Smart Structures and Systems, *Vol. 24, No. 6 (2019) 745-757*
DOI: https://doi.org/10.12989/sss.2019.24.6.745

Autonomous pothole detection using deep region-based convolutional neural network with cloud computing

Longxi Luo[1,2a], Maria Q. Feng[3b], Jianping Wu*[1,2] and Ryan Y. Leung[3c]

[1]*Department of Civil Engineering, Tsinghua University, Beijing, China*
[2]*Jiangsu Province Collaborative Innovation Center of Modern Urban, Traffic Technologies, Southeast University Road #2, Nanjing, China*
[3]*Department of Civil Engineering and Engineering Mechanics, Columbia University, New York, USA*

(Received May 26, 2019, Revised July 30, 2019, Accepted August 6, 2019)

Abstract. Road surface deteriorations such as potholes have caused motorists heavy monetary damages every year. However, effective road condition monitoring has been a continuing challenge to road owners. Depth cameras have a small field of view and can be easily affected by vehicle bouncing. Traditional image processing methods based on algorithms such as segmentation cannot adapt to varying environmental and camera scenarios. In recent years, novel object detection methods based on deep learning algorithms have produced good results in detecting typical objects, such as faces, vehicles, structures and more, even in scenarios with changing object distances, camera angles, lighting conditions, etc. Therefore, in this study, a Deep Learning Pothole Detector (DLPD) based on the deep region-based convolutional neural network is proposed for autonomous detection of potholes from images. About 900 images with potholes and road surface conditions are collected and divided into training and testing data. Parameters of the network in the DLPD are calibrated based on sensitivity tests. Then, the calibrated DLPD is trained by the training data and applied to the 215 testing images to evaluate its performance. It is demonstrated that potholes can be automatically detected with high average precision over 93%. Potholes can be differentiated from manholes by training and applying a manhole-pothole classifier which is constructed using the convolutional neural network layers in DLPD. Repeated detection of the same potholes can be prevented through feature matching of the newly detected pothole with previously detected potholes within a small region.

Keywords: pothole detection; deep learning; faster-RCNN; road condition monitoring

1. Introduction

The 2017American Infrastructure Report Card by the American Society of Civil Engineers gave US roads a grade of "D", because they are frequently in poor condition, and are becoming increasingly dangerous (ASCE 2017). Potholes and deteriorating roads contributed to an estimated $120 billion cost to U.S. motorists in extra vehicle repairs and operating costs in 2015. Potholes are one of the major contributions to the problem. Fixing the potholes on time can increase the road condition, and save motorists money. The current practice is road patrol by an inspection team, followed by road maintenance. This practice, however, is time inefficient and tedious, so potholes might not be timely fixed. If the potholes can be autonomously detected by passing vehicles, and the locations of the potholes can be stored on a cloud server and overlaid on road maps, the maintenance vehicles can then be dispatched with the best working routes optimized based on the densities and severities to fix the potholes effectively. To achieve this, techniques capable of automatic pothole detection are required.

A number of techniques have been developed for detecting pothole using different sensors. The accelerometer is one of the sensors applied for pothole detection or pavement distress detection (Eriksson *et al.* 2008, Mednis *et al.* 2011, Jang *et al.* 2016). The acceleration data collected by accelerometers or smartphones installed on a vehicle are analyzed to find the vibrational patterns of vehicles moving over potholes. However, the vehicle needs to drive over the pothole in order to trigger the detection, resulting in missing of some potholes. Also, new vehicles tend to have better and better damping systems, which can significantly reduce vehicle vibration due to potholes. Sudden vehicle maneuvers and other objects on the roads, such as borderline expansion joints, train tracks, and manholes, can also trigger false alarms.

