4-2022

Fast-PGMED: Fast and Dense Elevation Determination for Earthwork Using Drone and Deep Learning

Sisi Han
Marquette University

Yuhan Jiang
South Dakota State University

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

Journal of Construction Engineering and Management, Vol. 148, No. 4 (April 2022): 04022008. DOI.

Fast-PGMED: Fast and Dense Elevation Determination for Earthwork Using Drone and Deep Learning

Sisi Han
Department of Civil, Construction, and Environmental Engineering, Marquette University, Milwaukee, WI
Yuhan Jiang
Department of Construction and Operations Management, South Dakota State University, Brookings, SD
Yong Bai
Department of Civil, Construction, and Environmental Engineering, Marquette University, Milwaukee, WI

Abstract

This paper presents a time- and cost-effective elevation determination method for earthwork operations using ready-to-fly imaging drones and deep learning technologies. The proposed method is named the *fast pixel grid/group matching and elevation determination* (*Fast-PGMED*) algorithm. The input data are a pair of approximate 2:1-scale top-view images, and the output is the determined

elevation map for the scanned station. Feature matching of the two multiscale images is conducted by calculating correlations between target patch predictions (via *DeepMatchNet*, a fully convolutional network) and potential target patches (via virtual elevation model). The overall processing time is about 21 s (including 5 s for low-high orthoimage assembly, 3 s for patch feature generation, and 13 s for pixel matching) to process a 2,500-pixel grid, and the generated elevation values are as accurate as photogrammetry (within 5-cm error) but took much less time. Moreover, the developed method has been evaluated with two different drones. Volume measurement was quickly conducted via 2D elevation maps and accurately estimated via dense point clouds and Civil 3D.

Introduction

Utilizing ready-to-fly imaging drones and photogrammetry still is an attractive cost-effective solution for construction site modeling, earthwork estimation, and progress monitoring. A typical workflow of drone-based soil measurement [Fig. 1(a)] includes: drone image acquisition, photogrammetry, point cloud file conversion, soil mesh surface creation, and volume estimation (Haur et al. 2018; Nassar and Jung 2012). This 3D reality model-based earthwork volume calculation is accurate but has limited capacity for improvement in construction administration because deploying this workflow requires "one processing day" to estimate on-site soil volume (Haur et al. 2018). In contrast, Jiang and Bai (2020b, 2021) developed a low-high orthoimage pairs-based 3D reconstruction method with a *pixel grid matching and elevation determination* (*PGMED*) algorithm for construction site elevation determination [Fig. 1(b)], where earthwork estimation was easily conducted on 2D plans.

This paper presents a new version of a *fast pixel grid/group matching and elevation determination* (*Fast-PGMED*) algorithm with deep learning-based patch feature generation and drone landing pad pixelwise segmentation [Fig. 1(c)]. As a result, the proposed method archives the accurate and nearly real-time dense elevation determinations, benefiting construction professionals in monitoring and controlling excavation progress via cost-effective drone imaging and fast and dense 3D reconstruction.

Background

Currently, the "DJI Phantom" series (Moon et al. 2019; Yang et al. 2018), "DJI Inspire" (Aguilar et al. 2019; Li and Lu 2018), and "DJI Mavic" (Park et al. 2019) are the most popular aerial imaging drones used in the architecture, engineering, and construction (AEC) community. These ready-to-fly (noncustomized) drones [like the *DJI Phantom 4 Pro V2.0* in Fig. 1(d) and *DJI Mavic 2 Pro* in Fig. 1(e)] are easily controlled and portable quadcopters, which have downward vision systems and global positioning system (GPS) for stable hovering at a planned position; digital cameras are mounted on 3-axis (pitch, roll, yaw) gimbals to enhance the camera's stabilization (DJI 2020, 2021). In this paper, orthoimaging is defined as setting the gimbal's pitch-axis at negative 90° to make the camera lens face down to ground surfaces; then, the captured images are top-views or plan views of the scanned sites with an approximate spatial resolution (a pixel's length stands for a physical length of ground in centimeters) of ground sampling distance (GSD), which has a linear relation to drone altitude.

Construction Site 3D-Mapping and Measurement Workflows

Currently, construction site surveying techniques are shifting from total station and GPS to laser scanning and LiDAR (Chen et al. 2018; Du and Teng 2007; Kwon et al. 2017; Li and Lu 2018; Moon et al. 2019), close-range photogrammetry (Aguilar et al. 2019; Arias et al. 2005; Barazzetti et al. 2010; Inzerillo

et al. 2018; Park et al. 2019; Sung and Kim 2016), drone photogrammetry (Haur et al. 2018; Nassar and Jung 2012; Nex and Remondino 2014; Siebert and Teizer 2014), and visual simultaneous localization and mapping (SLAM) (Shang and Shen 2018). In general, *Structure from Motion* (SfM) photogrammetry [Fig. 1(a)] starts with capturing highly overlapping image series with a minimum of 70% and 40% overlap in longitudinal and traversal coverage, respectively (Siebert and Teizer 2014; Takahashi et al. 2017). Then, extracting and matching feature points from images via scale-invariant feature transform (SIFT) (Wu 2007). For determining the spare points' geometrical data, the aerial triangulation method is applied in the case of ordered series of top-views, and the SfM is applied in the case of unordered image collections (Snavely 2010). A key task in SfM is to determine the camera's movements $\boldsymbol{T} = [xyz]^T$ and rotations ($\boldsymbol{R}$) to align the sequence stations to the initial station's coordinate. After getting the sparse point clouds, the patch-based multiview stereo (PMVS) or clustering views for multiview stereo (CMVS) are used to generate dense point clouds (Furukawa and Ponce 2010; Wu 2011; Wu et al. 2011). These complicated procedures would yield accurate elevation results (error within 5.00 cm) (Takahashi et al. 2017).

Additionally, Fig. 1(a) shows a typical workflow of drone-based soil measurement (Haur et al. 2018; Nassar and Jung 2012), which includes: (1) high ratio overlapping drone image acquisition; (2) 3D point cloud generation via photogrammetry software packages, e.g., Metashape (Agisoft LLC, St. Petersburg, Russia), Pix4Dmapper (Pix4D, Prilly, Switzerland), and ReCap Photo (Autodesk, Mill Valley, CA); (3) point cloud file conversion via ReCap (Autodesk, Mill Valley, CA); and (4) soil mesh surface creation (via Triangulated Irregular Network, TIN) and volume estimation (by comparing to design mesh surfaces) using Civil 3D (Autodesk, Mill Valley, CA). This 3D reality model-based earthwork volume calculation is accurate but requires "one processing day" to estimate on-site soil volume (Haur et al. 2018).

Requirements and Limitations in Feature Matching

Feature matching is the primary task in image-based 3D reconstruction, such as SIFT applied in sparse reconstruction and PMVS applied in the dense reconstruction of SfM. Existing feature points also include the histogram of oriented gradients (HOG) (Kim and Kim 2018; Memarzadeh et al. 2013), speeded up robust features (SURF) (Bay et al. 2008), oriented fast and rotated brief (ORB) (Rublee et al. 2011), as well as some newly developed feature points, e.g., guided local outlier factor (GLOF) (Wang and Chen 2021), advanced neighborhood topology consensus (ANTC) (Liu et al. 2021), and multitask feature extraction network and self-supervised feature points (Li et al. 2021). Feature matching via those feature points is robust in image rotation, scaling, and even in perspective transformation; however, the matched feature points are only sparsely and irregularly distributed in image overlaps and are excluded in low-contrast regions. In contrast, the patch feature-based correlation method, e.g., normalized sum of squared difference (NSSD) and normalized cross correlation (NCC) (Kaehler and Bradski 2016; Lewis 1995; OpenCV 2018a), has advantages in customization and dense matching shown in PMVS and PGMED (Jiang and Bai 2021).

Additionally, deep learning methods of neural networks (NNs) showed their ability in fast object detection with images and point clouds, where potential target objects' feature maps are extracted via NNs to match with the reference object's feature map (Li et al. 2021; Zhu et al. 2019). Based on that, the remote sensing (RS) community has applied NNs in RS image matching from different data sources, such as synthetic aperture radar (SAR)-optical image matching (Hughes et al. 2019, 2020), drone image and

geotagged orthomosaic matching (Mughal et al. 2021), and matching RS images with the same scenes from different times (Zhu et al. 2019). Dense matching was not necessary and was not conducted in these image matching tasks. In contrast, the computer vision (CV) community applied NNs in stereo matching tasks for depth (disparity) estimation (Choe et al. 2021; Luo et al. 2016) and 3D reconstruction (Knyaz et al. 2017), in which pixelwise dense matching is necessary. Different from the RS image matching, stereo matching tasks always have the same-sized and -scaled reference and target (left-right) images. Thus, Siamese networks architectures were utilized to parallelly process the reference and target images to generate reference and target feature maps (Choe et al. 2021), or the same-sized reference and target patches to generate reference and target patch representations (codes) (Knyaz et al. 2017; Luo et al. 2016). In both approaches, Siamese networks' reference and target outputs have the same dimensions. Like RS image matching, NNs were used in pixelwise matching and disparity estimation with reference and target feature maps (Choe et al. 2021), but semiglobal matching (Bethmann and Luhmann 2015) was used in processing patch representations (Knyaz et al. 2017; Luo et al. 2016).

However, these deep learning-based feature matching methods cannot be utilized to speed up the dense pixel grid matching of a low-high orthoimage pair due to the images' 2:1 scaling relation. To address this issue, this paper proposed an NN to generate half-sized feature maps for reference patches (in low-orthoimage) in order to quickly match the potential target patches (in high-orthoimage).

Research Scope and Contributions

This paper presents a time- and cost-effective solution for construction site elevation determination and earthwork measurement [Fig. 1(c)]. The new method is powered by the virtual elevation model (Fig. 2), where a drone captures two top-view images at different altitudes over target sites to assemble a pair of low-high orthoimage. The top-view captured at the high altitude (has an approximate distance H to ground surface) is called a high-orthoimage, and the top-view at the low altitude (has an approximate distance $H/2$ to the high altitude) is named a low-orthoimage. The low-high orthoimages have an approximate $1:2$ ratio in GSD and $2:1$ ratio in image scales. The virtual elevation model simplifies the process of matching reference points/pixels p and target points/pixels $\boldsymbol{p}'$ in low-high orthoimages (Jiang and Bai 2021). For a given reference pixel $p[v,u]$, each virtual elevation plane $Ele._i$ (range from $Ele._{Lower\ Boundary}$ to $Ele._{Upper\ Boundary}$ with an increment of $Ele._{step}$) can generate a potential target pixel $p_i'[v_i', u_i']$ via Eq. (1a)

(1a)

$$p_i'[v_i', u_i'] = f(v, u, Ele._i)$$

Inside Eq. (1a)

(1b)

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} u - w/2 + 0.5 \\ v - h/2 + 0.5 \end{bmatrix}$$

converts the reference pixel $[v,u]$ to image coordinate (x,y), where ww is image width, h is image height, and 0.5 transfers the coordinate from its top-left corner to the pixel center. Then

(1c)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x\dfrac{1 - Ele.\times 2/H}{2 - Ele.\times 2/H} \\ y\dfrac{1 - Ele.\times 2/H}{2 - Ele.\times 2/H} \end{bmatrix}$$

calculates the target point (x', y'), and the detailed derivations of which are discussed in Jiang and Bai (2021). Next

(1d)

$$\begin{bmatrix} v' \\ u' \end{bmatrix} = \begin{bmatrix} int(y' + h/2) \\ int(x' + w/2) \end{bmatrix}$$

converts target point (x', y') to pixel coordinate $[v', u']$. If the reference pixel $p[v, u]$ matches with the potential $p_i'[v_i', u_i']$, then the site point P has a virtual elevation value Ele_i (ideally virtual plane $Ele. = \pm 0.00$ has the distance H to high altitude). Therefore, in a low-high orthoimage pair, when all reference pixels $\boldsymbol{p}$ and target pixels $\boldsymbol{p}'$ are matched, the elevations are determined in the meantime for construction site $\boldsymbol{P}$.

