12-2021

Building and Infrastructure Defect Detection and Visualization Using Drone and Deep Learning Technologies

Yuhan Jiang

Sisi Han

Yong Bai

Journal of Performance of Constructed Facilities, Vol. 35, No. 6 (December 2021). DOI.

Building and Infrastructure Defect Detection and Visualization Using Drone and Deep Learning Technologies

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

Abstract

This paper presents an accurate and stable method for object and defect detection and visualization on building and infrastructural facilities. This method uses drones and cameras to collect three-dimensional (3D) point clouds via photogrammetry, and uses orthographic or arbitrary views of the

target objects to generate the feature images of points' spectral, elevation, and normal features. U-Net is implemented in the pixelwise segmentation for object and defect detection using multiple feature images. This method was validated on four applications, including on-site path detection, pavement cracking detection, highway slope detection, and building facade window detection. The comparative experimental results confirmed that U-Net with multiple features has a better pixelwise segmentation performance than separately using each single feature. The developed method can implement object and defect detection with different shapes, including striped objects, thin objects, recurring and regularly shaped objects, and bulky objects, which will improve the accuracy and efficiency of inspection, assessment, and management of buildings and infrastructural facilities.

Introduction

In the architecture, engineering, and construction (AEC) industry, the most common case of object detection is automatic identification of cracks on buildings, structures, and pavements. Previous studies of crack detection are primarily based on two-dimensional (2D) images captured by handheld, vehicle-mounted, and drone-mounted cameras (Ali et al. 2019; Dadrasjavan et al. 2019; Dorafshan et al. 2019; Liu et al. 2020), while some are based on three-dimensional (3D) images, which were generated by laser line profile sensors (Edmondson et al. 2019; Zhang et al. 2019; Zhou and Song 2020a, b) or converted from 3D laser scanning point clouds and photogrammetric point clouds (Edmondson et al. 2019; Roberts et al. 2020). Using scattered 2D images for cracking detection yields good results for each isolated target area; however, it has a high probability to underrate cracks and thus it is impossible to comprehensively represent the current condition of an entire building element or infrastructure component, such as an interior wall, a section of pavement, or a deck slab. However, photogrammetry has shown the advantages of 2D reconstruction in generating high-resolution orthophotos (Dadrasjavan et al. 2019) and 3D reconstruction in generating dense point clouds (Edmondson et al. 2019; Roberts et al. 2020) and 3D mesh models (Kalfarisi et al. 2020), which are scaled and measurable models of the scanned buildings and infrastructural facilities. With these real models, the defect detection, quantification, and visualization of the scanned targets could be done in a comprehensive and continuous way, unlike the interval and fragmentary method of using multiple 2D images.

Furthermore, previous studies of building and infrastructure components and defect detection via deep learning and $2\mathrm{D}/3\mathrm{D}$ images were separately using either the red, green, blue (RGB) color feature or the elevation feature as the input. Then multiple convolutional operations are used to generate complex feature maps based on the input $2\mathrm{D}/3\mathrm{D}$ images. However, the traditional machine learning methods of support vector machine (SVM) classifier (Dadrasjavan et al. 2019) and random forest (RF) classifier (Li et al. 2019; Shi et al. 2016) prefer multiple structured features, such as the combination of spectral, textural, and structural features extracted from drone photogrammetric orthophotos for asphalt pavement crack/noncrack classification (Dadrasjavan et al. 2019), and the integration of elevation, reflection intensity, multiscale roughness index, multiscale Gaussian curvature, and several object-oriented geometric features extracted from the unmanned aerial vehicle (UAV) lidar point cloud for potholes, subsidence, and undamaged pavement classification (Li et al. 2019). Thus, it is worth evaluating whether the performance of deep learning–based object detection, semantic, and instance

segmentation methods can be improved by using structured multiple features in the case of AEC scenes.

Therefore, this research project adopted photogrammetry point clouds in building and infrastructure object and defect detection, quantification, and visualization, and comparatively evaluated the performances of RGB, digital elevation model (DEM) (elevation), and Normal (indicates the shape of the surroundings) features of photogrammetry point clouds and their combinations in deep learning–based pixelwise segmentation. The remainder of this paper is organized as "Literature Review," which summarizes the deep learning (neural networks)–driven building and infrastructure objects and defects detection methods; "Methodology," which presents the detailed procedures and algorithms of utilizing a deep learning model to process the multiple feature images and yield the pixelwise segmentation results; "Performance Comparison" and "Applications and Discussions," which present four AEC application experimental results and evaluate the performance of the developed method; and "Conclusion," which summarizes the findings and limitations of this research project.

Literature Review

This section summarizes the reviewed studies that relate to deep learning–driven image segmentation for AEC static scenes.

Deep Learning for Image Segmentation

In general, the orthographic faces of building and infrastructure components are relatively flat planes, such as building facades, pavement surfaces, and deck slabs; thus, it is feasible to use a single frame of large-sized 2D image, such as the photogrammetric orthophoto (Dadrasjavan et al. 2019), to continuously represent their spectral features (red, green, blue) or to use a single frame of large-sized 3D image, such as range image (Zhou and Song 2020a, b), 3D pavement image (Hsieh and Tsai 2020), surface height plot (Edmondson et al. 2019), or depth map (Roberts et al. 2020), to represent their structural features, especially the elevation feature. Moreover, the convolutional neural networks (CNNs)–based image classification and patch-wise segmentation (Ali et al. 2019; Fan et al. 2019; Jiang et al. 2020, 2021; Maniat 2019; Protopapadakis et al. 2019; Yang et al. 2020; Zhou and Song 2020a) and fully convolutional networks (FCNs)–based pixelwise segmentation (Alipour et al. 2019; Augustaukas and Lipnickas 2019; Dung and Anh 2019; Ji et al. 2020; Liu et al. 2019; Song et al. 2020; Zou et al. 2019) are the most common deep learning approaches that can be used for detection of objects and defects from the $2D/3D$ images in AEC.

Specifically, a CNN model starts with a convolutional layer; its hidden layers contain multiple max-pooling, convolutional, and dense layers. The CNN model generally ends with a dense layer with a softmax activation function for classification (Jiang et al. 2020). For image segmentation, CNNs could use a sliding window scheme (or overlapping small patches) (Jiang et al. 2020; Protopapadakis et al. 2019) to classify crack/noncrack (Ali et al. 2019; Jiang et al. 2021; Maniat 2019; Protopapadakis et al. 2019; Zhou and Song 2020a), pavement cracking categories (Maniat 2019), and construction site objects (Jiang et al. 2020) in each small patch of large-resolution $2D/3D$ images. In contrast, FCNs typically use convolutional and deconvolutional layers to generate and explain feature maps; use max-pooling and up-sampling layers to resize feature maps and keep the main features after convolutional and deconvolutional layers; use the rectified linear unit (ReLU) activation function in hidden layers for

faster model training; use dropout layers to prevent overfitting; and use merging layers to combine the feature maps (tensors) from two different layers as a new feature map (tensors) (Chollet 2020), such as the element-wise addition layers used in FCN (Shelhamer et al. 2017) and channel concatenation layers used in U-Net (Ronneberger et al. 2015). For image segmentation, a softmax function in the ending convolutional layer of an FCN is able to obtain the probability of each pixel belonging to the predefined classes, such as crack and noncrack (Badrinarayanan et al. 2017; Song et al. 2020).