Image processing techniques in combination with vision sensors have also been applied for pothole detection (Koch and Brilakis 2011, Jog *et al.* 2012). But some of the methods depend on traditional techniques, such as image segmentation and shape fitting, which can be difficult in adapting to scenarios with changing lighting conditions, camera angles, object distances, etc. Depth sensors such as the Kinect camera have also been proposed for pothole detection (Jahanshahi *et al.* 2012; Chen *et al.* 2016). Kinect can detect the distance between the road surface and the receptive sensor. Due to the current capacity limit of the

*Corresponding author, Ph.D., Professor
E-mail: jianpingwu@tsinghua.edu.cn
[a] Ph.D., Postdoctoral research fellow
[b] Ph.D., Professor
[c] Ph.D. student

ISSN: 1738-1584 (Print), 1738-1991 (Online)

Kinect, the detection system has limited measurement distance and can only detect a small area. Another difficulty that needs to be tackled before field application of the Kinect is the bounces of the carrying vehicle, which can result in large errors on bumpy roads. The installation of the data-acquisition which is composed of a Kinect, computer, and power supply can also cumbersome. In general, the above-mentioned methods have difficulties in field applications due to their sensitivity to noises and condition changes.

Recent decades have witnessed fast developments in the field of computer vision. Computer vision techniques have been widely used for dynamic displacement monitoring (Ye *et al.* 2013, Ye *et al.* 2016b, Cha *et al.* 2017a, Yoon *et al.* 2018, Luo *et al.* 2018.), surface displacement/strain measurement (Luo *et al.* 2015), vision-based structural analysis (Chen *et al.* 2015, Park *et al.* 2018), cable tensile force evaluation (Kim *et al.* 2013, Ye *et al.* 2016a), rocking motion and landslide monitoring (Debella-Gilo and Kääb 2011, Greenbaum *et al.* 2015), etc.

Among the new computer vision techniques, object detection techniques have been implemented in the field of civil engineering for applications such as surface crack/defection detection (Cord and Chambon 2012, Cha *et al.* 2017b, Jahanshahi *et al.* 2017), and intelligent transportation (Fernandez-Llorca *et al.* 2017). With the fast advances in computer hardware and computer vision algorithms, deep neural networks have been performing better and better on the benchmark object classification/detection competitions. In 2012, the AlexNet (Krizhevsky *et al.* 2012) won the ImageNet competition with errors much less than other methods that integrate feature extraction algorithms and traditional machine learning algorithms such as the support vector machine (SVM). Since then, deep learning methods have been dominating the object classification/detection competitions. Technology corporations have also started wide applications of deep learning for services such as automatic tagging, photo search, product recommendation, etc. The region-based convolutional neural network (RCNN) (Girshick *et al.* 2014, Girshick 2015, Ren *et al.* 2015) enabled object localization in addition to the object classification in a scene image.

One challenge for the object detection methods based on deep neural networks is that the training process can be computationally expensive with the increase of the architecture complexity. Two solutions in this study include the application of the computational effective method faster region convolutional neural network (faster-RCNN) and application of elastic cloud computing technology. In the faster-RCNN, the convolutional layers are not only used by object detection network but also shared by the region proposal network for identifying possible object regions, thus reducing redundancies in the training of convolutional layers. The cloud computing services offer on-demand shared servers with high computational capabilities, thus obviating the need to purchase expensive high-performance graphic cards that upgrade fast nowadays.

Therefore, in this paper, a new pothole detection technique based on deep learning DLPD is proposed for automatic pothole detections from images and videos. And for the proposed DLPD, a deep neural network based on the structure of faster-RCNN method is constructed. Images of potholes and road surfaces are collected and stored in the cloud server. The collected data are divided into training images and testing images. Parameters of the network in DLPD are calibrated through sensitivity tests. Then, the calibrated DLPD is trained using the training data. Performance of the trained DLPD is evaluated on the testing data. Potholes are differentiated from manholes by applying a manhole-pothole classifier trained using the convolution layers of the DLPD. Multi-detections of the same pothole can be avoided by matching the newly detected pothole with previously detected potholes through feature extraction and tracking methods, to enable effective pothole detection on videos. This study, in general, provides a framework for developing a smart detector for road surface deteriorations such as potholes.