For quickly deploying this proposed method for accurate and fast soil volume measurement [Fig. 1(c)], the following specific objectives and tasks are addressed: (1) a *DeepMatchNet* for generating patch features for reference pixels (in the low-orthoimage) to match the target pixels (in the high-orthoimage); (2) a *fast pixel grid/group matching and elevation determination* (*Fast-PGMED*) algorithm with a multiprocessing scheme for quickly matching pixel pairs in a dense pixel grid-style, while simultaneously determining elevation values; and (3) an elevation map based fast earthwork planning and estimation. Furthermore, parameter analysis and performance evaluation were conducted for the proposed method, and experimental applications compared the proposed method with existing methods on earthwork estimation.

Fast and Dense Elevation Determination Method Development

In this section, an FCN, named *DeepMatchNet*, is proposed to generate patch features. Next, a *Fast-PGMED* algorithm is proposed to quickly match pixels and determine elevation values.

DeepMatchNet Design

Patch Features

When reference and target images have the same scale, the matching of reference and target pixels equals the matching of reference and target patches (centered at the reference and target pixels), which can be solved by the template matching via calculating NCC score (Lewis 1995). However, the NCC cannot be directly applied to the low-high orthoimages because of their $2:1$ scaling relation. The previous *PGMED* algorithm utilized a 2×2 pooling operation to build up the approximate $2:1$ scaling relationship between the low-high orthoimages (Jiang and Bai 2021). In detail, a potential target patch [centered with a target pixel $p'[v', u']$ with radius R, see Fig. 3(a)] in a high-orthoimage matched a reference patch [centered with a reference pixel $p[v, u]$ and its three neighbors with radius $2R$, see Fig. 3(c)] in a low-orthoimage through a 2×2 pooling resized reference patch [centered with a pooled pixel with radius R, see Fig. 3(b)]; NCC scores between potential target patches and the resized reference patch were calculated to determine the best match result of reference and target pixel pairs.

In addition, Fig. 3 indicates a potential target pixel also is the corresponding pixel for the three potential pixels in the center of the reference patch; in reverse, the matching results of the low-high orthoimages should be reference-pixel and target-subpixel pairs with the best NCC scores, like the bottom-right subpixel example in Fig. 3(b). Thus, in *PGMED*, four NCC calculations (between the potential target patch and four-direction resized reference patches) were applied to further determine the best matched pixel-to-subpixel pairs and resulted in a slower *PGMED* (Jiang and Bai 2021).

In this paper, the resized reference patches are replaced with a *DeepMatchNet* generated target patch prediction to minimize the number of NCC calculations to one. In detail, the defined reference patch feature $\boldsymbol{I}[v,u]$ or $\boldsymbol{I}[v-2R-1{:}v+2R, u-2R-1{:}u+2R]$ is an RGB image patch with a size of $(4\mathrm{R}+2)\times(4\mathrm{R}+2)$, whose center is the reference pixel $p[v,u]$ and three neighboring pixels [Fig. 3(c)]. The defined target patch feature $\boldsymbol{I}'[v',u']$ or $\boldsymbol{I}'[v'-R{:}v'+R, u'-R{:}u'+R]$ is an RGB image patch with a size of $(2\mathrm{R}+1)\times(2\mathrm{R}+1)$ (which is half the size of the reference patch, to make the GSD the same), whose center is the target pixel $p'[v',u']$ [Fig. 3(a)]. The defined target patch prediction $\boldsymbol{I}_{FCN}[v,u]$ is an RGB fuzzy image patch with a size of $(2\mathrm{R}+1)\times(2\mathrm{R}+1)$ (half the size of the reference patch) [Fig. 3(b)], which is the output of *DeepMatchNet* for one reference patch $\boldsymbol{I}\ [v,u]$ input. Moreover, to speed up the NCC calculation, the RGB 3-channels target patch prediction $\boldsymbol{I}_{FCN}[v,u]$ and target patch feature $\boldsymbol{I}'[v',u']$ are first converted to 1-channel grayscale image patches $\boldsymbol{I}_{FCN}$ and $\boldsymbol{I}'$ with a size of $(2\mathrm{R}+1)\times(2\mathrm{R}+1)$ and then the NCCNCC score is calculated for them by Eq. (2). Here, $\boldsymbol{I}$ is the mean of $\boldsymbol{I}_{FCN}$, and the calculation is based on $\bar{I}=\frac{1}{N}\sum I_{FCN}$; and $\overline{I'}$ is the mean of $\boldsymbol{I}'$ patch, and the calculation is based on $\overline{I'}=\frac{1}{N}\sum I'$

(2)

$$NCC=\frac{\sum(I_{FCN}-\bar{I})(I'-\overline{I'})}{\sqrt{\sum(I_{FCN}-\bar{I})^2}\sqrt{\sum(I'-\overline{I'})^2}}$$

Model Architecture

The *DeepMatchNet* (a fully convolutional network, Fig. 4) is proposed to learn the features of the approximate 2:1 scaling transformation between the corresponding reference and target patches in the low-high orthoimages. After training the model, the *DeepMatchNet* can be used to generate target patch predictions $\boldsymbol{I}_{FCN}[v,u]$. Table 1 shows the detailed parameters of each model layer. The designed model output layer uses a 3-channel convolutional operation to create 3-channel RGB image patches as model outputs, which are half the size of the model input image patches. The hidden layers are eight convolution layers and one max-pooling layer. In detail, the first convolutional block, *Conv1-4*, translates input image patches to feature maps (which are the same size as the input patch) via convolution operation with zero padding (each layer output is the same size as the layer input) (Chollet 2015; Jiang and Bai 2020a). The single max-pooling layer (MPMP) reduces the layer input (*Conv4*'s output) to half the size of the layer output (*Conv5*'s input). In the second convolutional block, *Conv5-8* are four zero-padded convolution layers, preparing the half-sized feature maps for the output layer to create model outputs.

Table 1. DeepMatchNet layer parameters

Model architecture							Layer output shapes		
Symbols	Layers (type and kernel size)	Strides	Padding	Activations	Filters/channels		Row	Column	Channels
Input	input_1 (Input Layer)	—	—	—	—	3	78	78	3
Conv 1	conv2d_1 (Conv2D 3×3)	1	Same	ReLU	32	—	78	78	32
Conv 2	conv2d_2 (Conv2D 3×3)	1	Same	ReLU	64	—	78	78	64
Conv 3	conv2d_3 (Conv2D 3×3)	1	Same	ReLU	128	—	78	78	128
Conv 4	conv2d_4 (Conv2D 3×3)	1	Same	ReLU	128	—	78	78	128
MP	max_pooling2d_1 (Max-Pooling 2×2)	2	Same	—	—	—	39	39	128
Conv 5	conv2d_5 (Conv2D 3×3)	1	Same	ReLU	128	—	39	39	128
Conv 6	conv2d_6 (Conv2D 3×3)	1	Same	ReLU	128	—	39	39	128
Conv 7	conv2d_7 (Conv2D 3×3)	1	Same	ReLU	64	—	39	39	64
Conv 8	conv2d_8 (Conv2D 3×3)	1	Same	ReLU	32	—	39	39	32
Output	conv2d_9 (Conv2D 3×3)	1	Same	Sigmoid	3	—	39	39	3

Furthermore, each convolutional layer also includes an activation function to perform nonlinear transformation of the features generated from the convolution operation (Table 1). The rectified linear unit (ReLU) activation function, $f(x) = \max(0, x)$, is used in hidden layers to speed up model training, and the Sigmoid activation function, $f(x) = \frac{1}{1+e^{-x}}$, is used in the model output layer to generate continuous values (Jiang and Bai 2020a; Nair and Hinton 2010). Moreover, other modified models, including different filters (in *Conv 4* and 5) and an added dropout function in all hidden layers (*Conv1-8* and MP), are discussed later.

Model Configuration

Table 1 also lists the detailed layer output shapes for the input size 78×78-px, where the layer output size is reduced to 39×39-px at the MP layer, and remains 39×39-px until the model output layer. The reference patch size of 78×78-px for reference pixels in low-orthoimages and the target patch size of 39×39-px for target pixels in high-orthoimages are the initial sizes ($R = 19$-px) of the self-adaptive patch features used in *PGMED*. During the early stage of this research, different patch sizes were tested from $R = 11$ to 61-px, while $R = 19$-px balanced the pixel matching accuracy and speed. The use of small-patch sizes is not able to get enough unique features to find the correct reference and target patch pairs. Enlarging patch sizes increases the target patch prediction generation time and the NCC calculation time, and only works well in poorly textured regions but results in similar NCC scores for adjacent potential target patches.

Moreover, in this paper, the following configurations were used in *DeepMatchNet* training with Keras 2.3.1: (1) sets "*rmsprop*" as the optimizer, and "*mean_squared_error*" as loss function for model compiling; (2) sets "*validation_split=0.5*" to divide 50% of samples for model training and the other 50% of samples for model validation; and (3) sets "*monitor='val_loss', patience=10, mode='min'*" to (early) stop model training if the monitoring validation loss degradation occurs for 10 epochs. Furthermore, in model training, both the reference and target patches (small-patch 24-bit RGB images) are normalized from the 8-bit range [0,255] to range [0,1] to fit with the ReLU activation functions in model hidden layers. Thus, all three channels of a model output by the Sigmoid activation function are in range [0,1], which are multiplied by 255 to recover the 8-bit range [0,255] in each channel.

Fast-PGMED Algorithm Design

Pixel Grid and Pixel Group

In this paper, the two top-views were captured by a drone (*DJI Phantom 4 Pro V2.0*) via yielding the pitch-axis of the drone's camera gimbal to negative 90°. Within 0–10 m, the downward vision systems control the drone's altitude with vertical and horizontal hover accuracy of ± 0.1 m and ± 0.3 m, respectively; beyond 10 m, the GPS controls the altitude with vertical and horizontal hover accuracy of ± 0.5 m and ± 1.5 m, respectively (DJI 2020). The captured images have an original size of $3{,}648 \times 4{,}864$ pixels with an approximate $\text{GSD} = 0.27$ cm/px at altitude $= 10$ m and $\text{GSD} = 0.54$ cm/px at altitude $= 20$ m. Then, a pair of low-high orthoimage is assembled via the following steps: (1) shrink original images ($3{,}648 \times 4{,}864$-px) to half size; (2) cut half-size images to a square shape ($1{,}824 \times 1{,}824$-px, this step can be skipped, see *Experiment* section); and (3) align high images to low images by translation and rotation. To automatically align one high image to the corresponding low image, the low image covered region was located in the high image (see Fig. S1, a black box indicates a low image's region in the high image) by the SIFT feature matching and homography stated in (OpenCV 2018b). Next, the perspective point of the low image's center in the high image was compared with the

high image's center in order to determine the translation value for the high image. After the translation, SIFT matching was conducted for the low image and the translated high image. Then, the angle difference of the lines from a matched SIFT keypoint pair to the two image centers was determined for the high image's rotation. Strict control of a drone's altitude at 10m and 20m from site surfaces is difficult but not necessary in this research; the quality of low-high orthoimage assembly is discussed later. Thus, the assembled low-high orthoimages have the same image center and orientation, and the same size of $1{,}824 \times 1{,}824$ pixels, with an approximate $\mathrm{GSD} = 0.54$ cm/px at altitude $= 10$ m and a $\mathrm{GSD} = 1.08$ cm/px at altitude $= 20$ m.