Based on the literature review, using multiple features is a blank area in the previous studies of CNNs and FCNs, which separately use either a spectral feature (2D imagery) or elevation feature (3D imagery) as model input data, while the traditional SVM classifier (Dadrasjavan et al. 2019) and RF classifier (Li et al. 2019; Shi et al. 2016) yielded better results by using multiple features as opposed to using a singular feature. Thus, a comprehensive performance comparison is required to evaluate the multiple features in AEC object detection via deep learning methods. Therefore, this research project proposed the approach of using orthographic or arbitrary views of a photogrammetric point cloud to create the RGB, elevation, and normal feature image for the target building and infrastructure components, which are originally linked in the same pixel coordinate.

FCNs for Pixelwise Segmentation

FCN (Shelhamer et al. 2017), U-Net (Ronneberger et al. 2015), SegNet (Badrinarayanan et al. 2017), and DeepLabv3+ (Chen et al. 2018), which were designed for image semantic segmentation tasks, have been adopted in AEC for object and defect detections with 2D images. Alipour et al. (2019) developed CrackPix for pixelwise crack detection based on FCN and reached a pixel accuracy of 92.1% for detecting concrete cracks in images of bridge surfaces, building walls and slabs, and sidewalk surfaces. U-Net is an FCN architecture for biomedical image semantic segmentation that has been adopted in concrete crack detection (Liu et al. 2019) and pavement crack detection and has reached a pixel accuracy of 98.92% and an intersection of union (IoU) of 0.4850 (Augustaukas and Lipnickas 2019). U-Net has been used as the generator for CrackGAN (Zhang et al. 2020) because U-Net has the advantage of reaching a higher accuracy with smaller training data sets (images and ground-truth labels) (Liu et al. 2019; Zhang et al. 2020). SegNet is a deep convolutional encoder-decoder architecture for pixelwise segmentation that has been adopted in identifying road networks in large forested areas from RapidEye satellite imagery (Kearney et al. 2020). In addition, DeepLabv3+ has been utilized for crack detection on asphalt pavement (Ji et al. 2020).

Furthermore, Dung and Anh (2019) developed a convolutional encoder-decoder for concrete crack image semantic segmentation with an average precision of 89.3% in testing; the encoder block contains convolutional and max-pooling layers, while the decoder block contains up-sampling layers and both convolutional and deconvolutional layers, which is different from DeconvNet (only uses deconvolutional layers in decoder) (Noh et al. 2015) and SegNet (only uses convolutional layers in decoder); and the end convolutional layer using the softmax function is the same as SegNet. DeepCrack (Zou et al. 2019) is modified from SegNet and added skip-layer fusion (contains channel concatenation, convolutional, deconvolutional, and crop layer) to connect the encoder and decoder networks. Its output is a 1-channel prediction map that indicates the probability of each pixel belonging to the crack by using a cross-entropy loss. In addition, CrackSeg (Song et al. 2020) focuses on road crack detection and achieved a precision of 98.0% and a mean IoU of 73.5%. It has a multiscale dilated convolution

module (Yu and Koltun 2015) for generating rich crack features, and also has an up-sampling module to restore crack feature maps to the input image size and to predict the crack spatial distribution with the softmax function. Moreover, the comparisons of CrackSeg and other models on the same CrackDataset (Song et al. 2020) show that CrackSeg has the best performance in pavement crack detection followed by DeepCrack, DeepLabv3+, PSPNet (Zhao et al. 2017), U-Net, and SegNet; however, the differences among them are not significant.

Based on the literature review, adopting U-Net in AEC object detection could be more reasonable. That is because U-Net showed good performance in thin objects detection (Majidifard et al. 2020) and U-Net requires much fewer training data sets than other FCNs, which can be easily and separately trained for different AEC applications, with different weather and illumination conditions and different photography devices. Therefore, this research project proposed four experiments to comprehensively evaluate U-Net with multiple features in detection of differently shaped AEC objects, which include striped objects, thin objects, recurring and regularly shaped objects, and bulky objects.

Methodology

This section presents the overall procedure of the proposed method (Fig. 1), including the acquisition of photogrammetric point cloud feature images and U-Net model training and testing setups with an example of path/nonpath detection.

Photogrammetric Point Cloud Feature Image Acquisition

Photogrammetric Point Cloud

The photogrammetry generated point clouds have multiple features for each point, such as RGB, elevation, intensity, and normal, which can be displayed in different colors in Autodesk ReCap Pro. In detail, the RGB option displays points with colors, which are captured by a camera; the elevation represents points' heights (or Z-coordinates); the intensity measures point reflectivity based on surface texture, surface angle, and the environment; and the normal option displays points with colors, which are associated with the direction of the normal for each point (Autodesk 2020).

Additionally, dense point clouds can be easily zoomed in and zoomed out, or rotated to the desired orientation. For AEC scenes, the top, front, back, right, and left views contain the most components of buildings and infrastructural facilities. The roof conditions of buildings and the pavement condition of roadways and bridge decks are accessible in top views; the conditions of building facades and interior walls, bridge abutments and piers, and roadway cut and fill slopes are visible in side views; and the conditions of ceiling and deck slab are contained in bottom views.

Furthermore, these views can be switched between the perspective and orthographic modes. When the perspective mode is turned off, the displayed and exported orthographic top view of an RGB point cloud is similar to the large-size orthophoto in Dadrasjavan et al. (2019) and the colorized 3D point cloud map in McLaughlin et al. (2020), the top view of an elevation point cloud is similar to the range image in Zhou and Song (2020a, b), and these feature images are originally linked in the same pixel coordinate. In addition, the exported normal feature image is a good supplement to the RGB and elevation feature images because it indicates a point and its surrounding surface's shape, which is especially important when large slope surfaces appear as a narrow strip in orthographic views.