This paper is arranged as follows: Section 2 explains the relevant methodologies and the network for developing the DLPD. Section 3 introduces the collected data and the implementation process. The experimental results are also presented in Section 3. Section 4 concludes this study.

2. Methodology

2.1 Overview of the Deep Learning Pothole Detector

The proposed DLPD is designed based on the object detection network. At first, a detector network, composed of a region proposal network (RPN), and a convolutional neural network (CNN) and the layers unique to the fast-RCNN network, is constructed by following the faster-RCNN method. Then the detector network was calibrated and validated using the training data. The calibrated network was then trained using the training data and its performance was evaluated by applying to the testing data.

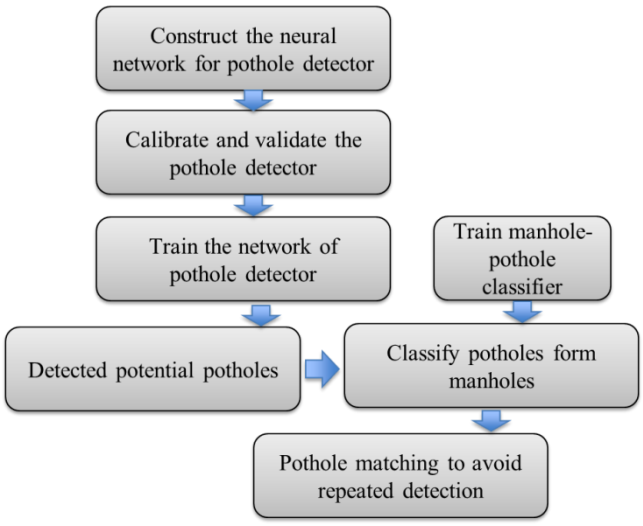

Fig. 1 Overview of the proposed DLPD

To differentiate the potholes from the manholes, the detected potential potholes were then classified using a manhole-pothole classifier using the convolutional neural network. Finally, the newly detected pothole was matched with previously detected potholes within a small region using feature extraction and tracking methods to ensure potholes were not repeatedly detected. The overview of the DLPD is illustrated in Fig. 1. Since usually cameras and smartphones have GPS localization nowadays, the detected potholes were recorded with GPS locations and detection times and sent to the cloud server to be stored. Maintenance companies or road owners can then make optimal maintenance decision based on the pothole distributions.

2.2 Convolutional neural network

Different from methods based on traditional feature extraction algorithms, methods based on the deep convolutional neural network do not require specific schemes for extracting features from images. A CNN, which is composed of sequential convolution layers, rectification layers, pooling layers, etc., is used to automatically generate features from the input images. The parameters of each layer are automatically trained for detecting a certain type of object such as potholes. Since the intermediate and final outputs of the convolution layers in the network cannot be controlled or explained physically, methods based on the CNN are usually called black-box methods. With the fast development in the computation capabilities of computing units, the convolutional neural network is getting deeper with more sequential layers and have been producing great results in object classification competitions since 2012. The fundamental components of CNN are briefly explained.

Convolutional layer

The most important layers in CNN are usually the Convolution layers (Conv). In a Conv, the size and number of the sliding filters are first assigned. The sliding filter (f_1) of size ($m_1 * n_1 * ch$) is then convolved with the input image (I_1) of size ($a_1 * b_1 * ch$) or the output from its previous layer. Note ch is the channel size of the image, so $ch = 1$ for grayscale and $ch = 3$ for RGB color images. The sliding filter is convolved with the input image to form a feature map (C_{conv}). Based on the definition of convolution, the value of a pixel on the feature map can be calculated as

$$C_{conv}(i, j) = \sum_f f_1 \cdot I_{1ij} \tag{1}$$

where (i, j) is the coordinate of the pixel on the feature map and coordinate of the sliding filter on the input image, and I_{1ij} is the part of the input image under the filter. The size of c is decided by the stride or step-size (k_1) of the sliding filter, so the size of c_{conv} is $\left((a_1 - m_1)/k_1 + 1\right) * \left((b_1 - n_1)/k_1 + 1\right)$. Multiple filters can be applied in this layer to produce multiple feature maps.