A pixel grid format was used to simplify low-high orthoimage matching, where reference pixels are selected in the low-orthoimage with a constant interval, $GridSize$, in both image width and height directions. Fig. 5(a) shows an example of a 6×6-pixel grid in a 20×20-px low-orthoimage with the $GridSize = 3$-px. The $Margin = 2$-px in each edge is used to guarantee all selected reference pixels have enough reference patches. Then the 20×20-px low-orthoimage is cropped to a 16×16-px orthoimage. The number of selected pixels in the pixel grid is determined by Eq. (3), which is $[int(16/3) + 1]^2 = 36$

(3)

$$Num.\,of\,Pixels\,in\,PixelGrid = \left[int\left(\frac{Cropped\ Image\ Height}{GridSize}\right) + 1\right] \times \left[int\left(\frac{Cropped\ Image\ Width}{GridSize}\right) + 1\right]$$

Moreover, Fig. 5(b) shows an example of a 50×50-pixel grid for an assembled low-orthoimage, in which $GridSize = 32$-px, $Margin = 128$-px, the cropped orthoimage has the size of $1{,}568 \times 1{,}568$-px, and the number of selected pixels via Eq. (3) is $[int(1568/32) + 1]^2 = 2{,}500$. The previous *PGMED* was designed with a one-by-one matching scheme within a pixel grid (Jiang and Bai 2021). To take advantage of the multiprocessing of modern computing systems, in this paper, the reference pixels are divided into several groups, and as a result, the process of matching pixel groups can parallelly run in several CPU cores/threads. For example, a 2,500-pixel grid can be divided into six groups as $[0, \ldots, 416]$, $[417, \ldots, 833]$, $[834, \ldots, 1250]$, $[1251, \ldots, 1667]$, $[1668, \ldots, 2084]$, and $[2085, \ldots, 2499]$]. The 1st group [annotated in Fig. 5(b)] contains $[int(2{,}500/6) + 1] = 417$, which include the reference pixel "Start 0." The 2nd to 5th group also contains 417 pixels in each group. The 6th group (the last group) contains the remaining $(2{,}500 - 417 \times 5) = 415$ pixels, which include the reference pixel "End 2499."

Multiprocessing and Pixel Dictionary

A key process of *PGMED* is using neighboring pixels' elevations $Ele.$ as the guess elevation $Ele._{guess}$ to speed up the pixel matching operations; as a result, the matching process of flat sites is faster than sites with large elevation changes. When processing all reference pixels in a single CPU core/thread, temporarily saving matched results on the process's memory pool (like maintaining a spreadsheet) is a suitable approach to quickly access them for the remaining pixels' matching operation. Similarly, the matched results should be accessible from different CPU cores/threads in multiprocessing. One feasible approach is using shared memory for direct access across processes; another usable approach is saving matched results on a hard drive as temporary files for direct access across processes. This paper utilized the latter option of saving individual reference pixel's matching results in separate temporary files on

the hard drive rather than saving them in a common spreadsheet. This setting, named *Pixel Dictionary*, avoids the interruption of opening and writing the common file at the same time due to the requests of multiprocessing different pixel groups. Fig. 6 shows a pixel dictionary, where each spreadsheet file contains two pieces of information: (1) the file's name, e.g., "*20G19128,256DenseEleMap.csv*", indicates that the station is 10–20 G, patch radius $R = 19$, and the selected reference pixel $[v, u] = [128{,}256]$ (Fig. 3); and (2) inside each spreadsheet file, the first cell is the determined virtual elevation value Ele.Ele. for $p[v, u]$, and the second cell is the NCC score between the target patch prediction and matched target patch.

Fast-PGMED Algorithm

Fig. 7 shows the proposed *Fast-PGMED* algorithm for matching pixels and determining elevations with the virtual elevation model (Fig. 2). For each reference pixel in a pixel group, the matching loops start at $Ele._{guess}$ and simultaneously go to the upper virtual elevation values by plus increment $Ele._{step}$ and the lower virtual values by minus $Ele._{step}$. The best matching result is returned after all values in range $[Ele._{guess} - Ele._{range}, Ele._{guess} + Ele._{range}]$ are processed. Since similarly textured pixels would have close elevation values, an elevation guessing scheme was proposed to initialize $Ele._{guess}$ and narrow down the value of $Ele._{range}$. In a pixel grid, a reference pixel $p[v, u]$ has up to eight neighboring reference pixels with a distance of *GridSize* in column or/and row direction. If some of these neighbors are matched and the temporary matching results are saved in the pixel dictionary (like Fig. 6), the calculations of NSSDNSSD are conducted via Eq. (4) between $I_{ref}[v, u]$ (33×33-px window of the reference pixel $p[v, u]$) and $I_{neighbor}[v'', u'']$ (33×33-px window of a neighboring pixel $p''[v'', u'']$). To speed up NSSDNSSD calculation, both windows are extracted from the low-orthoimage and converted to 1-channel grayscale image patches

(4)

$$NSSD = \frac{\sum\left(I_{ref}[v, \mathrm{u}] - I_{neighbor}[v'', u'']\right)^2}{\sqrt{\sum\left(I_{ref}[v, \mathrm{u}]\right)^2 \times \sum\left(I_{neighbor}[v'', u'']\right)^2}}$$

Moreover, the following steps are proposed to determine $Ele._{guess}$ and $Ele._{range}$: (1) Find the most similarly textured neighboring pixel with the smallest *NSSD* value. (2) Set $Ele._{guess}$ and NCC_{guess} as the smallest *NSSD* neighboring pixel's values. (3) Choose an $Ele._{range}$ via the rules that if $NCC_{guess} > 0.25$, $Ele._{range} = 1$ m; otherwise, $Ele._{range} = 2.5$ m. (4) In case of no matched neighboring pixel in the pixel dictionary, set $Ele._{guess} = 0$ and $Ele._{range} = 5$ m.

Furthermore, after obtaining the matched pixel grid and determining the elevation values, the elevation map and *NCC* score map are created and saved as two separate 8-bit grayscale images (or spreadsheet with rows and columns, which are the same as the orthoimage's height and width) by assigning each determined pixel's elevation value and the calculated NCC value to neighboring pixels, respectively. Then, in the generated elevation map and NCC score map, each $\mathrm{Gri}dSize \times GridSize$ patch shares the same elevation value and NCC value. In addition, a median filter with the size of $(4GridSize + 1) \times (4GridSize + 1)$ is proposed to smooth the raw elevation map; thus, the noisy elevation values from the poorly matched pixels would be removed, while the edges of elevation changes are kept.

Parameter Analysis and Performance Evaluation

The proposed *DeepMatchNet* model and *Fast-PGMED* algorithm were programmed in Python 3.6.8 and ran with 2 × Xeon Gold 5122 CPUs (with 8 cores/16 threads) and 4 × GeForce RTX 2080 Ti GPUs for evaluation.

Image Acquisition and Assembly Evaluation

The researchers captured top-view images via a drone, *DJI Phantom 4 Pro V2.0*, on experimental sites A and B (Shorewood, WI). Images' GPS coordinates (see Table S1) were mapped as Fig. 8(a). The distribution of GPS altitude differences between the low and high images are shown in Fig. 9(f), which has $\mathrm{Q1} = 9.5\ \mathrm{m}$, $\mathrm{median} = 9.8\ \mathrm{m}$, and $\mathrm{Q3} = 9.9\ \mathrm{m}$ for the designed $H/2 = 10\ \mathrm{m}$ (low-high altitudes at 10–20 m) and $\mathrm{Q1} = 19.9\ \mathrm{m}$, $\mathrm{median} = 20.2\ \mathrm{m}$, and $\mathrm{Q3} = 20.4\ \mathrm{m}$ for $= 20\ \mathrm{mH}/2$ (low-high altitudes at 20–40 m). These results indicate most images were captured within the drone's hover accuracy range $\pm 0.5\ \mathrm{m}$ (GPS positioning), but low-high altitude differences are not equal to the designed value in the virtual elevation model (Fig. 2). Thus, a process of elevation alignment to GCPs is necessary, and the impacts of altitude difference are investigated and discussed later.

Low-high orthoimages were assembled via the steps stated in the *Pixel Grid* Section. In this paper, a pair of assembled low-high orthoimage was referenced as $10\text{-}20G$ or $20G$, in which 10 refers to the designed low altitude $H/2 = 10\ \mathrm{m}$, 20 is the designed high altitude $H = 20\ \mathrm{m}$, and G names the station. On experimental site A [see Figs. 8(a and b)], 25 pairs of 10-20 orthoimages and 14 pairs of 20-40 orthoimages were assigned as training low-high orthoimages (39 pairs in total, see Table S1). In addition, 11 pairs of 10-20 orthoimages and five pairs of 20-40 orthoimages (16 pairs in total) with previously matched pixel grids (Jiang et al. 2020; Jiang and Bai 2020a, b, 2021) were used for testing. Nine of them are located on experimental site B [which is much more complex than site A, see Figs. 8(a and b)], and the other seven pairs are on site A. Figs. 9(a–c) show the distributions of translations and rotations of the 55 low-high orthoimages, in which 10-20 C has the maximum absolute value of width translation of 48.75 pixels; 10-20 B has the maximum absolute value of height translation of 38.20 pixels and also has the maximum absolute value of rotation of 17.986°; and 20-40 BA has the largest 3,583 matched SIFT points. In addition, the assembled 10-20 orthoimages have the $\mathrm{Per}\ spectiveScale$ (see definition in Table 2) with $\mathrm{Q1} = 0.45658$, $\mathrm{median} = 0.49612$, and $\mathrm{Q3} = 0.53126$; which are different from the 20-40 orthoimage subsets with $\mathrm{Q1} = 0.46259$, $\mathrm{median} = 0.48271$, and $\mathrm{Q3} = 0.50342$ [Fig. 9(e)].

Table 2. Pearson correlations of assembly duration and other variables

Variable	Correlation	95% CI for ρ	P-value	Impaction	Comment
Absolute value of translation in image width direction	0.043	$(-0.225, 0.305)$	0.755	No	—
Absolute value of translation in image height direction	-0.218	$(-0.456, 0.051)$	0.111	No	—
Absolute value of rotation	-0.281	$(-0.508, -0.016)$	0.038	—	Existing two outliers
Absolute value of rotation, inliers	0.006	$(-0.264, 0.276)$	0.964	No	—
Rotation, inliers	0.010	$(-0.261, 0.280)$	0.941	No	—
Num of matched SIFT keypoint	0.148	$(-0.123, 0.397)$	0.282	No	—
$Perspective\ Scale = \frac{Perspective\ length\ of\ low\ image's\ diagonal\ line}{Length\ of\ high\ image's\ diagonal\ line}$	0.018	$(-0.249, 0.282)$	0.898	No	The ideal scale is 0.5, which means the low and high images have the 2:12:1 scaling ratio
$GPS\ Altitude\ Difference\ Ratio = \frac{(High_{GPS\ Altitude} - Low_{GPS\ Altitude}) - H/2}{H/2}$	-0.016	$(-0.280, 0.250)$	0.906	No	The designed H/2 is 10 m for 10-20 orthoimages, and 20 m for 20-40 orthoimages

The 55 low-high orthoimages have the median assembly duration of 4.604 s for 10-20 orthoimages and 4.213 s for 20-40 orthoimages [Fig. 9(d)]. The 10-20 C has the maximum assembly duration of 6.144 s, and 20-40 BA has the second-longest assembly duration of 5.924 s; and the 10-20CA and 10-20 B have the shortest assembly duration of about 2.71 s. Correlations of SIFT, translations, rotation, *PerspectiveScale*, and *GPS Altitude Difference Ratio* (see definition in Table 2) to assembly duration were summarized in Table 2. There were two rotation outliers of 10-20A and B ($-17.1752°$ and $-17.9860°$, respectively). After removing them, the correlations between the assembly duration and the rotation and its absolute values are as insignificant as the other variables. Therefore, in practicing the method in this paper, it is not necessary to strictly align the low-high top-views in image acquisition for reducing the assembly time, and the other impactions are discussed later.

Data Set Creation and Quality Evaluation

The 39-training low-high orthoimages were processed by *PGMED* with $GridSize = 32$-px. The matched 2,500-pixel grids are summarized in Table S1 and visualized in Fig. S1(a), in which the 2,500 reference pixels are distributed in each low-orthoimage with the constant spacing of $GridSize$.