Feature Image and Label Image Acquisition

Fig. 2 shows two sets of photogrammetric point cloud feature images that were generated via the proposed workflow in Fig. 1. Two sets of overlapping top views of an experimental site were captured by a drone (DJI Phantom 4 Pro V2.0, SZ DJI Technology, Shenzhen, China). Then these two image sets were separately imported into a photogrammetry software (Autodesk ReCap Photo 21.0) to generate point clouds. After that, the point clouds were separately imported into Autodesk ReCap Pro to export the two sets of feature images (in orthographic top view), which includes two $4{,}096 \times 4{,}096$ pixels RGB spectral feature images; two $4{,}096 \times 4{,}096$ pixels DEM elevation feature images, which have elevation range $[-4, 4]$ m for the grayscale value [0, 255], and the drone takeoff pad as elevation ± 0.00; and two $4{,}096 \times 4{,}096$ pixels Normal feature images, where the colors of points indicate their normal directions. The intensity feature was not used in this research project because it overlaps with the normal feature.

Moreover, the two $4{,}096 \times 4{,}096$ pixels labels were manually crafted via the developed tool in Jiang et al. (2020), where the white (pixel value $= 255$) regions indicate the wooden paths on the experimental site, and the black (pixel value=0value=0) regions are nonpath objects. In Fig. 2, the exported RGB and DEM feature images are close to the orthophoto and DEM, but not exactly the same. That is because these feature images have gaps among points, while the orthophotos are smooth and textured images without gaps.

Pixelwise Segmentation Model Training and Testing

Model Training Data Sets Preparation

In general, considering the limitation of graphics processing unit (GPU) memory, the modern FCN models, such as U-Net, would be training with small-sized image and label data sets. However, they require much less GPU memory in the model prediction stage, and the well-trained models would be able to process some large-sized image inputs. Thus, the researchers' initial plan was to divide each $4{,}096 \times 4{,}096$ pixels feature image in Fig. 2 into 16 non-overlapped $1{,}024 \times 1{,}024$ pixels images. For each feature image, the four central small images were set as the model testing data sets, and the other 12 small images were used as the model training data sets.

However, the experimental deep learning workstation [which was equipped with 4×11 GB memory GeForce RTX 2080 Ti GPUs (Nvidia Corporation, Santa Clara, California)] was insufficient for handling the U-Net with $1{,}024 \times 1{,}024$ pixels images (up to seven channels) both at the model training and prediction stages. Thus, the researchers adopted the input images 50% overlapping disassembling and outputs 50% overlapping assembling algorithm in Jiang and Bai (2020) to supplement the U-Net-based pixelwise segmentation model for processing large-sized feature images at the model training and prediction stages. Table 1 lists the number of small-patch model training data sets (feature image and label) generated from each set of feature image and label in Fig. 2 via the 50% overlapping disassembling. The 128×128, 256×256, and 512×512 pixels slide windows were moved with the strides of 64, 128, and 256 pixels on a $1{,}024 \times 1{,}024$ pixels feature image, respectively, to generate the small-patch model training data sets. In addition, each $1{,}024 \times 1{,}024$ pixels feature image was rotated 90°, 180°, and 270° for data augmentation.

Table 1. Number of model training data sets

Patch size (pixels)	Number from a $1{,}024 \times 1{,}024$ pixels feature image, $A = (2H/patch\ size - 1)(2W/patch\ size - 1)$	Number with 90°, 180°, and 270° rotations, $B = A \times 4$	Total model training data sets, $C = B \times 12 \times 2$
128×128	225	900	21,600
256×256	49	196	4,704
512×512	9	36	864

Furthermore, this research project evaluated multiple features in pixelwise segmentation; thus, seven combination sets of multiple feature images were assembled as Table 2. For example, an assembled 128×128 pixels RGB + DEM feature image has four channels, where the first to third channels represent the R, G, and B, and the fourth channel represents elevations (DEM). This 4-channel feature image was linked with a 1-channel label image as one model training data set.

Table 2. Shape of model training inputs and labels

Feature	Feature type	Model training inputs		Model training labels	
		Number, width, height	Channel	Number, width, height	Channel
DEM	Single		1		1
DEM + Normal	Multiple	21,600, 128, 128	4	21,600, 128, 128	1
Normal	Single	3	1		
RGB	Single	4,704, 256, 256	3	14,704, 256, 256	1
RGB + DEM	Multiple	4	1		
RGB + DEM + Normal	Multiple	864, 512, 512	7	1,864, 512, 512	1
RGB + Normal	Multiple	6	1		

Model Setup and Training Configuration

The U-Net-based pixelwise segmentation model was set up with software packages of Keras 2.3.1, Python 3.6.8, OpenCV 3.4.2, and TensorFlow-GPU 1.14. In the wooden path/nonpath detection case, 21 U-Net models were trained with 21 model training data sets, respectively, which included seven sets of features and three different small-patch sizes (Table 2).

The researchers used the following common configurations for model training and validation, while avoiding model overfitting: *model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = [IoU_calc, 'accuracy']*; *callbacks = [EarlyStopping(monitor = 'val_loss', patience = 10)]*, which means the model training will be stopped when the validation loss is not significantly changed in the latest 10 epochs; *epochs = 100*, which means the model training will be stopped at the 100th epoch; and *validation_split = 0.1*, which means 10% of data sets will be used for model validation during model training. In addition, the researchers set *batch_size = 16* (due to insufficient GPU memory) for 512×512 pixels, *batch_size = 64* for 512×512 pixels, and *batch_size = 256* for 128×128 pixels small patches in U-Net model training.

Model Prediction and Postprocess

In the model testing and prediction stage, a large-sized feature image was first disassembled into multiple small patches, each having 50% overlaps to the adjacent small patches. Then, for each large-sized feature image, the U-Net model processed the disassembled small patches (equal number to column A in Table 1) rather than directly processing the large-sized input itself.

Furthermore, the U-Net model generated small-patch outputs that were 50% overlapping assembled via the algorithm in Jiang and Bai (2020). Thus, the assembled U-Net prediction had the same dimension as the large-sized input image, which was a 1-channel pixelwise segmented label image with a pixel value range of 0 to 255. Moreover, in the assembled predictions, all pixels with value $\geq 255/2$ were updated to 255 to indicate the wooden path, and otherwise replaced with 0 to represent nonpath objects. The modified pixelwise segmentations are referenced as modified U-Net predictions in this paper.

Performance Comparison

This section presents the comparative results of 21 trials of U-Net model training and testing for the wooden path/nonpath detection case, which includes all seven sets of features and three different small-patch sizes for U-Net model input and output.

Model Training and Testing

Model Training and Validation

The U-Net model training and validation loss and IoU of 128×128 pixels small patches are shown in Fig. 3, where the multiple features of DEM + Normal and RGB + DEM + Normal trials had better performance (lower validation loss and higher validation IoU) than the single feature of DEM and RGB. In addition, the same conclusion is verified by model validation results of 512×512 pixels and 512×512 pixels trials in Fig. 4. Therefore, using multiple features is more robust than a single feature in classifying the wooden path and nonpath pixels when gaps exist among points.