Rectified linear unit layer

Convolutional layers are usually followed by a nonlinear activation function in the rectified linear unit (ReLU) layer. The ReLU layer converts the input (usually the feature map from the Conv layer) into binary signals. The ReLU layer does not change the size of its input. Common ReLU layer applies one of the following operations to each element of the input (I_2): hyperbolic tangent function $f(x) = \tanh(x)$, logistic function (aka. sigmoid) $f(x) = \left(1 + e^{-x}\right)^{-1}$ or simple binary step function $f(x) = \begin{cases} 1, x > 0 \\ 0, x \le 0 \end{cases}$.

Max-pooling layer

The max-pooling layer (maxPool) is always placed behind the ReLU layer for down-sampling the data to reduce the number of parameters to be trained in the next layer. At first, values of the size and stride of a pooling window are defined. The pooling window f_3 of size ($m_3 * n_3$) then moved along two axes of the input by the predefined step-size. The maximum value of the pixels overlaid by the pooling window on the input $I_{3_{ij}}$ is used to form a maxPool map $C_{maxPool}$:

$$C_{maxPool}(i, j) = \max\left(I_{3ij}\right) \tag{2}$$

where I_{3ij} is the part of the input image under the pooling window and (i, j) represents the position of the pooling window on the input. If the stride is assigned with the value k_2 , the size of the output max-pool map is $\left(\frac{(a_3 - m_3)}{k_3} + 1\right) * \left(\frac{(b_3 - n_3)}{k_3} + 1\right)$.

Average-pooling layer

The average-pooling (averagePool) layer is very similar to the maxPool layer except that the average value instead of the maximum value within the region I_{4ij} of the input I_4 overlapped by the pooling window w_4 is returned to form an averagePool map $c_{averagePool}$.

Local response normalization layer

The local response normalization (LocalRespNorm) layer was introduced in AlexNet based on the concept of lateral inhibition in neurobiology. In the LocolRespNorm

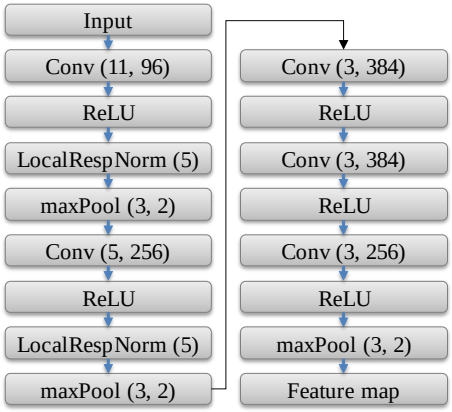

Fig. 2 Example CNN based on AlexNet

layer, the response or input (I_5) is normalized locally within a sliding window (f_5) that moves pixel by pixel. Within the window, each response value at position (i, j) is divided by $\left(r+\alpha\sum_{ij} I_{5ij}^2\right)^{\beta}$, where r, α, β are hyper-parameters arbitrarily defined, and the sum is taken over the window centered at that value. The final normalized output (C_5) is thus obtained

$$C_5(i,j)=\frac{I_{5ij}}{\left(r+\alpha\sum_{ij} I_{5ij}^2\right)^{\beta}} \quad (3)$$

Fully connected layer

The fully connected layer transforms the high-level features, which are generated by abovementioned layers, into a vector of length K , which is equal to the number of object classes. This is achieved by combining the features nonlinearly with weight and a bias term. Essentially, the weight reflects the correlation between a feature and a corresponding class. If the size of the high-level features is N , and the number of object classes is K , the size of the weight matrix of the fully connected layer is $N*K$ such that $R^{1*N}\times R^{N*K}=R^{K*1}$.