The *PGMED* algorithm repeats the pixel grid matching in four rounds; in each round, the row-by-row sequence matching starts from one of the four corners of the low-orthoimage; thus, for each reference pixel, there are four matching results for enhancing a reference pixel's results. In Figs. 5(b) and S1, annotations used the following rules: Green pixels have good matching scores in all four-pixel matching processes and have the matching quality label "1234;" Cyan pixels have good matching scores in three of the four matching and the quality labels [123,124,134,234]; Blue pixels have good matching scores in two of the four matching and the quality labels [12,13,14,23,24,34]; Pink pixels have a good matching score in one of the four matching and the quality labels [1,2,3,4]; and the weakest matched pixels are annotated with Red pixels, which have all bad matching scores in all four-pixel matching processes, and have the quality label "0" (Jiang and Bai 2021). Based on that, in Table S1, the useful pixels were filtered in nonred pixels (green, cyan, blue, and pink) with the following steps: (1) remove nonred pixels that have distances to the low-orthoimage center ≤ 192-px because the pixel matching results may contain errors in the image center region due to the slight rotations and translations in capturing low-high orthoimages; (2) remove pixels that have NCC scores less than 0.25 for resized reference patches [Fig. 3(b)] and target patches [Fig. 3(a)]; and (3) remove outliers that have a large angle difference ($>$ $1°$), the angle differences are measured between reference lines (reference pixels to low-orthoimage center) and target lines (target pixels to high-orthoimage center).

During the presented screening processes, the 97,500 matched pixels (from the 39-training low-high orthoimages, and each has a 2,500-pixel grid) dropped to 75,824 nonred pixels [Fig. 10(b)], and then only 71,411 useful pixels were kept [Fig. 10(a)]. The 20-40 AK has the largest number of 2,387 useful pixels, the 10-20 O has the smallest number of 475 useful pixels, and the 10-20 AH, P, W, Z, and U have less than 1,000 useful pixels. The correlations of the number of useful pixels and other variables were examined in the 39-training low-high orthoimages and summarized in Table 3. The number of useful pixels was not impacted by translation and rotation but impacted by the absolute value of ($Perspective\ Scale - 0.5$) [Fig. 10(e)]. Either the scaling ratio is smaller or larger than the ideal value of 0.5 and has a negative correlation of $\rho = -0.677$ to the number of useful pixels. This is because the self-adaptive patch features in the *PGMED* were designed for the ideal case of a scaling ratio of 0.5 between the high and low images. The number of matched SIFT keypoints was not impacted by scaling, translation, and rotation as its design; however, the number of matched SIFT keypoints was much less

than the useful pixels. A paired t-test showed the mean value of useful pixels is greater than the mean value of SIFT keypoints at the 0.05 level of significance, and the two mean values' difference is greater than 612.55 with 95% confidence. In addition, previous work showed the matched SIFT keypoints were distributed in low-orthoimages with a sparse and irregular style compared to the pixel grid matching results (Jiang and Bai 2021). Therefore, this paper used the *PGMED* matched useful pixels alternative to matched SIFT keypoints for training data set creation. The developed LHPG (low-high orthoimage pixel grid matching) dataset is available on (Jiang 2021a).

Table 3. Pairwise Pearson correlations

Sample 1	Sample 2	Correlation	95% CI for ρ	P-value	Impaction
Number of matched SIFT keypoint	Number of useful pixels	0.449	$(0.156, 0.670)$	0.004	—
Absolute value of translation in image width direction	Number of useful pixels	0.121	$(-0.202, 0.421)$	0.463	No
Absolute value of translation in image height direction	Number of useful pixels	-0.050	$(-0.360, 0.269)$	0.760	No
Absolute value of rotation	Number of useful pixels	0.014	$(-0.302, 0.328)$	0.930	No
Absolute value of $(Perspective\ Scale - 0.5)$	Number of useful pixels	-0.677	$(-0.818, -0.459)$	0.000	Yes
Absolute value of translation in image width direction	Number of matched SIFT keypoint	-0.173	$(-0.463, 0.151)$	0.292	No
Absolute value of translation in image height direction	Number of matched SIFT keypoint	0.097	$(-0.225, 0.401)$	0.556	No
Absolute value of rotation	Number of matched SIFT keypoint	-0.056	(−0.365, 0.264)	0.735	No
Absolute value of $(Perspective\ Scale - 0.5)$	Number of matched SIFT keypoint	-0.201	$(-0.486, 0.122)$	0.219	No

Based on the *Patch Features* Section, the reference patches [Fig. 3(c)] were cropped from the RGB 3-channel low-orthoimages with the square window of 78×78-px, and the target patches [Fig. 3(a)] were cropped from the RGB 3-channel high-orthoimages with the size of 39×39-px. Fig. 11 shows four samples of reference and target patches. Fig. 11(a) was cropped from the "Start 0" pixel shown in Fig. 5(b), which was not included in the training data set because it is a red pixel. Fig. 11(b) is the "End 2499" pixel in Fig. 5(b) with the matching quality label "124;" Fig. 11(c) is the "Start" pixel of 20-40 AK [see Fig. S1(a)] with label "34;" and Fig. 11(d) is the "End" pixel of 20-40AK with label "1234." These three nonred reference and target patches would be included in the training data set if the useful pixel screening steps (1) to (3) were all satisfied. Thus, 71,411 useful pixels were used to create the training data sets for *DeepMatchNet* as a reference and target patches like Fig. 11. In addition, these reference

and target patches were rotated 90°, 180°, and 270° to augment model training data sets to as much as 285,644 samples.

Furthermore, the useful pixel screen was also conducted on the 11-testing 10-20 orthoimages, each of which has a matched 2,500-pixel grid. Results show the 23,825 useful pixels account for 86.6% of the total matched 27,500 pixels [Fig. 10(c)], which is much better than the 73.2% of the training data sets [Fig. 10(a)]. The testing orthoimages also have more percentage of green (strongest matched) pixels [Fig. 10(d)] than training orthoimages [Fig. 10(b)]. In addition, Fig. 10(e) shows the collected testing low-high orthoimages are much closer to the ideal scale ratio of 0.5 than training low-high orthoimages. These results confirmed a positive relationship between the absolute value of ($Perspective\ Scale - 0.5$) and the number of useful pixels shown in Table 3. In the testing low-high orthoimages, the 5 pairs of 20-40 orthoimages are closer to the ideal scale 0.5 than the 10-20 orthoimages, and the collected matched pixel grids had the different $GridSize = 24$-px; thus, the useful pixel screen was not performed on them.

DeepMatchNet Training and Overfitting Prevention

According to the *Model Configuration* Section, the *DeepMatchNet* was trained on 142,822 samples and validated on 142,822 samples (*validation_split=0.5*). In addition, this paper set $batch\ size = 64$ and $epoch = 100$ in model training. The initial training applied the *Configuration 0* ($C0$) of training 100 epochs without early stopping, which has the plots of training and validation loss and accuracy in Figs. 12(a and b). The training accuracy became stable after the 35th epoch and maintained at around 0.87 for the remaining epochs. However, the validation accuracy was not keeping stable but changed from 0.8 to 0.88, and the validation loss was not decreased like the training loss. Thus, the trained model with $C0$ was an overfitting model.

Early stopping (monitor model performance on a validation set and stop training when performance degrades) and adding dropout [stated as "randomly sets input units to zero with a frequency of rate at each step during training" in Chollet (2021)] are two common techniques for preventing overfitting in deep learning model training. This paper tried the following configurations $C2$-4 before the decision of using the configuration $C1$ [which trained the model with early stopping (monitoring validation loss for 10 epochs)]. $C2$ trained the model with early stopping (monitoring validation loss for 5 epochs). $C3$ trained the model with early stopping (monitoring validation accuracy for 10 epochs). $C4$ added dropout layers ($\text{ratio} = 0.5$) for all hidden layers and trained the modified model with early stopping (monitoring validation loss for 10 epochs). Figs. 12(c and d) show plots of training and validation loss and accuracy of configurations $C1$-4. The added dropout layers in $C4$ resulted in the model performance degradation in loss and accuracy compared to $C1$-3. For example, the training accuracy of $C4$ maintained at 0.8, which is less than the 0.86 of $C1$-3. Thus, the dropout layers impacted the performance of *DeepMatchNet*.

Moreover, the efficiency of monitoring validation accuracy and loss were compared in $C3$ and $C1$-2. The $C3$ had the longest model training time of 65 min for 32 epochs (about 122s/epoch); however, the validation loss was not decreased as the training loss did after the 13th epoch, which means overfitting occurred in $C3$ the same as $C0$. The $C2$ had the shortest model training time of 24 min for 11 epochs (about $131\ \text{s/epoch}$), and the $C1$ had the model training time of 37.5 min for 18 epochs (about 125s/epoch). The lowest validation loss of $C2$ occurred at the 6th epoch as 0.0022528, the lowest validation loss of $C1$ occurred at the 8th epoch as 0.0022131, and the lowest validation loss of C3

occurred at the 13th epoch as 0.0021550. Thus, in early epochs, the *DeepMatchNet* may not be well trained; in other words, adding epochs can improve the model performance on the validation set and make it more stable. During model training, model performance was judged via the Keras "metrics=['accuracy']" function (Chollet 2015), which has the best validation accuracy of 0.869, and the validation accuracy of 0.859 was for the saved model at the 18th epoch in $C1$. *DeepMatchNet* outputs are fuzzy RGB images (see FCN patches in Fig. 11) with the main features of ground truth (GT) target patches. Therefore, the remaining experiments were conducted with the saved *DeepMatchNet* in $C1$.

Additionally, 256 filters were used in layers *Conv 4* and 55. The modified model was trained with configurations $E1$ (with early stopping by monitoring validation accuracy for 5 epochs) and $E2$ (with early stopping by monitoring validation loss for 5 epochs). Figs. 12(e and f) compare the training and validation loss and accuracy of configurations $C1$-2 *and* $E1$-2. There is an overfitting issue present in $E1$ as the validation loss did not decrease as the training loss did after the 14th epoch. This common issue of $C3$ and $E1$ indicates early stopping by monitoring validation accuracy cannot avoid model overfitting. The doubled filters resulted in the average training time increasing to 149s/epoch, which is about a 20% increase compared to $C1$ with 128 filters. The model training of $E2$ spent 32.77 min and got the best validation accuracy of 0.867 at the 7th epoch and an ending validation accuracy of 0.850 at the 13th epoch, which are both less than $C1$. The downgraded performance indicates adding filters is not necessary for *DeepMatchNet* with 78×78-px RGB 3-channel inputs and 39×39-px RGB 3-channel outputs.

Furthermore, the time efficiency of the well-trained *DeepMatchNet* was evaluated in the collected training and testing low-high orthoimages with $GridSize = 32$, 16, and 8-px, which resulted in 2,500, 9,801, and 38,809 pixels in the pixel grid, respectively, via Eq. (3). The *DeepMatchNet* only used about 1.42, 5.47, and 21.88 s to generate 2,500, 9,801, and 38,809 target patch predictions with speeds of about 1,761, 1,791, and $1{,}773$ patch/s (time and speeds all are median values, individual values are listed in Table S2), respectively. These times were measured in the mass production, and additional time may be needed for a singular run or the initial running of mass production.

Pixel Matching and Performance Evaluation

The 39-training and 11-testing low-high orthoimages (which have the same scope as *Data Set Quality Evaluation* in Fig. 10) were processed by the developed *Fast-PGMED* algorithm with $GridSize = 32$-px, $Margin = 128$-px, and $CPUcores = 3$. That means each 2,500-pixel grid was divided into three groups and processed by three CPU cores/threads, in which the first two groups have $[int(2{,}500/3) + 1] = 834$ pixels in each group, and the third group has 832 pixels. Figs. 13(a and b) show distributions of pixel matching time and speed for the 50 low-high orthoimages (individual values are listed in Table S3), in which 20-40 orthoimages have a median time duration of 12.08 s, which is slightly slower than 10-20 orthoimages. This is reasonable because 20-40 orthoimages cover about four times more area than 10-20 orthoimages with more elevation changes. Compared to the pixel matching durations of 1.86 to 3.37 min via *PGMED* (Jiang and Bai 2021), the time efficiency has improved significantly.