Model Testing and Prediction Assembly

The eight $1{,}024 \times 1{,}024$ pixels model testing data sets and the two sets of $4{,}096 \times 4{,}096$ pixels feature images (Fig. 2) were input into the 21 trained models for generating the small-patch pixelwise segmentations, which were used for assembling the large-size predictions, such as the RGB + Normal examples in Fig. 5. For each patch size, the number of patches is listed in Table 3. Between the two assembling options, the 50% overlapping assemblies had better results than the side-by-side results.

Table 3. Number of patches in prediction assembly

Patch size (pixels)	$1{,}024 \times 1{,}024$ pixels feature image		$4{,}096 \times 4{,}096$ pixels feature image	
	Side by side	50% overlapping	Side by side	50% overlapping
128×128	64	225	1,024	3,969
256×256	16	49	256	961
512×512	4	9	64	225

Pixelwise Segmentation Results Evaluation

Evaluation Metric

The 50% overlapping assembled pixelwise segmentation results of each model training and testing trial were evaluated in pixel accuracy, average pixel accuracy, IoU, average IoU, and mean IoU [Eqs. (1)–(5)]. The evaluation results of all 21 trials are shown in Fig. 6, where the testing group represents the eight $1{,}024 \times 1{,}024$ pixels testing data sets, and the overall group represents the two $4{,}096 \times 4{,}096$ pixels data sets (containing both model training and testing data sets). The evaluation results of mean IoU of the overall group matched the 128×128 pixels small-patch model training results in Fig. 3. The evaluation results of the testing group showed that the multiple features of RGB + Normal, RGB + DEM, and RGB + DEM + Normal have better results than other feature combinations

(1)

$$\text{For each data: Pixel Accuracy} = \frac{\text{Number of Pixels}_{\text{Prediction=Label}}}{\text{Image}_{\text{Height}} \times \text{Image}_{\text{Width}}}$$

(2)

$$\text{For overall or testing group: Average Pixel Accuracy} = \frac{\sum \text{Pixel Accuracy}}{\text{Number of Data}}$$

(3)

$$\text{For each data: IoU} = \frac{\text{Area of Overlap}_{\text{Prediction} \cap \text{Label}}}{\text{Area of Union}_{\text{Prediction} \cup \text{Label}}}$$

(4)

$$\text{For overall or testing group of each class: Average IoU} = \frac{\sum \text{IoU}}{\text{Number of Data}}$$

(5)

$$\text{For overall or testing group: Mean IoU} = \frac{\sum \text{Average IoU}}{\text{Number of Classes}}$$

Patch Size and Feature Comparison and Discussion

The evaluation results of the testing group are summarized in categories of patch size and categories of feature, respectively, in Fig. 7. For the comparison of patch sizes, the 128×128 pixels small patch has the best performance but is not significantly different than the other two patch sizes. For features, RGB + Normal had the best performance, increasing about 2% in the average of path and nonpath IoUs from the RGB, RGB + DEM, and RGB + DEM + Normal features. The DEM feature had the worst performance: using it alone led to a 17% decrease in the average of path and nonpath IoUs from the RGB features, but it did not have negative impacts on the combination features.

Furthermore, hypothesis tests were conducted and claimed that in the eight model testing data sets the multiple-features group (sample size: 96 for pixel accuracy, 181 for path and nonpath IoUs) had better performance than the single-feature group (sample size: 72 for pixel accuracy, 137 for IoUs) both in pixel accuracy and IoU. In detail, the 2-sample t-tests were conducted as null hypothesis,

mean(multiple)–mean(single) ≤ 0; alternative hypothesis, mean(multiple)–mean(single) > 0; α level $= 0.05$; p-value $= 0.014 < \alpha$ for pixel accuracy, and p-value $= 0.048 < \alpha$ for IoU, which both can claim their mean(multiple) > mean(single); and the 95% lower bounds for mean(multiple)–mean(single) are 0.0060038 and 0.00066584, respectively.

Therefore, to get the best performance of binary pixelwise segmentation with a U-Net model, it is worth using multiple features of photogrammetric point clouds, such as RGB + DEM when the elevations are available, or RGB + Normal when the points' normal directions are available. This is because the elevation data are a meaningful feature to the cases of construction sites and roadways, while they are a useless feature in the building facade case.

Applications and Discussions

In this section, a pavement cracking detection application is conducted for further comparing the seven sets of singular or multiple features. Then a highway slope detection application and a window detection (on building facades) application are conducted to further compare the RGB and RGB + Normal features, where elevations change quickly on target object surfaces. Moreover, several transfer learning applications are conducted and discussed with the trained models.

Pavement Cracking Detection

Crack Detection Model Training

Five sets of pavement images (with multiple cracks) were captured by a smartphone [iPhone SE second generation (Apple, Cupertino, California), positioned about 1.5 m from the pavement surface]. These image sets were imported into ReCap Photo for photogrammetry, like the example in Fig. 8. Then the generated point clouds were imported into ReCap Pro to export the features images of RGB, DEM [−0.2, 0.2] m, and Normal. In addition, fourteen $1{,}024 \times 1{,}024$ pixels features images and manually crafted cracking/noncrack labels were used as the U-Net model training and validation data sets, like the example in Fig. 9. Moreover, the researchers used 128×128 pixels small patches, which means each $1{,}024 \times 1{,}024$ pixels image data generated 900 small-patch data for model training with image rotations (data augmentation), and set *batch_size* = *128* and kept the other parameters the same as the "Model Setup and Training Configuration" section.

Feature Comparison and Discussion

The U-Net model validation results of different features are shown in Fig. 10. The 98 validation results (14 training data sets by seven features, each being $1{,}024 \times 1{,}024$ pixels) have an average pixel accuracy of 0.9817, average noncrack IoU of 0.9813, and average cracking IoU of 0.5728. Among the seven features (models), RGB + DEM has the best pixelwise cracking segmentation performance, followed by RGB + DEM + Normal, RGB, DEM + Normal, and RGB + Normal, which have better average cracking IoU performances than the single feature of Normal and DEM. Moreover, the common outlier in Fig. 10 is the data P4_2_2, which has the best cracking IoU (0.3178) from the multiple-feature RGB + DEM (Fig. 9), and second best (0.3123) from RGB + Normal. These are much better than the cracking IoUs of 0.0831, 0.1538, and 0.2286 from DEM, DEM + Normal, and Normal, respectively. Therefore, adding the features of points' elevations (DEM) or points' normal directions (Normal) to the RGB feature can increase the performance of U-Net-based pixelwise segmentation in pavement cracking detection compared to the RGB feature alone.