Basically, a trained FC layer reflects how well a certain high-level feature correlates to a particular class. Ideally, the product of the weight matrix and the input features I_6 can correctly represent the probabilities of an object belongs to different classes.

Other layers

There are a number of other types of layers. For example, in the Softmax layer, the softmax function is applied to the input. In the Classification layer, the cross-entropy is applied to the input. The deep CNN is formed by a smart combination of a variety of layers. An example of the CNN which follows the basic structure of the AlexNet (Krizhevsky *et al*. 2012) is illustrated in Figure 2. The fully connected layers are not included in the example illustration since they are used for classification and will be included later. In the plot, the values for the Conv layers represent the filter size and number of filters respectively. The value for the LocalRespNorm defines the filter size, and the values for the maxPool represent the filter size and the step-size separately.

2.3 Fast Region CNN

The CNN can only be trained for classifying the whole images, which are usually close-up images with single objects. However, it can be more useful but more difficult to detect (locate and classify) objects on the image with a large scene of background, such as detecting potholes on the images of road surfaces. The Region CNN (RCNN) algorithm was developed to enable object localization and detection from a large scene. Furthermore, the RCNN was developed into the Fast Region CNN (fast-RCNN) to improve computational efficiency and detection accuracy. In the fast-RCNN, images are first fed into CNN to extract feature maps. Then region proposals, which are regions likely to include objects, are identified and located on the feature maps. These region proposals are transformed into vectors from pooling to be sent to the fully-connected layers which are followed by two parallel layers softmax layer and regression layer. The softmax layer returns the probability of the region proposal belongs to each object class, in this case, pothole and background ($p_{pothole}, p_{background}$). The regressor layer returns the center coordinates and the sizes of the region proposal.

In the RCNN, classification of the region proposals is performed by the SVM algorithm (Girshick *et al*. 2014). In the fast-RCNN, to reduce the computational cost, a classification layer and a box regression layer are included as part of the neural network (Girshick 2015). Also, in the fast-RCNN, feature maps of the region proposals with different sizes are normalized to have the same size through region-of-interest pooling (ROI pooling). The graphic illustration of the fast-RCNN method is presented in Fig. 3.

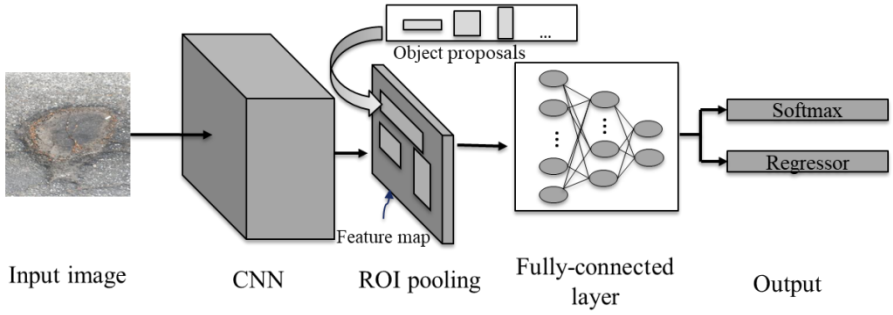

Fig. 3 Structure of the Fast-RCNN

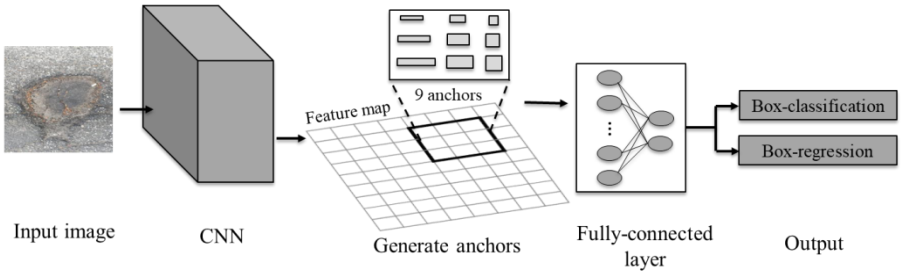

Fig. 4 Structure of the RPN

Both the RCNN and fast-RCNN apply the Selective Search for region proposals, which became the bottleneck for both methods.

