN encoder layers and CNN decoder layers to connect the added elevation features. In addition, increasing the drone flight height can enlarge an image's spatial resolution and include more objects. Then, future research can train the proposed model with more data sets at different flight heights other than the 10-m height in this paper.

To increase the accuracy of the elevation estimation, future research would use image segmentation or image classification to assign a class label for each patch (32×32-pixel). On one side, the label can be used as the additional reference information (feature map) to increase the accuracy of prediction. On the other side, the vegetation areas can be detected from the label and removed to obtain the ground elevations on the construction site, which is more important to earthwork operations than knowing the site surface height.

Data Availability Statement

The model training and testing data sets [orthoimage and elevation map pairs appear in Figs. 8(a and b)] are available from the corresponding author upon reasonable request. The Python code (convolutional encoder-decoder network model appears in Fig. 5 and Table 2) is available from the corresponding author upon reasonable request.

Acknowledgments

This work was supported by the McShane Endowment fund at Marquette University.

References

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.

Chen, W., Z. F1u, D. Yang, and J. Deng. 2016. “Single-image depth perception in the wild.” In Proc., Conf. on 30th Neural Information Processing Systems (NIPS 2016), 730–738. Barcelona, Spain: NeurIPS.

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

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.

Eigen, D., C. Puhrsch, and R. Fergus. 2014. “Depth map prediction from a single image using a multi-scale deep network.” In Proc., 28th Conf. on Neural Information Processing Systems (NIPS 2014), 2366–2374. Montréal, Canada: NeurIPS.

Garg, R., V. K. BG, G. Carneiro, and I. Reid. 2016. “Unsupervised CNN for single view depth estimation: Geometry to the rescue.” In Proc. 14th European Conf. on Computer Vision (ECCV 2016), 740–756. Amsterdam, Netherlands: Springer.

Haur, C. J., L. S. Kuo, C. P. Fu, Y. L. Hsu, and C. Da Heng. 2018. “Feasibility study on UAV-assisted construction surplus soil tracking control and management technique.” In Proc., 5th Annual Int. Conf. on Material Science and Environmental Engineering (MSEE2017). Xiamen, China: IOP Publishing.

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

Kim, H., and H. Kim. 2018. “3D reconstruction of a concrete mixer truck for training object detectors.” *Autom. Constr.* 88 (Apr): 23–30. https://0-doi-org.libus.csd.mu.edu/10.1016/j.autcon.2017.12.034.

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.

Laina, I., C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. 2016. “Deeper depth prediction with fully convolutional residual networks.” In Proc., 4th Int. Conf. on 3D Vision (3DV 2016), 239–248. New York: IEEE.

Li, Z., and N. Snavely. 2018. “MegaDepth: Learning single-view depth prediction from internet photos.” In Proc., 2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition, 2041–2050. New York: IEEE.

Liu, F., C. Shen, and G. Lin. 2015. “Deep convolutional neural fields for depth estimation from a single image.” In Proc., 2015 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2015), 5162–5170. New York: IEEE.

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.

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.

Takahashi, N., R. Wakutsu, T. Kato, T. Wakaizumi, T. Ooishi, and R. Matsuoka. 2017. “Experiment on UAV photogrammetry and terrestrial laser scanning for ICT-integrated construction.” In Proc., 2017 Int. Conf. on Unmanned Aerial Vehicles in Geomatics, 371–377. Bonn, Germany: ISPRS.

Zhou, X., G. Zhong, L. Qi, J. Dong, T. D. Pham, and J. Mao. 2017. “Surface height map estimation from a single image using convolutional neural networks.” In Proc., 8th Int. Conf. on Graphic and Image Processing (ICGIP 2016). Tokyo: Society of Photo-Optical Instrumentation Engineers.