Transfer Learning and Discussion

The seven trained pavement cracking detection U-Net models were used to detect cracks on a pedestrian crossing in Fig. 8. The testing results among the seven trained U-Net models showed that the RGB + Normal model has the best performance in detecting alligator cracking. The model also has a good performance in detecting sealed cracks even though no sealed cracks were labeled in the model training data sets. Moreover, there are other pavement defects, such as potholes, rutting, and raveling, that could feasibly be detected from the multiple features and yield better performance than separately using each singular feature of RGB (2D imagery) or DEM (3D imagery) because those pavement distresses are rated by area and severity; for example, the severity of rutting is described in terms of depth (Stacks 2019).

Furthermore, for obtaining the detailed shape of pavement surfaces [depth resolution of 0.1 mm, transverse and longitudinal spacing resolution of 1 to 2 mm, as the laser line profile sensor (Zhou and Song 2020a, b)], the drone should be flying in close range to pavement surfaces. According to the online ground sample distance (GSD) calculator (Propeller Aero 2018), a drone's (DJI Phantom 4 Pro V2.0, built-in camera) flight heights of 2, 4, 7, 14, 18, and 36 m have corresponding GSDs of 0.05, 0.11, 0.19, 0.38, 0.49, and 0.99 cm/pixel0.99 cm/pixel, respectively. Thus, the minimum altitude for guaranteeing pavement defect quantification accuracy needs to be determined for different types of sensors, including optical cameras, hyper spectral cameras (HSCs), thermal imaging cameras, or infrared night vision cameras. Furthermore, determining flying altitude for both accuracy and safety (minimizing impact on drivers) needs to be addressed in future research.

Highway Slope Detection

Slope Detection Model Training

Two sets of highway images were collected from a drone photogrammetry highway demo (Pix4D 2018a). Both have 100 images, which were imported into ReCap Photo for photogrammetry. Then the generated point clouds were imported into ReCap Pro to export the RGB and Normal feature images from the orthographic top views of the two point clouds. In addition, twelve $4{,}096 \times 4{,}096$ pixels feature images and manually crafted highway slope/nonslope labels were created (like in Fig. 11), of which seven were used as the U-Net model training and validation data sets; the other five were used for testing. Moreover, the researchers used the 256×256 pixels small patches, and the data sets were not rotated for data augmentation, meaning each $4{,}096 \times 4{,}096$ pixels data generated 961 data for model training. Other parameters were kept the same as the "Model Setup and Training Configuration" section.

Feature Comparison and Discussion

The U-Net model validation and testing results of different features are shown in Fig. 12. The 15 testing results (five testing data sets by three features, each with 4,096 × 4,096 pixels) have an average pixel accuracy of 0.9301, an average nonslope IoU of 0.9205, and an average slope IoU of 0.6008; the 21 validation results (seven training data sets by three features, each with 4,096 × 4,096 pixels) have an average pixel accuracy of 0.9883, an average nonslope IoU of 0.9861, and an average slope IoU of 0.9253. Among the three features, the Normal and RGB + Normal have much better pixelwise segmentation (slope detection) performance than RGB.

Furthermore, the paired t-test results of IoU (sample size 24, 12 training and testing data sets by two classes) showed that mean(RGB + Normal) > mean(RGB) (p-value $= 0.003$) and mean(Normal) > mean(RGB) (p-value $= 0.006$); however, there is not enough evidence to conclude that mean(RGB + Normal) and mean(Normal) differ at the 0.05 level of significance (p-value $= 0.656$). Additionally, Fig. 11 shows the U-Net model validation results of data H_4 and testing results of data H_1, where the annotated slopes show that RGB + Normal has better performance results than Normal. Therefore, combining the features of RGB and Normal can increase the performance of U-Net-based pixelwise segmentation in highway slopes detection.

Transfer Learning and Discussion

The trained U-Net-based highway slope detection models were tested with an arbitrary view of a quarry site (Pix4D 2018c) (Fig. 13), where the detected slopes were annotated. The RGB feature detected most areas of the quarry site as slopes, the Normal feature only detected surfaces with specific colors (normal features) as slopes, while the RGB + Normal feature detected the most accurate slopes. Due to the collected images (drone flew at constant altitude, and camera faced ground, see Fig. 13) having insufficient side views of the quarry site, the vertical slopes are shown as gaps in the feature images; thus, while most vertical slopes are missed in the dense point cloud, the occurring slopes are well annotated in the Normal and RGB + Normal results.

Window Detection on Building Facade

Window Detection Model Training

Two sets of building facade images were captured by a smartphone (iPhone SE second generation, facing buildings), and a set of drone-captured images of Michigan Central Station (Pix4D 2018b) was collected. These three image sets were imported into ReCap Photo for photogrammetry. Then the generated point clouds were imported into ReCap Pro to export the feature images of RGB and Normal from the front, back, right, and left views (if they were present, like the example in Fig. 14). In addition, eight $2{,}048 \times 2{,}048$ pixels feature images and manually crafted window/nonwindow labels were used as the U-Net model training and validation data sets. Moreover, the researchers used the 128×128 pixels small patches, which means each $2{,}048 \times 2{,}048$ pixels data generated 3,844 data for model training with image rotations, and the other parameters were kept the same as the "Model Setup and Training Configuration" section.

Feature Comparison and Discussion

The U-Net model validation results of different features are shown in Fig. 15. The 24 validation results (eight training data sets by three features, each with $2{,}048 \times 2{,}048\ pixels$) have average pixel accuracy of 0.9843, average nonwindow IoU of 0.9829, and average window IoU of 0.8127. Among the three features, RGB + Normal has the best pixelwise window segmentation performance, followed by RGB, which are much better than the results of Normal.

Furthermore, a paired t-test was conducted between IoUs of RGB and RGB + Normal (sample size 16, eight data by two classes): null hypothesis, mean(RGB + Normal)–mean(RGB) ≤ 0; alternative hypothesis, mean(RGB + Normal)–mean(RGB) > 0; α level $= 0.05$; p-value $= 0.043 < \alpha$, which can claim mean(RGB + Normal) > mean(RGB); and 95% lower bound for mean(RGB + Normal)–mean(RGB) is 0.00024077. Moreover, the arbitrary view testing results in Fig. 14 show the RGB + Normal feature (model) detected more windows than the RGB results. Therefore, adding the feature of points' normal

directions to the RGB feature can increase the performance of U-Net-based pixelwise segmentation (window detection) on building facades.

Transfer Learning and Discussion

The well-trained U-Net models were tested with the four full-size data sets of Michigan Central Station ($4{,}096 \times 4{,}096\ pixels$), where the detected windows were annotated in the RGB feature images (Fig. 16). The model training data sets only included the bottom-left part ($2{,}048 \times 2{,}048\ pixels$) of Data Sets S1, S2, and S3, and the bottom half ($2{,}048 \times 4{,}096\ pixels$) of Data Set S4; for Data Sets S1 and S2, the RGB feature yielded the better window detection results in visual, while the RGB + Normal feature yielded the better results in Data Sets S3 and S4.