2.4 Region proposal network

Before the development of the RPN, independent scanning methods such as Selective Search is implemented by the fast-RCNN for generating bounding boxes around object proposals, which are likely to be objects. The Selective Search is an exhaustive method that proposes a large number of bounding boxes with different dimensions, and selects the top 2000 proposals. Sub-segmentation groups of proposals with small dimensions are first generated and grouped into proposals with larger dimensions from bottom-up based on the similarities measured in terms of texture, color, size, and shape of the sub-segmentation groups. All the groups with different dimensions are then ranked based on a metric called average best overlap (Uijlings *et al*. 2013). The finally selected object proposals are used as potential objects in the RCNN and fast-RCNN. The application of the Selective Search, however, is computationally expensive and is a separate module independent from the detection network, thus it became the bottleneck for both the RCNN and fast-RCNN.

To reduce the computation cost and to take advantage of the convolutional layers, the low computational cost RPN was proposed in the faster-RCNN, based on the observation that the final feature maps created by the convolutional layers fast-RCNN can be used to generate object proposals. The objective of the RPN is to propose potential regions of objects with low computational cost by taking advantage of the convolutional layers. Each object proposal is assigned an objectiveness score that measures the possibility of the object proposal belongs to an object class rather than the background. Feature maps are extracted from an image by the convolutional layers. The series of convolutional layers are followed immediately by an intermediate proposal convolution layer, which is the core element of the RPN and is applied to the feature map to general object proposals. The object proposals are then fed into the fully connected layer, and box-regression and box-classification layers for refining the bounding boxes and classifying the object proposals. The region proposal network is illustrated in Fig. 4.

In the intermediate proposal convolution layer placed behind the convolution layers in the RPN, a sliding window is applied to stride over the feature map with a step-size of P. In each step, the sliding window proposes K different regions, called anchors with different sizes decided by a variety of scales and aspect ratios. Usually, three scales and three aspect ratios that can define nine anchors are usually sufficient. For each anchor, the box-classification layer assigns the anchor with objectiveness scores that estimate the probability of such anchor belonging to different objects or the background. For a feature map with the size of $M \times H$, the total number of anchors is $M \times H \times \frac{K}{P}$. Because the anchors are designed with multiple scales, object proposals with different scales can be obtained from single scale image and one sliding window.