Since the *Fast-PGMED* algorithm used the *DeepMatchNet* generated target patch predictions (FCN patches) to match target pixels (patches) in high-orthoimages, the following two objective criteria were measured for pixel matching performance evaluation (Fig. 11): (1) u and v coordinate differences (closer to 0 are better) between the matched pixels and GT target pixels; and (2) NCCNCC scores (closer to 1 is better) between the FCN patches and the matched patches. The distributions of differences

in u and v of the matched 2,500-pixel grids in the 50 low-high orthoimages are shown in Figs. S2(a and b), and the individual values for each pixel are listed in Tables S4 and S5, in which 10-20 AH, O, P, U, W, and Z have much worse results than the others. The previous section stated these six low-high orthoimages also have the smallest useful pixel. Figs. 14(a and b) show useful pixels have much smaller differences in u and v than nonuseful pixels in the 39-training and 11-testing low-high orthoimages. Thus, performance metrics were analyzed in useful pixels (the same scope as the model training data sets). Fig. 14(c) shows that in training data sets, 44.1% of the matched pixels are the same as GT target pixels, which is 42.8% in testing [Fig. 14(e)]. Since the subpixel coordinate was not considered for target pixels in this paper, any matched pixel within 1-px difference in u and v compared to GT is an acceptable result. Fig. 14(d) shows 91.8% of pixels are matched within 1-px compared to GT in training data sets, which is 90.3% in testing [Fig. 14(f)]. Thus, the *DeepMatchNet* outputs matched at least 90% of GT target patches both in the training and testing data sets.

Moreover, in Figs. S2(c and d), the distributions of NCCNCC scores of the useful pixels in the 50 low-high orthoimages show that 10-20AH has the worst $\mathrm{Q1} = 0.28$ in training and 10-20B has the worst $\mathrm{Q1} = 0.58$ in testing. The correlations between u and v differences and NCC score were analyzed for the matched 2,500-pixel grids, which have $\rho_{u\_NCC} = -0.498$ and $\rho_{v\_NCC} = -0.475$ in training and $\rho_{u\_NCC} = -0.352$ and $\rho_{v\_NCC} = -0.339$ in testing. The NCC score maps are shown in Fig. 15 for the 2,500-pixel grids with bad NCC scores, where pixels in the objects' boundaries and the richly textured regions (e.g., lumber platform edges in 10-20B and 20-40V, vegetation boundaries in 10-20 CJ, and 20-40V) have better NCC scores than the uniformly textured (textureless) regions (e.g., sand surface in 10-20AH, the umbrella surface in 10-20CG).

Furthermore, the 14-training 20-40 orthoimages were processed with different configurations of $CPUcores$ from 1 to 16, and the duration and average core speed are shown in Figs. 13(c and d) and individual values are listed in Table S6. As the number of used cores increased, the processing duration of matching a 2,500-pixel grid dropped from 31.297 s (with one core/thread) to 6.581 s (with eight cores/threads). Beyond that, dividing the 2,500 pixels into more than eight groups (and processing each group in a CPU core/thread) will not decrease the total processing time, which slightly increased to 6.842 s (with 16 cores/threads). In addition, for each CPU core/thread, its speed decreased as groups increased because the increasing groups lead to more pixels (starting pixels in each group) having no saved neighboring matching result in the pixel dictionary. Then, those starting pixels use the setting of $Ele._{guess} = 0$ and $Ele._{range} = 5$ m in matching potential target pixels, which costs more time than the remaining pixels in each group that have a narrow $Ele._{range}$ and a nonzero $Ele._{guess}$ via the elevation guessing scheme. Moreover, the rule of $NCC \geq \max(LowerFence, 0.25)$ was used to select well-matched (inner) pixels from the 2,500-pixel grid for point cloud creation (in *Experiment* section), where $LowerFence = Q1 - 1.5 \times (Q3 - Q1)$. The more pixels remaining in the point cloud means better matching quality for the pixel grid. Table 4 shows four hypothesis test results and concluded that the pixel matching was not impacted by the number of pixel groups.

Table 4. Summary of hypothesis tests

Subset	Type	Hypothesis	Alpha level	P-value	Test results
Two subsets, "Odd" contains cores/groups of 1,3,5,7,9,11,13,15;	2-sample t-test	Mean of "Odd" different from	0.05	0.886	Not enough evidence to conclude that

and “Even” contains 2,4,6,8,10,12,14,16		the mean of “Even”			the means differ at the 0.05 level of significance.
2-sample standard deviation test	Standard deviation of “Odd” different from the Standard deviation of “Even”	0.05	0.974		
Three subsets, “Odd” contains cores/groups of 1,3,5, and 7; “Even” contains 2,4,6, and 8; and “9+” contains 9 to 16	One-way ANOVA	Means of “odd,” “even,” and “9+” are different	0.05	0.959	
Standard deviations test	Standard deviations of “odd,” “even,” and “9+” are different	0.05	1.00		

Experimental Results and Discussion

The applications of elevation determination and earthwork estimation were both conducted on two sites of B and C that were different from site A (which collected the model training data sets), see Fig. 8. Experiments were conducted and results were comparatively analyzed following the workflows in Figs. 1(a and c).

3D Reconstruction and Elevation Measurement

The collected testing 10-20CI (altitude difference 9.9m) and 20-40 CI (altitude difference 20.4m) were processed by *Fast-PGMED* with a $GridSize$ of 32, 16, and 8-px to match the 2,500, 9,801, and 38,809-pixel grids, respectively. The processing time of low-high orthoimage assembly, target patch prediction generation via *DeepMatchNet*, and pixel matching and elevation determination via *Fast-PGMED* are listed in Table 5. Those dense reconstructions were much faster than *PGMED-32*, which spent 1.86 min for a 2,500-pixel grid of 10-20 CI, and *PGMED-24*, which spent 3.00 min for a 4,761-pixel grid of 20-40CI in Jiang and Bai (2021).

Table 5. Application processing time

Orthoimage		Pixel grid			Duration,s				Overall time	Pixel/s/core	Overall speed, px/s	Inner points
Data	Height, px	Width, px	Gridsize, px	Matched pixels	Core/group	Assembly	DeepMatchNet	Fast-PGMED				
10-20CI	1,568	1,568	32	2,500	3	5.56	3.14	12.67	21.37	65.77	116.99	24,88
10-20CI	1,568	1,568	16	9,801	3	5.56	5.90	46.94	58.40	69.60	167.83	9,742
10-20CI	1,568	1,568	8	38,809	3	5.56	22.77	174.16	202.49	74.28	191.66	38,601
10-20CI	1,568	1,568	8	38,809	8	5.56	22.77	74.98	103.31	64.70	375.66	38,588
20-40CI	1,568	1,568	32	2,500	3	5.32	3.15	13.19	21.66	63.18	115.42	2,400
20-40CI	1,568	1,568	16	9,801	3	5.32	5.89	47.51	58.72	68.76	166.91	9,379
20-40CI	1,568	1,568	8	38,809	3	5.32	22.12	183.16	210.60	70.63	184.28	37,200
20-40CI	1,568	1,568	8	38,809	8	5.32	22.12	77.10	104.54	62.92	371.24	37,196
10-20CA	1,568	2,176	32	3,450	3	3.94	3.70	17.43	25.07	65.98	137.61	3,377
10-20CA	1,568	2,176	16	13,563	3	3.94	7.90	64.04	75.88	70.60	178.74	13,285
10-20CA	1,568	2,176	8	53,781	3	3.94	31.50	249.59	285.03	71.83	188.69	52,788
10-20DA	1,568	2,464	32	3,900	3	10.86	4.00	19.08	33.94	68.13	114.91	3,819
10-20DA	1,568	2,480	16	15,444	3	10.86	9.32	73.07	93.25	70.45	165.62	14,025
10-20DA	1,568	2,480	8	61,267	3	10.86	35.45	284.44	330.75	71.80	185.24	59,952

The generated elevation maps were aligned to the same point and compared to *PGMED-32* ($GridSize = 32\text{-px}$) results. In 10-20 CI, Figs. 16(a and b) show *Fast-PGMED-32* has the best elevation results compared to *PGMED-32* with an average pixel elevation difference of -0.0112 m/px [Fig. 16(c)]. The average differences for *Fast-PGMED*-16 and -8 are -0.0481 and -0.0489 m/px, respectively. Both are less than 0.05 m (5 cm). In 20-40 CI, Figs. 16(e and f) show the *Fast-PGMED-8* has the best elevation results compared to *PGMED-24* with an average pixel elevation difference of -0.0464 m/px [Fig. 16(g)]. The average differences for *Fast-PGMED*-16 and -32 are -0.0513 and -0.0761 m/px, respectively. Both are slightly larger than 5 cm. In addition, *Fast-PGMED-8* returned 38,601 inner points for 10-20CI [Fig. 16(d)] and 37,200 inner points for 20-40 CI [Fig. 16(h)] from the matched 38,809-pixel grids within 3.5 min via three-pixel groups (three-CPU cores/threads). Processing time reduced by half with eight-pixel groups on eight CPU cores/threads. The number of inner points of the eight-pixel groups is very close to those of the three-pixel groups (Table 5), confirming the Table 4 hypothesis test's conclusion that the number of pixel groups has no impact on pixel matching.

Another improvement of *Fast-PGMED* is that cutting image to square shape became unnecessary in low-high orthoimage assembly, which increased the coverage area by 1.39 times (from $1{,}568 \times 1{,}568$-px to $1{,}568 \times 2{,}176$-px, like Fig. 17). Since the number of pixels increased, the *DeepMatchNet* spent slightly more time in prediction and the *Fast-PGMED* spent slightly more time in matching pixels (speed of pixel/s/core kept the same), yet the overall processing times were slightly increased compared to square-shaped low-high orthoimages. Figs. 16(i and j) show the X/Y-profiles of the testing 10-20CA (altitude difference 9.7 m), in which *Fast-PGMED-16* has the best elevation results compared to *PGMED-32* with an average pixel elevation difference of 0.0246 m/px [Fig. 16(k), only shows the overlap]. The average differences for *Fast-PGMED*-32 and -8 are 0.0253 and 0.0267 m/px, respectively. Both are less than 5cm.

Additionally, elevation measurements were conducted on testing 10-20 CA, CG, CI, and CJ and 20-40 CA and CI (see Fig. S3), and compared to the true elevation in Table 6. The slight altitude different from the designed $H/2$ did not impact the elevation determination of *Fast-PGMED*, which has elevation differences $[-2.76, 0.94]$ cm for the 10-20 orthoimages and -5.09, 6.82] cm for 20-40 orthoimages. These results of the 10-20 orthoimage less than 5 cm and 20-40 orthoimages slightly larger than 5 cm are the same as the pixelwise comparison results (Fig. 16). Moreover, the elevation comparison and measurement results show the *Fast-PGMED* works for 3D reconstruction sites with slopes in the range of 0 to 90°, including: (1) the flat surfaces with the flat lumber surfaces; (2) the vertical slopes at the edges of the lumber platform; and (3) other slopes on the stairs and ground surfaces. The testing of the trained *DeepMatchNet* with a different drone is discussed later. Moreover, the elevation comparison in Fig. 16 shows the *Fast-PGMED* has better performance in modeling the stairs, especially the first stair in [Fig. 16(b)] and better performance in tall-tree reconstruction, as shown in Fig. 16(g).