Furthermore, the researchers adopted the image perspective transformation–based data augmentation approach (Jiang 2020b) for retraining the RGB model to enhance its performance of window detection on arbitrary views. In addition, the researchers developed a Python application to continuously capture on-screen feature images and conduct the instance segmentation with the retrained RGB U-Net model (Jiang 2020c). In Fig. 17, the RGB feature images were directly snapped from the window of ReCap Pro (not via exporting), and almost every visible window was precisely detected and annotated at the instance segmentation level, where outlines of each singular window object were extracted via the *Contours* functions in OpenCV (2020). In addition, sequence numbers of several starting windows were displayed next to them (all can be displayed, while showing less to keep figures clean). Moreover, the comparison with the results in Fig. 14 also show that the retrained RGB U-Net yielded the better pixelwise segmentation result in the arbitrary view. Thus, the image perspective transformation–based data augmentation approach successfully made the U-Net model work with photogrammetric point clouds in arbitrary views, even though the manually prepared model training data sets were the same as the previous models (which only contain a few orthographic views of the feature images and labels).

Conclusion

The researchers of this project studied the use of multiple features of photogrammetric point clouds in object detection (image segmentation) in the architecture, engineering, and construction industry. The utilized deep learning model is U-Net, which can achieve a high accuracy and has the advantage of requiring relatively few model training data sets of images and labels. The hypothesis testing results show the evaluation metrics of pixel accuracy and IoU have been significantly improved from using a single-feature input by integrating multiple-feature input for path/nonpath detection. In addition, the comparative analysis results of the four applications of objects and defects detection show that integrating either point's elevations or normal directions (which indicate the shape of the surroundings) with the point's spectral (red, green, blue) features can improve the performance of pixelwise segmentation in feature images (generated from orthographic and arbitrary views of point clouds) than only with the point's spectral features via U-Net. Moreover, the researchers conducted experiments with differently shaped objects detection, including path detection (striped objects, RGB + Normal has the best pixelwise segmentation performance, average testing pixel $\text{accuracy} = 0.9782$, path and nonpath $\text{IoU} = 0.8107$), pavement cracking detection (thin objects, RGB + DEM has the best pixelwise cracking segmentation performance, average validation noncrack $\text{IoU} = 0.9869$, cracking $\text{IoU} = 0.7254$), highway slope detection (bulky objects, RGB + Normal has a better pixelwise

segmentation performance, average testing nonslope $\mathrm{IoU} = 0.9368$, slope $\mathrm{IoU} = 0.6239$), and building facade window detection (recurring, small and regularly shaped objects, RGB + Normal has the best pixelwise window segmentation performance, average validation pixel $\mathrm{accuracy} = 0.9910$, nonwindow $\mathrm{IoU} = 0.9901$, and window $\mathrm{IoU} = 0.8985$). These results prove that the U-Net model works with different object shapes (thin and striped, small and bulky) and styles (occurred in scattered, recurring or continuous style). Furthermore, detailed comparisons and discussion of existing methods in relation to the developed method are summarized in Table 4, in terms of data source, image accessibility, longitudinal coverage, model training, implementing efficiency, detection, quantification, and visualization efficiency.

Table 4. Achievement in this work

Performance	Limitations in existing methods	Fulfilled in this work
Data source	Scattered images (2D), photogrammetric orthophotos (2D), range images (3D), photogrammetric mesh model (3D)	Feature images (in orthographic or arbitrary view) of photogrammetric point clouds (3D) can be generated by exporting images or taking screenshots, which include RGB, DEM, Normal, and their combinations of RGB + DEM, RGB + Normal, DEM + Normal, and RGB + DEM + Normal
Image accessibility	When images were captured in intervals, some target objects were skipped; when images were captured in overlapping, some target objects were double counted	3D navigation of point clouds in Autodesk ReCap Pro is very convenient by panning, zooming, and orbiting; thus, no scanned object would be hidden for rating
Longitudinal coverage	Targets with large longitudinal dimension (e.g., roadway pavement) can be smoothly panned and continuously rated without gaps (e.g., cracking underrating) and overlaps (e.g., cracking overrating)	
Model training	Benchmark data set only available in color images and labels; thus, model training data set preparation is required for each specific task; existing deep learning models were separately using color (2D imagery) and depth (3D imagery) features in cracking detection	With image perspective transformation–based data augmentation (Jiang 2020b), only a small number of feature images and labels are required for the U-Net model training; the pretrained model and the existing data sets can be used to transfer learning and applied for detection of similar objects as well
Implementing efficiency	When a single image is unable to cover a target object or area, obtaining multiple images requires more in-person/in-field work; the captured images only reflect the scanned areas, which may need additional in-person/in-field work	Well-trained U-Net models can run automatically in the background for processing screenshots while receiving them (Jiang 2020c); drone photogrammetry, 3D laser scanning, and simultaneous localization and mapping (SLAM) technologies (Intel 2020; McLaughlin et al. 2020; Shang and

	to access other areas of interest for investigation	Shen 2018) can obtain point clouds (even in real time) without manual intervention; and the point clouds represent the current conditions of the scanned objects, which can be used without any additional field investigation
Detection	Existing FCNs have no significant improvement for crack detection using 2D images	Comparative experimental results show that adding features of DEM and Normal can enhance the pixelwise segmentation performance in the U-Net model compared to only the RGB feature; the experimental results also show the developed method had a high pixel accuracy and IoU, which can conduct instance segmentation (Jiang 2020c) using *Contours* functions in OpenCV (2020)
Quantification	It is hard to continuously rate the pavement cracking without gap (underrating) and overlap (overrating) using scattered images; arbitrary images lack scale reference and are not measurable in physical units	After point cloud alignment via ground control point or precise GPS, point clouds are measurable reality models (in physical units); the dimension and depth (elevation) of detected objects and defects are accessible; thus, the severity of defects can be rated in terms of depth and area; for a large-sized target, the quantification can be conducted on a large-sized feature image or a stitched feature image (in the case of extremely large/long targets)
Visualization	Defects can be marked on the original images	Defects can be marked pixelwise, annotated with contours, and displayed with sequence numbers (Fig. 17); moreover, the object and defect contours can be used to create an as-built model using the automated computer-aided design (CAD) drawing tool in Jiang (2020a)
Overall	N/A	The developed method works on detection of objects and defects with flat background surfaces, which are not limited to cracks, windows, road paths, and slopes presented in this paper; experimental results show that the U-Net model works with different shapes: thin, striped, small, and bulky, scattered, recurring or continuous style. Moreover, when elevations changed quickly in target areas, using the RGB + Normal feature is a reasonable choice; otherwise, the RGB + DEM feature is preferred