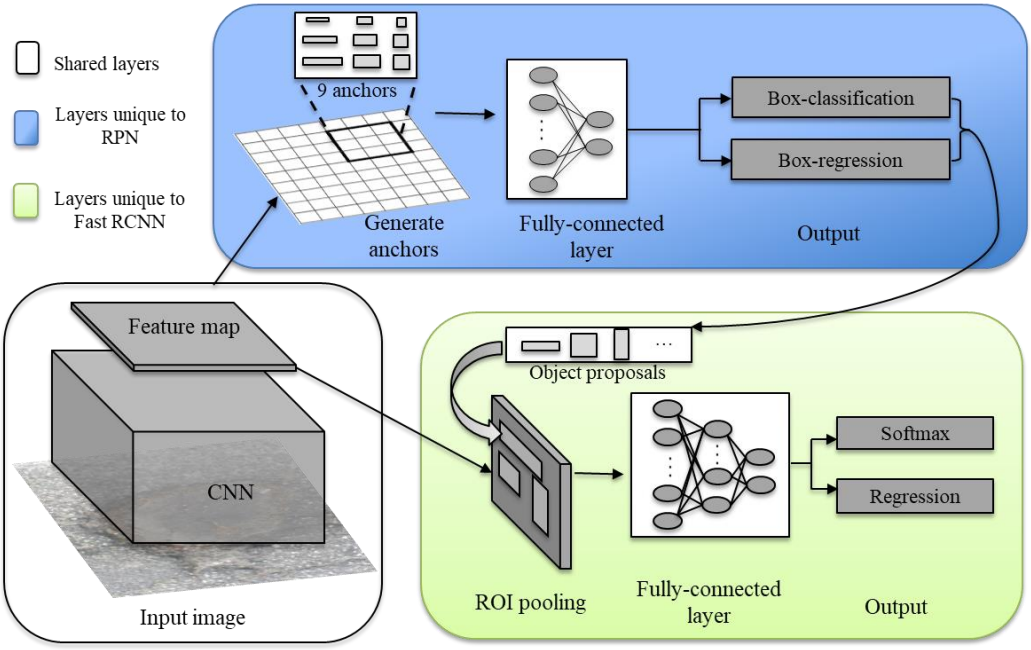

Fig. 5 Structure of the faster-RCNN with CNN shared by RPN and fast-RCNN

2.5 Faster-RCNN by sharing CNN between RPN and fast-RCNN

Since both the RPN and the fast-RCNN need a CNN for extracting feature maps. Therefore, CNN is shared between the RPN and the fast-RCNN for improving computational efficiencies. The sharing process is composed of four steps.

In the first step, object proposals are generated by the RPN. An RPN initiated by random distributions or extracted from previously trained networks can be used to generate object proposals likely to be potholes from positive and negative sample images.

In the second step, a fast-RCNN is trained using object proposals obtained from the first step. The CNN in the fast-RCNN is initialized by random distributions. The fast-RCNN is trained to minimize the errors of the class labels of the object proposals comparing with the ground truth. The first two steps are only conducted for preliminary training of the RPN and fast-RCNN and do not share CNN.

In the third step, the CNN, which are mainly composed of convolutional layers, in the trained fast-RCNN from the second step is used to replace that in the RPN. Thus, the same convolutional layers are shared by both the fast-RCNN and the RPN. Also, layers unique to the RPN are fine-tuned through training.

In the fourth step, the fully-connected, box-classification, and box-regression layers unique to the fast-RCNN are fine-tuned through repeated training.

The structure of the faster-RCNN with shared CNN between RPN and fast-RCNN is presented in Fig. 5.

After training the detector's network by following the four steps, the DLPD capable of detecting potholes with high computational efficiencies can be obtained. If a new image is provided to the DLPD, at first, a feature map is generated by the CNN. Then, object proposals are provided by layers unique to the RPN, and are overlaid on the feature map. Finally, layers unique to the fast-RCNN can decide the objectiveness score to the object proposals and adjust their bounding boxes. The output is an image with detected potholes encircled by bounding boxes. In this study, the structure of the renowned AlexNet (Krizhevsky *et al*. 2012) was applied as CNN.

2.6 Training of the Deep Learning Pothole Detector

The training process of the DLPD's network is then explained. Before training the DLPD, ground-truth rectangular boxes are drawn around the actual potholes. However, locations and boundaries of the object proposals produced by the RPN cannot always exactly match the ground-truth boxes. Therefore, during the training process of the detector, the true label of each object proposal is determined by comparing its bounding box with the ground-truth boxes. If the Intersection over Union (IOU) between the object proposal and any the ground-truth boxes of potholes is higher than 0.5, it is considered a pothole. The IOU is defined as the intersection of two regions (e.g., $P1$ and $P2$) over their union

$$IOU = \frac{P1 \cap P2}{P1 \cup P2} \tag{4}$$

If the highest IOU between the object proposal and the ground-truth boxes falls between 0.1 and 0.5, the region proposal is considered a negative detection. The objective function L for the DLPD is defined by combining the box regression losses and the classification losses (Ren *et al*. 2015)

$$L = L_{reg} + L_{cls} \tag{5}$$

where, L_{reg} is the objective function for the box regression results and L_{cls} is that for the classification results.