Table 6. Elevation measurement

Point-point (Fig. S3)	True elevation difference (cm) (Jiang and Bai 2021) (A)	Low-high orthoimage		PGMED (Jiang and Bai 2021)	Fast-PGMED (this paper)			
		Station	Altitude difference (m)	Elevation coordinate (m)	Elevation coordinate (m)	Measured difference (cm) (B)	Elevation difference (cm) (B-A)	Elevation error (cm)
C-A	17.78	10-20 CI	9.9	C(0.00)-A(-0.1765)	C(0.00)-A(-0.1765)	17.65	-0.13	0.13
A-B	81.28	10-20 CJ	10.2	A(0.8039)-B(0.00)	A(0.8039)-B(0.00)	80.39	-0.89	0.89
C-B	99.06	10-20 CI	9.9	C(0.00)-B(-1.0000)	C(0.00)-B(-1.0000)	100	0.94	0.94
20-40 CI	20.4	C(0.00)-B(-0.9804)	C(0.00)-B(-1.0588)	105.88	6.82	6.82		
D-C	361.95	20-40 CI	20.4	D(3.6471)-C(0.00)	D(3.5686)-C(0.00)	356.86	-5.09	5.09
G-E	106.68	10-20 CA	9.7	G(0.00)-E(-1.0784)	G(0.00)-E(-1.0392)	103.92	-2.76	2.76
20-40 CA	20.4	G(0.00)-E(-1.0588)	G(0.00)-E(-1.0588)	105.88	−0.8−0.8	0.8		
F-G	320.04	10-20 CG	9.7	F(3.1961)-G(0.00)	F(3.1961)-G(0.00)	319.61	-0.43	0.43
20-40 CA	20.4	F(3.1765)-G(0.00)	F(3.2549)-G(0.00)	325.49	5.45	5.45		

Volume Measurement and Earthwork Estimation

For volume calculation, this paper set each pixel as the unit base, which has the unit area of $\mathrm{GSD} \times \mathrm{GSD}$, and then multiplied the unit area by the elevation difference of each pixel between the design elevation and the surveyed elevation map to sum the earthwork estimations (Fig. 18). Thus, the horizontal scale (GSD) of the orthoimage and elevation map are important to the earthwork estimation accuracy. Fig. 9(f) and Table 6 indicate a high probability that the altitude difference of the collected low-high top-views is different from the designed value of $H/2$; thus, the actual GSD of the assembled low-orthoimage needs to be determined for volume estimation. In the case of using a drone landing pad as GCP: (1) for small sites, one or more landing pads can be placed in any place of the low-orthoimage covered area [Figs. 17(a) and 18(a)]; and (2) for large sites and linear projects, one or more landing pads can be placed in any station, and the orthoimages and elevation maps can be stitched and aligned station-by-station. Figs. 18(a and b) show an example of orthoimage and elevation map based volume estimation, in which the $\mathrm{X/Y}$ -coordinate's origin was updated to the U-net detected pad's center. The $GSD_{adj.} = 0.53\ \mathrm{cm/px}$ was calculated via $GSD_{adj.} = 75/(2\sqrt{Pixel_{GCP}/PI})$, where $Pixel_{GCP}$ is the pixel number of the pad in U-net predictions and 75 cm is the pad's diameter (Jiang and Bai 2021). Four corners of the lumber platform were picked up in the orthoimage, as shown in Fig. 18(a), then, a cut plane1 (elevation $= -1.0\ \mathrm{m}$) and a zero plane2 (elevation $= 0.0\ \mathrm{m}$) were set for the enclosed area. Ten trials were conducted for each design plane and summarized in Table 7. Those independent measurements via orthoimage and elevation map are stable and have an average cut volume of $35.795\ \mathrm{m}^3$ for plane1 with a standard deviation (SD) of 0.158 and an average net cut volume of $0.787\ \mathrm{m}^3$ for plane2 with a SD of 0.024.

Table 7. Volume measurements via elevation maps

Trial	10-20 CA						10-20 DA		
	Design plane1 elevation $= -1.0$ m			Design plane2 elevation $= 0.0$ m			Design plane3 elevation $= 0.0$ m		
	CUT, m^3	FILL, m^3	Net, m^3	CUT, m^3	FILL, m^3	Net, m^3	CUT, m^3	FILL, m^3	Net, m^3
1	35.80	0.00	$35.80 <$ Cut $>$	1.03	0.24	$0.79 <$ Cut $>$	68.679	1.305	$67.374 <$ Cut $>$
2	35.74	0.00	$35.74 <$ Cut $>$	1.02	0.29	$0.73 <$ Cut $>$	67.300	0.163	$67.137 <$ Cut $>$
3	35.67	0.00	$35.67 <$ Cut $>$	1.01	0.19	$0.82 <$ Cut $>$	68.687	1.180	$67.507 <$ Cut $>$
4	35.85	0.00	$35.85 <$ Cut $>$	1.01	0.22	$0.79 <$ Cut $>$	69.289	0.642	$68.647 <$ Cut $>$
5	36.16	0.00	$36.16 <$ Cut $>$	1.00	0.22	$0.78 <$ Cut $>$	69.732	0.523	$69.209 <$ Cut $>$
6	35.73	0.00	$35.73 <$ Cut $>$	1.01	0.21	$0.80 <$ Cut $>$	67.484	1.228	$66.256 <$ Cut $>$
7	35.66	0.00	$35.66 <$ Cut $>$	1.01	0.20	$0.81 <$ Cut $>$	68.470	0.630	$67.840 <$ Cut $>$
8	35.74	0.00	$35.74 <$ Cut $>$	0.99	0.20	$0.79 <$ Cut $>$	69.206	0.797	$68.409 <$ Cut $>$
9	35.95	0.00	$35.95 <$ Cut $>$	1.01	0.23	$0.78 <$ Cut $>$	69.380	0.610	$68.770 <$ Cut $>$
10	35.65	0.00	$35.65 <$ Cut $>$	0.97	0.19	$0.78 <$ Cut $>$	69.852	0.327	$69.525 <$ Cut $>$
Mean	35.795	—	35.795	1.006	0.219	0.787	68.808	0.741	68.067
Standard deviation	0.158	—	0.158	0.016	0.030	0.024	0.871	0.386	1.020

Table 8. Volume measurements via point clouds

Data	Point cloud	Area, m^2	Cut, m^3	Fill, m^3	Net, m^3
10-20 CA	Fast-PGMED-Plane1	35.63	0.00	35.71	$35.71 <$ Fill $>$
Compared between SfM point cloud and the inner points of Fast-PGMED-8					
SfM-Plane1	35.86	0.00	35.21	$35.21 <$ Fill $>$	
fixed-SfM-Plane1	35.86	0.00	36.08	$36.08 <$ Fill $>$	
Fast-PGMED-Plane2	35.63	0.58	0.66	$0.08 <$ Fill $>$	
SfM-Plane2	35.86	1.86	1.21	$0.65 <$ Cut $>$	
fixed-SfM-Plane2	35.86	0.30	0.52	$0.22 <$ Fill $>$	
10-20 DA	Fast-PGMED -Plane3	82.76	1.37	67.90	$66.53 <$ Fill $>$
Compared between SfM point cloud and the orthoimage and elevation map (via Fast-PGMED-32) converted point cloud					
SfM-Plane3	82.76	2.41	67.47	$65.06 <$ Fill $>$	
Fast-PGMED-SfM	162.01	10.06	18.28	$8.22 <$ Fill $>$	

Moreover, point cloud-based volume measurements were also conducted following the workflow in Fig. 1(a). The lumber surface was presented as a slope in the raw SfM photogrammetry result (via ReCap Photo), which was fixed via rotating it to a level plane (normal up). The plane2 was set to examine this correction and the flatness of the flat plane reconstruction. The volume measurement results (via Civil 3D) in Table 8 confirmed the SfM point cloud was successfully rotated and aligned with plane2, in which small cut and fill volumes were caused by the gaps and unevenness of the lumber surface. The *Fast-PGMED-8* produced 52,788 inner points, which well-modeled this flat surface, and the net fill volume of $0.08\ \mathrm{m}^3$ is smaller than the fixed SfM of $0.22\ \mathrm{m}^3$. In addition, *Fast-PGMED* only has a $0.37\ \mathrm{m}^3$ (1.03%) difference compared to the SfM point cloud for the cut plane1.

Additionally, the trained *DeepMatchNet* was also tested with *DJI Mavic 2 Pro* [Fig. 1(e)], which has different parameters compared to the drone used in model training data sets collection (Table 9). The 10-20DA was collected by a beginner at site C (Shaoguan, China) with an 11-m GPS altitude difference and noticed position shift [Fig. 8(c)], which resulted in image translations of 51.26 pixels in width and 25.55 px in height, and a rotation of 0.075 degrees in low-high orthoimage assembly. The *Fast-PGMED-32* generated elevation map and NCC score map are shown in Figs. 19(b and c). Since the landing pad was not placed on the site, the $\mathrm{GSD} = 0.65\ \mathrm{cm/px}$ was determined by measuring the two edges of the SfM point cloud [Fig. 19(e) which was cropped to the same region as the orthoimage] and the width and height of the orthoimage [Fig. 19(a)]. Then, the dense point cloud [Fig. 19(d)] was converted from the orthoimage and elevation map by selecting pixels at intervals of 8-px. Ten independent trials of volume measurements were conducted on the orthoimage and elevation map [Figs. 18(c and d)], which have an average net cut volume of 68.067 m368.067 m3 with a SD of 1.020 (Table 7).

Table 9. Drone specifications

Parameters	DJI Phantom 4 Pro V2.0	DJI Mavic 2 Pro
Price	USD $1,599	USD $1,599
Max flight time/max hovering time (no wind)	Approximately $30\ \mathrm{min}$/no data	31 min/29 min
Image size	$3{,}648 \times 4{,}864$ (used)	$3{,}648 \times 5{,}472$ pixel (default)
Sensor	8.8 mm (1-in. CMOS)	8.8 mm (1-in. CMOS)
Lens 35 mm format equivalent	24 mm	28 mm
Focal length	8.8 mm	Approximately $10.27\ \mathrm{mm}$($= 8.8/24 \times 28$)
GSD: original top-views at 10 m	Approximately $0.27\ \mathrm{cm/pixel}$[$= 10 \times 8.8/(8.8 \times 3{,}648)$]	Approximately $0.23\ \mathrm{cm/pixel}$[$= 10 \times 8.8/(10.27 \times 3{,}648)$]
GSD: assembled low-orthoimage at 10 m	Approximately $0.54\ \mathrm{cm/pixel}$($= 0.27 \times 2$)	Approximately $0.46\ \mathrm{cm/pixel}$($= 0.23 \times 2$)

Furthermore, the point cloud-based volume measurements were conducted in Civil 3D and summarized in Table 8. The design plane3 (elevation $= 0.0\ \mathrm{m}$) has an area of $82.76\ \mathrm{m}^2$, where the *Fast-PGMED* and SfM point clouds have a $1.04\ \mathrm{m}^3$ cut difference and a $0.43\ \mathrm{m}^3$ fill difference. In addition, these two-point clouds generated mesh surfaces that were compared in Civil 3D, which have a $10.06\ \mathrm{m}^3$ cut difference, $18.28\ \mathrm{m}^3$ fill difference, and $8.22\ \mathrm{m}^3$ net fill difference. Thus, for the overlapped region

of $162.01\ \mathrm{m}^2$, the two-point clouds have an average elevation difference of 0.05 m (5 cm). Therefore, the altitude difference of 11-m of 10-20 DA is larger than the designed $\mathrm{H}/2 = 10\text{-m}$ did not affect the Fast-PGMED, and the trained DeepMatchNet can be used in other drones.

Discussion, Performance Comparison, and Potential Application

Highly overlapped image series are essential raw data for SfM photogrammetry and make fast (or real-time) 3D reconstruction impossible. A pair of low-high top-views based SfM and dense reconstruction by VisualSFM (Wu 2011) is shown in Fig. 19(f), which took one minute (12 CPU threads) but got a useless result either for 3D reconstruction or volume estimation. In this paper, the SfM photogrammetry (via ReCap Photo) used 52 top-views and took 27.5 min to generate the point cloud for the lumber platform and used the additional processing to fix the rotation issue. The SfM photogrammetry used 99 images (setting the pitch-axis of the gimbal to negative 90° and 45°) and took 73 min to generate the point cloud of Fig. 19(e). These SfM point clouds are too dense and large to be smoothly ran in Civil 3D (on a workstation with Core i7-7800X and GeForce GTX 1080 Ti), and the additional process of decimation grid (spacing in 100 mm, via ReCap Pro) is needed to make it possible. In contrast, the *Fast-PGMED-8* only used a pair of low-high top-views and took 4.75 min to generate the point cloud for the lumber platform [Fig. 16(l)], and *Fast-PGMED-32* used low-high top-views and took 0.57 min to generate the point [Fig. 19(d)], see Table 5. Capturing stable top-views on a target station is very convenient with a ready-to-fly imaging drone, and any beginner can perform the following operation steps and get high-quality top-views of the target stations: (1) Placing a landing pad in the target station; if the scanned site is in excavation, put the landing pad on the nonexcavation area. (2) Launching the drone, moving and hovering it over the target station at the designed low altitude of $10\ \mathrm{m}/20\ \mathrm{m}$ (via reading the height from the drone controller). (3) Setting/checking the pitch-axis of the gimbal at negative 90° and taking the first image, then moving the drone to the high altitude of $20\ \mathrm{m}/40\ \mathrm{m}$ and taking the second image. Slight rotation and position differences will not impact the results. As the ready-to-fly drones have approximately 30 min flight time (Table 9), using multiple stations to cover a large site or linear project is possible, then the drone can fly in an “up-forward-down” path for moving between adjacent stations and take two top-views in either high-low or low-high altitude order like (Jiang and Bai 2021).