The developed method can be applied to roadway assessment (paved and nonpaved) on suburban and rural areas, high-rise building facade inspection (e.g., crack detection), solar panel inspection, on-site improperly assembled scaffolding inspection, on-site excavation slopes inspection, and other AEC applications. The operations of photography and photogrammetry would impact the quality of photogrammetric point clouds, which would limit the performance of the developed method. In practice, overlapping images for photogrammetry can be captured either manually by handheld/vehicle-mounted cameras, or automatically by drones/unmanned ground vehicles (UGVs) (robotics). Image acquisition operations should follow the rules of positioning the camera lens toward the target objects (areas) to capture high ratio overlapping images and avoiding single-axis rotation at each camera station. The rear guideline is very important; otherwise, the photogrammetry would yield a poor performance in point cloud generation, even though a number of images were captured at each camera station, like fragmentary point cloud in Fig. 17(c). Thus, for inspecting vertical surfaces (e.g., building facades), the drone needs to fly in a spiral path (Shang and Shen 2019) as shown in Fig. 17(d); for inspecting top surfaces (e.g., building roofs, solar panels), the drone needs to fly in a co-optimal coverage path (Shang et al. 2020); and for handheld recording ground surfaces (e.g., pavement) and side surfaces (e.g., interior walls), a camera or smartphone should move parallel to and keep a nearly constant distance from target surfaces as much as possible.

Data Availability Statement

The model training and testing data sets are available from the corresponding author upon request. The Python codes are also available from the corresponding author upon request.

Acknowledgments

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

REFERENCES

Ali, L., N. K. Valappil, D. N. A. Kareem, M. J. John, and H. Al Jassmi. 2019. "Pavement crack detection and localization using convolutional neural networks (CNNs)." In Proc., 2019 Int. Conf. on Digitization (ICD), 217–221. New York: IEEE.

Alipour, M., D. K. Harris, and G. R. Miller. 2019. "Robust pixel-level crack detection using deep fully convolutional neural networks." *J. Comput. Civ. Eng.* 33 (6): 04019040. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000854.

Augustaukas, R., and A. Lipnickas. 2019. "Pixel-wise road pavement defects detection using U-net deep neural network." In Proc., 2019 10th IEEE Int. Conf. on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 468–471. New York: IEEE.

Autodesk. 2020. "3D view." Accessed July 4, 2020. https://help.autodesk.com/view/RECAP/ENU/?guid=Reality_Capture_View_and_Navigate_Point_Cloud_Color_Settings_3D_View_html.

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

Chen, L. C., Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. 2018. “Encoder-decoder with atrous separable convolution for semantic image segmentation.” In Proc., Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 833–851. Cham, Switzerland: Springer.

Chollet, F. 2020. “Concatenate layer.” Accessed June 13, 2020. https://keras.io/api/layers/merging_layers/concatenate/.

Dadrasjavan, F., N. Zarrinpanjeh, A. Ameri, G. Engineering, and Q. Branch. 2019. “Automatic crack detection of road pavement based on aerial UAV imagery.” Preprints 2019: 2019070009. https://doi.org/10.20944/preprints201907.0009.v1.

Dorafshan, S., R. J. Thomas, and M. Maguire. 2019. “Benchmarking image processing algorithms for unmanned aerial system-assisted crack detection in concrete structures.” *Infrastructures* 4 (2): 19. https://doi.org/10.3390/infrastructures4020019.

Dung, C. V., and L. D. Anh. 2019. “Autonomous concrete crack detection using deep fully convolutional neural network.” *Autom. Constr.* 99 (Mar): 52–58. https://doi.org/10.1016/j.autcon.2018.11.028.

Edmondson, V., J. Woodward, M. Lim, M. Kane, J. Martin, and I. Shyha. 2019. “Improved non-contact 3D field and processing techniques to achieve macrotexture characterisation of pavements.” *Constr. Build. Mater.* 227 (Dec): 116693. https://doi.org/10.1016/j.conbuildmat.2019.116693.

Fan, R., M. J. Bocus, Y. Zhu, J. Jiao, L. Wang, F. Ma, S. Cheng, and M. Liu. 2019. “Road crack detection using deep convolutional neural network and adaptive thresholding.” In Proc., 2019 IEEE Intelligent Vehicles Symp. (IV), 474–479. New York: IEEE.

Hsieh, Y.-A., and Y. J. Tsai. 2020. “Machine learning for crack detection: Review and model performance comparison.” *J. Comput. Civ. Eng.* 34 (5): 04020038. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000918.

Intel. 2020. “Intel RealSense LiDAR camera L515.” Accessed August 24, 2020. https://www.intelrealsense.com/lidar-camera-l515/.

Ji, A., X. Xue, Y. Wang, X. Luo, and W. Xue. 2020. “An integrated approach to automatic pixel-level crack detection and quantification of asphalt pavement.” *Autom. Constr.* 114 (Jun): 103176. https://doi.org/10.1016/j.autcon.2020.103176.

Jiang, Y. 2020a. “As-built CAD drawing tool.” Accessed November 9, 2020. https://www.yuhanjiang.com/research/DT/CAD.

Jiang, Y. 2020b. “Data augmentation.” Accessed November 9, 2020. https://www.yuhanjiang.com/research/FM/DA.

Jiang, Y. 2020c. “Object detection via point cloud and U-net.” Accessed November 9, 2020. https://www.yuhanjiang.com/research/FM/PC.

Jiang, Y., and Y. Bai. 2020. “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., 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.

Jiang, Y., S. Han, and Y. Bai. 2021. “Development of a pavement evaluation tool using aerial imagery and deep learning.” *J. Transp. Eng. Part B Pavements* 147 (3): 04021027. https://doi.org/10.1061/JPEODX.0000282.

Kalfarisi, R., Z. Y. Wu, and K. Soh. 2020. “Crack detection and segmentation using deep learning with 3D reality mesh model for quantitative assessment and integrated visualization.” *J. Comput. Civ. Eng.* 34 (3): 04020010. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000890.

Kearney, S. P., N. C. Coops, S. Sethi, and G. B. Stenhouse. 2020. “Maintaining accurate, current, rural road network data: An extraction and updating routine using RapidEye, participatory GIS and deep learning.” *Int. J. Appl. Earth Obs. Geoinf.* 87 (May): 102031. https://doi.org/10.1016/j.jag.2019.102031.

Li, Z., C. Cheng, M. P. Kwan, X. Tong, and S. Tian. 2019. “Identifying asphalt pavement distress using UAV LiDAR point cloud data and random forest classification.” *ISPRS Int. J. Geo-Inf.* 8 (1): 39. https://doi.org/10.3390/ijgi8010039.

Liu, Y., J. K. W. Yeoh, and D. K. H. Chua. 2020. “Deep learning–based enhancement of motion blurred UAV concrete crack images.” *J. Comput. Civ. Eng.* 34 (5): 04020028. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000907.

Liu, Z., Y. Cao, Y. Wang, and W. Wang. 2019. “Computer vision-based concrete crack detection using U-net fully convolutional networks.” *Autom. Constr.* 104 (Aug): 129–139. https://doi.org/10.1016/j.autcon.2019.04.005.

Majidifard, H., Y. Adu-Gyamfi, and W. G. Buttlar. 2020. “Deep machine learning approach to develop a new asphalt pavement condition index.” *Constr. Build. Mater.* 247 (Jun): 118513. https://doi.org/10.1016/j.conbuildmat.2020.118513.

Maniat, M. 2019. “Deep learning-based visual crack detection using Google Street View images.” Ph.D. thesis, Dept. of Civil Engineering, Univ. of Memphis.

McLaughlin, E., N. Charron, and S. Narasimhan. 2020. “Automated defect quantification in concrete bridges using robotics and deep learning.” *J. Comput. Civ. Eng.* 34 (5): 04020029. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000915.

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

OpenCV. 2020. “Contours in OpenCV.” Accessed November 9, 2020. https://docs.opencv.org/3.4/d3/d05/tutorial_py_table_of_contents_contours.html.

Pix4D. 2018a. “Highway.” Accessed July 31, 2020. https://cloud.pix4d.com/dataset/272051/map?shareToken=6fb4d298cade4a8c81b206f68fd757c7.

Pix4D. 2018b. “Michigan central station.” Accessed July 31, 2020. https://cloud.pix4d.com/dataset/258513/files/inputs?shareToken=060367e6115f4185902cd33556a70e38.

Pix4D. 2018c. “Quarry 2.0.” Accessed July 31, 2020. https://cloud.pix4d.com/dataset/256164/map?shareToken=97a07d231fbc47b1b105d6cc7bcab0a4.

Propeller Aero. 2018. “What is ground sample distance (GSD) and how does it affect your drone data?” Accessed June 13, 2020. https://www.propelleraero.com/blog/ground-sample-distance-gsd-calculate-drone-data/.

Protopapadakis, E., A. Voulodimos, A. Doulamis, N. Doulamis, and T. Stathaki. 2019. “Automatic crack detection for tunnel inspection using deep learning and heuristic image post-processing.” *Appl. Intell.* 49 (7): 2793–2806. https://doi.org/10.1007/s10489-018-01396-y.

Roberts, R., L. Inzerillo, and G. Di Mino. 2020. “Exploiting low-cost 3D imagery for the purposes of detecting and analyzing pavement distresses.” *Infrastructures* 5 (1): 6. https://doi.org/10.3390/infrastructures5010006.

Ronneberger, O., P. Fischer, and T. Brox. 2015. “U-net: Convolutional networks for biomedical image segmentation.” In Proc., Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 234–241. Cham, Switzerland: Springer.

Shang, Z., J. Bradley, and Z. Shen. 2020. “A co-optimal coverage path planning method for aerial scanning of complex structures.” *Expert Syst. Appl.* 158 (Nov): 113535. https://doi.org/10.1016/j.eswa.2020.113535.

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.

Shang, Z., and Z. Shen. 2019. “Indoor testing and simulation platform for close-distance visual inspection of complex structures using micro quadrotor UAV.” Preprint, submitted April 10, 2019. https://arxiv.org/abs/1904.05271.

Shelhamer, E., J. Long, and T. Darrell. 2017. “Fully convolutional networks for semantic segmentation.” *IEEE Trans. Pattern Anal. Mach. Intell.* 39 (4): 640–651.

Shi, Y., L. Cui, Z. Qi, F. Meng, and Z. Chen. 2016. “Automatic road crack detection using random structured forests.” *IEEE Trans. Intell. Transp. Syst.* 17 (12): 3434–3445. https://doi.org/10.1109/TITS.2016.2552248.

Song, W., G. Jia, H. Zhu, D. Jia, and L. Gao. 2020. “Automated pavement crack damage detection using deep multiscale convolutional features.” *J. Adv. Transp.* 2020 (Jan): 1–11. https://doi.org/10.1155/2020/6412562.

Stacks, D. L. 2019. “Pavement manual: Visual pavement condition surveys.” Accessed May 1, 2020. http://onlinemanuals.txdot.gov/txdotmanuals/pdm/visual_p_cond_surveys.htm.

Yang, Q., W. Shi, J. Chen, and W. Lin. 2020. “Deep convolution neural network-based transfer learning method for civil infrastructure crack detection.” *Autom. Constr.* 116 (Aug): 103199. https://doi.org/10.1016/j.autcon.2020.103199.

Yu, F., and V. Koltun. 2015. “Multi-scale context aggregation by dilated convolutions.” Preprint, submitted November 23, 2015. https://arxiv.org/abs/1511.07122.

Zhang, A., K. C. P. Wang, Y. Fei, Y. Liu, C. Chen, G. Yang, J. Q. Li, E. Yang, and S. Qiu. 2019. “Automated pixel-level pavement crack detection on 3D asphalt surfaces with a recurrent neural network.” *Comput.-Aided Civ. Infrastruct. Eng.* 34 (3): 213–229. https://doi.org/10.1111/mice.12409.

Zhang, K., Y. Zhang, and H.-D. Cheng. 2020. “CrackGAN: Pavement crack detection using partially accurate ground truths based on generative adversarial learning.” *IEEE Trans. Intell. Transp. Syst.* 22 (2): 1306–1319. https://doi.org/10.1109/TITS.2020.2990703.

Zhao, H., J. Shi, X. Qi, X. Wang, and J. Jia. 2017. “Pyramid scene parsing network.” In Proc., 2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 6230–6239. New York: IEEE. https://ieeexplore.ieee.org/document/8100143.

Zhou, S., and W. Song. 2020a. “Deep learning-based roadway crack classification using laser-scanned range images: A comparative study on hyperparameter selection.” *Autom. Constr.* 114 (Jun): 103171. https://doi.org/10.1016/j.autcon.2020.103171.

Zhou, S., and W. Song. 2020b. “Robust image-based surface crack detection using range data.” *J. Comput. Civ. Eng.* 34 (2): 04019054. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000873.

Zou, Q., Z. Zhang, Q. Li, X. Qi, Q. Wang, and S. Wang. 2019. “DeepCrack: Learning hierarchical convolutional features for crack detection.” *IEEE Trans. Image Process.* 28 (3): 1498–1512. https://doi.org/10.1109/TIP.2018.2878966.