Additionally, volume estimation is more convenient for construction professionals and much faster to perform in the 2D orthoimages and elevation maps on construction sites via a mobile device and on the cloud (web) without any powerful graphics card and 3D-engine. When multiple stations’ top-views are collected, parallelly processing several low-high orthoimages is also possible. In the case of the *Fast-PGMED* with three CPU cores/threads, up to five low-high orthoimages can be multiprocessing on the workstation (eight CPU cores/16 threads) to generate five elevation maps simultaneously. The automatic stitching of overlapped 2D images is much easier than the merging of 3D point clouds. Thus, once two adjacent stations’ orthoimages are stitched, the stitching parameters can also be applied to merge and align elevation maps (Jiang 2021b). In addition, a pair of an orthoimage and elevation map has a much smaller file size than a point cloud for storage and wireless transmission, which is important and necessary for future deployment of the *DeepMatchNet* and *Fast-PGMED* on an onboard computer for real-time 3D reconstruction, such as monitoring excavation progress. Once a low-top view is captured, the *DeepMatchNet* can be activated to generate the target patch predictions for the 2,500-pixel grid during the drone’s movement. Once a high-top view is captured, the low-high orthoimage assembly can be activated immediately and followed by the *Fast-PGMED-32*. Thus, the overall time can be reduced to about 11.2 s (including 4.604 s for 10-20 orthoimage assembly [Fig. 9(d)], and 6.581 s for

pixel matching and elevation determination with eight CPU cores/threads [Fig. 13(c)]. The time efficiency can also be improved by coding the low-high orthoimage assembly and the *Fast-PGMED* algorithms in $\mathrm{C/C++}$ instead of Python.

Limitation and Recommendation

In this paper, the prepared model training data sets are samples of matched reference and target patches. The *DeepMatchNet* learned the image transformation features between the low and high altitudes' top-view images, alternative to the objects' elevations (Jiang and Bai 2020a). Thus, training *DeepMatchNet* with the collected training data sets on site A can generate accurate target patch predictions [at least 90% accuracy, see Fig. 14(f)] for *Fast-PGMED* to quickly match pixels and accurately determine elevations on site B. However, the NCC score map in Fig. 15(c) shows bad NCC scores, and the elevation map in Fig. 17(c) indicates bad 3D reconstruction results in part of the uniformly textured red umbrella surface in 10-20CG and the same for 20-40 CA in Fig. S3. Since there are several training data sets, like 20-40 AN, BA, and S [Fig. S1(a)] that contain similar umbrellas and have good pixel matching results, the initial guess of *DeepMatchNet* is not well trained, which was investigated and discussed. The training configuration *C3* [in DeepMatchNet training (Fig. 12), C3 was trained with 32 epochs and have the maximum training accuracy and the minimum training loss] saved model was used to generate target patch predictions for *Fast-PGMED-32* and returned the elevation map shown in Fig. 17(d), showing improvement for the umbrella but bad results for the lumber platform around the landing pad compared to the 20-40 CA results in Fig. S3. This change confirmed that C3 is an overfitting model. Since no training 10-20 orthoimage has a similarly shaped and textured umbrella, the overfitting model was useless in fixing the 10-20 CG [Fig. 17(c)]. Thus, adding more differently shaped and textured surfaces to model training data sets is necessary for the specific application.

Moreover, future research can consider the following additional approaches to address this issue and improve the 3D reconstruction performance of uniformly textured (textureless) surfaces: (1) Increasing the size of the patch feature. The size of the target patch and target patch prediction (39×39-px in this paper) may be too small to distinguish the best target patch from the other potential target patches via NCC scores due to their uniform texture and the same NCC score. The size self-adjusted patch feature was used in PGMED because increasing the patch size can enclose more neighboring nonumbrella pixels; as the results show, most of the umbrella pixels in training and testing data sets are well matched [see Figs. S1(a and b)]. The potential problem with using larger patch sizes is that it requires more RAM in model training, and requires more processing time for *Fast-PGMED* in target patch prediction and NCC calculation. Moreover, training the model with small patches and using the trained model to output large-sized target patch predictions cannot solve the issue, which was tested during this research. (2) Increasing the depth of the patch feature. This approach means adding more channels to *DeepMatchNet* outputs instead of using RGB 3-channel image patches, and then a reference pixel (in the pixel grid) will be presented as a multichannel feature map as (Choe et al. 2021). In addition, another parallel FCN is required to process the target patches to generate the equal channel feature map for each potential target pixel. In this case, using NCC scores to compare channels is not fast enough or useful, so a CNN can be used to match them and determine the matching quality level (Hughes et al. 2019). In the developed LHPG dataset, the training and testing samples have a hierarchic matching quality label {0, [1,2,3,4], [12,13,14,23,24,34], [123,124,134,234], 1234} for each matched pixel in the pixel grids that can be converted to the matching level of [0,10,100,1000].

Conclusion

This paper presents a fast and dense 3D reconstruction method, named fast pixel grid/group matching and elevation determination (*Fast-PGMED*) algorithm (Fig. 7), for construction site elevation determination and earthwork estimation using low-cost ready-to-fly imaging drones and deep learning technologies. The workflows are summarized in Fig. 1(c), and the performances of the new method are compared with existing SfM, SLAM, and *PGMED* methods and are listed in Table 10. Parameter analysis and experimental results also concluded that:

1. Training the proposed *DeepMatchNet* with early stopping configuration of monitoring validation loss for 10 epochs can prevent the overfitting issue and get the well-trained model, see Fig. 12.
2. Using the *DeepMatchNet* to generate target patch predictions (39×39-px) as reference pixels' features can match at least 90% of the GT target pixels via target patch (39×39-px), see Fig. 14.
3. Dividing a pixel grid into several pixel groups for multiprocessing on separate CPU cores/threads and temporarily saving the matched pixels in the proposed pixel dictionary for access across processes can rapidly match a 2,500-pixel grid in 12.08 s (using three cores/treads) and as fast as 6.581 s (using eight cores/threads), see Fig. 13.
4. Using the virtual elevation model to provide potential target pixels for matching is effective no matter the altitude difference, which is not the same as the designed value $H/2$ for the low-high top-views' acquisition, see Table 6.
5. Applying the image translation and rotation-based low-high orthoimage assembly can eliminate the impact of a drone's shifting and rotation during movement and make the acquisition of top-views easier.
6. Applying the trained *DeepMatchNet* for processing another drone's captured top-views can also have a good performance and have an average elevation difference of 5 cm compared to SfM photogrammetry.

The success of this research contributes to the advancement of the low-cost and ready-to-fly imaging drone-based construction site surveying method. Producing elevation and volume data from the developed *Fast-PGMED* algorithm is a time- and cost-effective and accurate solution for earthwork operations. Construction professionals can gather the construction site elevations in nearly real-time and remotely, which will not interfere with the other on-site construction operations. Furthermore, this research developed a benchmark dataset of LHPG, which can be downloaded from (Jiang 2021a).

Data Availability Statement

The training and testing data sets are available in (Jiang 2021a). The Python codes are available from the corresponding author upon reasonable request.

Table 10. Achievement in this research

Process	Performance	Existing methods	PGMED (Jiang and Bai 2021)	Fast-PGMED (this paper)
Data acquisition	Image type	Same-scale or multiscale top-view images and side-view images; contain objects' top and side surfaces of the scanned sites	Multiscale top-view images at a low and high position with designed altitude difference $H/2$; only contain objects' top surfaces	
	Flight altitude	Close range (about 5 m) in Visual SLAM	Low altitude at 10 m or 20 m for detailed 3D reconstruction, and can be extended to 70 m for coarse elevations	Low altitude at 10 m or 20 m for accurate, dense, and fast 3D reconstruction
	Image coverage	Covered by at least two adjacent images with the same scale	Assembled square (1:1 aspect ratio) low-high orthoimage with approximate 2:1 scale ratio overlap	Any aspect ratio low-high orthoimage with approximate 2:1 scale ratio; coverage area increased by 1.39 times compared to PGMED
	Image number	Highly overlapped image series are essential raw data for SfM and SLAM	Two top-view images per target station (about 160 m2160 m2 with 10-20 orthoimages in Fast-PGMED, see Fig. 19); multiple stations for the large-sized site and linear projects; the stitching of adjacent results only needs narrow overlaps	
	Safety and efficiency	Over construction sites via drone, move toward target objects via state-of-the-practice surveying techniques, e.g., GPS, total station	Hovering and away from construction sites; image acquisition requires less time, without interrupting other construction operations; suitable for obtaining as-built elevations and monitoring construction progress up to 30 min per battery life	
Data processing	Feature descriptor	Key point-based features, e.g., SIFT; robust in scaling, rotation, translation, and perspective transformation	Four-resized reference patches and a target patch with self-adaptive size; support for 2:1 scaling, translation, and brightness change	*DeepMatchNet* generated target patch prediction and target patch with fixed size; support for scaling, rotation, translation, and perspective transformation
	Feature matching	Matched key points are irregularly distributed and missed in edges and in regions of low contrast and variation	Matched pixel grid, where pixels are densely and uniformly distributed in the image; no missed pixels, as bad matches are "smoothed" via neighboring good matches	
	Preprocessing	Uses SIFT key points for sparse reconstruction by SfM	Uses SIFT key points for low-high orthoimage assembly via translation and rotation; cutting	

			to square shape is not necessary for Fast-PGMED, see Fig. 17	
	Dense reconstruction	PMVS/CMVS; requires a powerful workstation with multi-core/thread CPUs	PGMED uses four CPU cores/threads to match pixels (one-by-one) in the pixel grid (each core/thread starts from different corners) and determines pixels' elevations simultaneously	Fast-PGMED divides a pixel grid into several pixel groups, and each group runs on a CPU core/thread, see Fig. 7
	Processing time	Depends on image numbers; two images took 1 min with 12 threads via VisualSFM; 52 images took 27.5 min and 99 images took 73 min via Autodesk ReCap Photo on servers	PGMED-32 spent 1.86 min to 3.37 min for 2,500−pixel2,500−pixel grids; PGMED-24 spent 2.66 min to 3.18 min for 4,761-pixel grids	38,809-p ixel grids took 3.5 min via three cores and reduced to 1.73 min with eight cores, about 70 px/s/thread70 px/s/thread for pixel matching (Table 5); 2,500-pixel grids took about 21 s (including 5 s for low-high orthoimage assembly, 3 s for patch feature generation, and 13 s for pixel matching with three threads), and as fast as 11.2 s, including 4.604 s for assembly [Fig. 9(d)], and 6.581 s for pixel matching with eight threads [Fig. 13(c)]
	Reconstruction quality	Two images-based dense point cloud is not useful, see Fig. 19(f)	A pair of low-high orthoimages-based elevation maps can be used for elevation measurement and volume estimation (in rough estimate)	
Modeling and measurement	Modeling	3D point cloud and mesh model	(Stitched) 2D orthoimage (24-bit RGB image) and elevation map (8-bit grayscale image), 3D point cloud and mesh model	
	Alignment	Requires precise GPS, or at least three GCPs	Detects a known size drone landing pad (via U-net) to adjust the GSD for orthoimages and elevation map, and updates the elevation coordinate origin to the pad's center, see Fig. 18	
	Elevation measurement	Measures on point cloud or mesh model	Measures on orthoimage and elevation map; due to the 8-bit grayscale format, the elevation maps have systematic errors of 0.0196 m for 10-20 orthoimage and 0.0392 m for 20-40 orthoimage; accurate elevations	

			can be saved on a spreadsheet file like the pixel dictionary (see Fig. 6)	
	Elevation accuracy	Elevation within 5 cm error via photogrammetry; SLAM has an average error of 3.3 cm compared to photogrammetry but has poor performance while exceeding the measurement range of the depth camera	Elevation within 5 cm error via 10-20 and 20-40 orthoimages	Elevation within 5cm error via 10-20 orthoimages; elevation error [−5.09−5.09, 6.82] via 20-40 orthoimages (Table 6)
	Volume measurement	Volume estimation based on the surveyed mesh surface and the designed mesh surface; requires a powerful workstation with graphics cards	Fast volume estimation based on elevation difference in each pixel and each pixel's coverage area (GSD×GSDGSD×GSD) of orthoimage and elevation map; accurate volume estimation based on mesh surfaces	
	Volume accuracy	For a flat lumber surface (gaps existing), Fast-PGMED's net fill volume of 0.08 m30.08 m3 is smaller than SfM's 0.22 m30.22 m3; For the cut design with the lumber surface, Fast-PGMED only has a 0.37 m30.37 m3 (1.03%) difference compared to SfM point cloud; for a soil stack with an area of 82.76 m2m2, the Fast-PGMED and SfM point clouds have a 1.04 m31.04 m3 cut difference and a 0.43 m30.43 m3 fill difference (see Table 8)		

Acknowledgments

This research was financially supported by the McShane Endowment fund at Marquette University. The authors are thankful to Mr. Weijing Nie (graduate student at Guangzhou University) for his help in capturing images at experimental site C. In addition, the authors are grateful to the reviewers for their valuable comments and feedback.

Supplemental Materials

Figs. S1–S3 and Tables S1–S6 are available online in the ASCE Library (www.ascelibrary.org).

Supplemental Materials

- supplemental_materials_co.1943-7862.0002256_han.zip (48 MB)

References

Aguilar, R., M. F. Noel, and L. F. Ramos. 2019. "Integration of reverse engineering and non-linear numerical analysis for the seismic assessment of historical adobe buildings." *Autom. Constr.* 98 (Feb): 1–15. https://doi.org/10.1016/j.autcon.2018.11.010.

Arias, P., J. Herráez, H. Lorenzo, and C. Ordóñez. 2005. "Control of structural problems in cultural heritage monuments using close-range photogrammetry and computer methods." *Comput. Struct.* 83 (21–22): 1754–1766. https://doi.org/10.1016/j.compstruc.2005.02.018.

Barazzetti, L., F. Remondino, and M. Scaioni. 2010. "Automation in 3D reconstruction: Results on different kinds of close-range blocks." *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* XXXVIII (5): 55–61.

Bay, H., A. Ess, T. Tuytelaars, and L. Van Gool. 2008. "Speeded-up robust features (SURF)." *Comput. Vision Image Understanding* 110 (3): 346–359. https://doi.org/10.1016/j.cviu.2007.09.014.

Bethmann, F., and T. Luhmann. 2015. "Semi-global matching in object space." *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* XL-3/W2 (3W2): 23–30, https://doi.org/10.5194/isprsarchives-XL-3-W2-23-2015.

Chen, K., W. Lu, F. Xue, P. Tang, and L. H. Li. 2018. "Automatic building information model reconstruction in high-density urban areas: Augmenting multi-source data with architectural knowledge." *Autom. Constr.* 93: 22–34. https://doi.org/10.1016/j.autcon.2018.05.009.

Choe, J., K. Joo, F. Rameau, and I. S. Kweon. 2021. *Stereo object matching network*. New York: IEEE. https://doi.org/10.1109/ICRA48506.2021.9562027.

Chollet, F. 2015. "Keras: The python deep learning library." Accessed March 7, 2020. https://keras.io/.

Chollet, F. 2021. "Dropout layer." Accessed October 10, 2021. https://keras.io/api/layers/regularization_layers/dropout.

DJI. 2020. "Phantom 4 Pro V2.0." Accessed March 7, 2020. https://www.dji.com/phantom-4-pro-v2/specs.

DJI. 2021. "Mavic 2." Accessed April 27, 2021. https://www.dji.com/mavic-2/info#specs.

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://doi.org/10.1016/j.autcon.2006.11.002.

Furukawa, Y., and J. Ponce. 2010. "Accurate, dense, and robust multiview stereopsis." *IEEE Trans. Pattern Anal. Mach. Intell.* 32 (8): 1362–1376. https://doi.org/10.1109/TPAMI.2009.161.

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.” *IOP Conf. Ser. Mater. Sci. Eng.* 301: 012145. https://doi.org/10.1088/1757-899X/301/1/012145.

Hughes, L. H., D. Marcos, S. Lobry, D. Tuia, and M. Schmitt. 2020. “A deep learning framework for matching of SAR and optical imagery.” *ISPRS J. Photogramm. Remote Sens.* 169: 166–179. https://doi.org/10.1016/j.isprsjprs.2020.09.012.

Hughes, L. H., N. Merkle, T. Burgmann, S. Auer, and M. Schmitt. 2019. “Deep learning for SAR-optical image matching.” In Proc., IGARSS 2019-2019 IEEE Int. Geoscience and Remote Sensing Symp., 4877–4880. New York: IEEE.

Inzerillo, L., G. Di Mino, and R. Roberts. 2018. “Image-based 3D reconstruction using traditional and UAV datasets for analysis of road pavement distress.” *Autom. Constr.* 96: 457–469. https://doi.org/10.1016/j.autcon.2018.10.010.

Jiang, Y. 2021a. “Automated ortho-image and elevation-map stitching.” Accessed May 11, 2021. https://www.yuhanjiang.com/research/DT/PGMED/Stitching.

Jiang, Y. 2021b. “Dataset.” Accessed May 5, 2021. https://www.yuhanjiang.com/dataset.

Jiang, Y., and Y. Bai. 2020a. “Determination of construction site elevations using drone technology.” In Proc., Construction Research Congress 2020, 296–305. 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://doi.org/10.1061/(ASCE)CO.1943-7862.0001869.

Jiang, Y., and Y. Bai. 2021. “Low–high orthoimage pairs-based 3d reconstruction for elevation determination using drone.” *J. Constr. Eng. Manage.* 147 (9): 04021097. https://doi.org/10.1061/(ASCE)CO.1943-7862.0002067.

Jiang, Y., Y. Bai, and S. Han. 2020. “Determining ground elevations covered by vegetation on construction sites using drone-based orthoimage and convolutional neural network.” *J. Comput. Civ. Eng.* 34 (6): 04020049. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000930.

Kaehler, A., and G. Bradski. 2016. *Learning OpenCV 3: Computer vision in C++ with the OpenCV library*. Sebastopol, CA: O’Reilly Media.

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

Knyaz, V. A., O. Vygolov, V. V. Kniaz, Y. Vizilter, V. Gorbatsevich, T. Luhmann, and N. Conen. 2017. “Deep learning of convolutional auto-encoder for image matching and 3D object reconstruction in the infrared range.” In Proc., 2017 IEEE Int. Conf. on Computer Vision Workshops (ICCVW), 2155–2164. New York: IEEE.

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: 21–28. https://doi.org/10.1016/j.proeng.2017.07.168.

Lewis, J. P. 1995. “Fast template matching.” In Proc., Vision Interface 95, 120–123. Quebec City: Canadian Image Processing and Pattern Recognition Society. http://scribblethink.org/Work/nvisionInterface/vi95_lewis.pdf.

Li, D., and M. Lu. 2018. “Integrating geometric models, site images and GIS based on Google earth and keyhole markup language.” *Autom. Constr.* 89: 317–331. https://doi.org/10.1016/j.autcon.2018.02.002.

Li, G., L. Yu, and S. Fei. 2021. “A deep-learning real-time visual SLAM system based on multi-task feature extraction network and self-supervised feature points.” *Measurement* 168: 108403. https://doi.org/10.1016/j.measurement.2020.108403.

Liu, Y., Y. Li, L. Dai, C. Yang, L. Wei, T. Lai, and R. Chen. 2021. “Robust feature matching via advanced neighborhood topology consensus.” *Neurocomputing* 421: 273–284. https://doi.org/10.1016/j.neucom.2020.09.047.

Luo, W., A. G. Schwing, and R. Urtasun. 2016. “Efficient deep learning for stereo matching.” In Proc., 2016 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 5695–5703. New York: IEEE.

Memarzadeh, M., M. Golparvar-Fard, and J. C. Niebles. 2013. “Automated 2D detection of construction equipment and workers from site video streams using histograms of oriented gradients and colors.” *Autom. Constr.* 32: 24–37. https://doi.org/10.1016/j.autcon.2012.12.002.

Moon, D., S. Chung, S. Kwon, J. Seo, and J. Shin. 2019. “Comparison and utilization of point cloud generated from photogrammetry and laser scanning: 3D world model for smart heavy equipment planning.” *Autom. Constr.* 98: 322–331, https://doi.org/10.1016/j.autcon.2018.07.020.

Mughal, M. H., M. J. Khokhar, and M. Shahzad. 2021. “Assisting UAV localization via deep contextual image matching.” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* 14: 2445–2457. https://doi.org/10.1109/JSTARS.2021.3054832.

Nair, V., and G. Hinton. 2010. “Rectified linear units improve restricted Boltzmann machines.” In Proc., 27th Int. Conf. on Machine Learning, 807–814. Madison, WI: Omnipress. https://doi.org/doi/10.5555/3104322.3104425.

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

Nex, F., and F. Remondino. 2014. “UAV for 3D mapping applications: A review.” In *Applied geomatics*. New York: Springer.

OpenCV. 2018a. “Feature matching + homography to find objects.” Accessed March 7, 2020. https://docs.opencv.org/3.4.12/d1/de0/tutorial_py_feature_homography.html.

OpenCV. 2018b. “Template matching.” Accessed March 7, 2020. https://docs.opencv.org/3.4.2/d8/dd1/tutorial_js_template_matching.html.

Park, J., P. Kim, Y. K. Cho, and J. Kang. 2019. “Framework for automated registration of UAV and UGV point clouds using local features in images.” *Autom. Constr.* 98: 175–182. https://doi.org/10.1016/j.autcon.2018.11.024.

Rublee, E., V. Rabaud, K. Konolige, and G. Bradski. 2011. “ORB: An efficient alternative to SIFT or SURF.” In Proc., 2011 Int. Conf. on Computer Vision, 2564–2571. New York: IEEE.

Shang, Z., and Z. Shen. 2018. “Real-time 3D reconstruction on construction site using visual SLAM and UAV.” In Proc., Construction Research Congress 2018, 305–315. Reston, VA: ASCE.

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

Snavely, N. 2010. “Bundler: Structure from motion (SfM) for unordered image collections.” Accessed March 7, 2020. http://www.cs.cornell.edu/~snavely/bundler/#S1.

Sung, C., and P. Y. Kim. 2016. “3D terrain reconstruction of construction sites using a stereo camera.” *Autom. Constr.* 64: 65–77. https://doi.org/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., ISPRS—Int. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 371–377. Hannover, Germany: International Society for Photogrammetry and Remote Sensing. https://doi.org/10.5194/isprs-archives-XLII-2-W6-371-2017.

Wang, G., and Y. Chen. 2021. “Robust feature matching using guided local outlier factor.” *Pattern Recogn.* 117 (Sep): 107986. https://doi.org/10.1016/j.patcog.2021.107986.
Wu, C. 2007. “SiftGPU: A GPU implementation of Scale Invariant Feature Transform (SIFT).” Accessed March 7, 2020. http://cs.unc.edu/~ccwu/siftgpu.
Wu, C. 2011. “‘VisualSFM: A visual structure from motion system.” Accessed March 22, 2021. http://ccwu.me/vsfm/index.html.
Wu, C., S. Agarwal, B. Curless, and S. M. Seitz. 2011. “Multicore bundle adjustment.” In Proc., CVPR 2011, 3057–3064. New York: IEEE.
Yang, C.-H., M.-H. Tsai, S.-C. Kang, and C.-Y. Hung. 2018. “UAV path planning method for digital terrain model reconstruction—A debris fan example.” *Autom. Constr.* 93: 214–230, https://doi.org/10.1016/j.autcon.2018.05.024.
Zhu, H., L. Jiao, W. Ma, F. Liu, and W. Zhao. 2019. “A novel neural network for remote sensing image matching.” *IEEE Trans. Neural Networks Learn. Syst.* 30 (9): 2853–2865. https://doi.org/10.1109/TNNLS.2018.2888757.