In training the DLPD, parameters of the detector's network are first forward initialized, then fine-tuned through backpropagation to minimize the objective function (Bousquet and Bottou 2008). In the forward initialization process, the convolutional layers are initiated by random distributions. In the backpropagation process, the gradient descent method can be applied to find the minimum of the objective function. In gradient descent, if a multi-variable objective function is differentiable in the neighborhood of a parameter (w), the local minimum of the objective function can be found by iteratively updating the parameter w by applying the following function

$$w^t = w^{t-1} - \gamma \nabla_w Q\left(w^{t-1}\right) \tag{6}$$

where Q is the whole dataset of samples, γ is the stepping size arbitrarily assigned, and t is the iteration sequence.

However, the application of gradient descent can be slow. Therefore, the minibatch stochastic gradient descent (MB-SGD) is applied so that only a small portion of random samples are used to update the function in each epoch. In the MB-SGD, samples are segmented into mini-batches. Each mini-batch is composed of a small portion of the object proposals (128 object proposals, for example), and different mini-batches are used for training in each epoch of iterations. About a quarter of the mini-batch are positive samples, while the others are negative samples to avoid bias towards negative samples. The samples are randomly chosen over permutations; therefore, the whole dataset is used over iterations. In the MB-SGD, the parameter w is updated iteratively by following the function below

$$w^t = w^{t-1} - \eta \frac{1}{N} \sum_{i=1}^{N} \nabla_w Q_i\left(w^{t-1}\right) \tag{7}$$

where Q_i is the mini-batch of samples, N is the number of iterations, η is the stepping size, and t is the iteration sequence.

2.7 Quantitative measurement index

The accuracy of the testing results is quantified using the average precision. To calculate the average precision, the recall and precision of the results are first calculated. The recall is defined as the ratio of the true positive classifications (TP) over the sum of TP and false negative classifications (FN). In other words, the recall (r) estimates the ratio of correctly classified positive samples over the total number of true positive samples

$$r = \frac{TP}{TP + FN} \tag{8}$$

Similarly, the precision (p) is defined as the ratio of correctly classified positive samples TP over the total number of TP and samples falsely classified as positive (FP)

$$p = \frac{TP}{TP + FP} \tag{9}$$

Theoretically, the average precision is defined as the area under the recall-precision curve. So, the average precision is the precision averaged across all values of recall between 0 and 1

$$AP = \int_0^1 p\left(x\right) dx \tag{10}$$

The average precision can be approximated by summing at discrete values of the recall

$$AP = \sum_{k=1}^{K} p\left(k\right) \cdot r\left(k\right) \tag{11}$$

More commonly, the average precision is approximated by 11 points interpolated average precision. The recall is divided into 11 cut-off points. And for each cut-off point, the maximum precision for recalls equal or above the cut-off recall point is used

$$AP = \frac{1}{11} \sum_{k=0}^{10} \max_{\tilde{k} > k} p\left(\frac{k}{10}\right) \tag{12}$$

3. Implementation and experiment demonstration

The implementation and experiment demonstration of the DLPD are presented. At first, training and testing datasets collected for calibrating, validating, and evaluating the DLPD are introduced. Then the implementation details of the DLPD are explained. The results of calibrating the detector based on sensitivity tests are presented. Then the DLPD is applied to the testing data to evaluate its performance. Potholes are differentiated from manholes by applying a manhole-pothole classifier. Repeated detection of the same potholes can be avoided by matching the newly detected pothole with previously detected potholes within the small region.

3.1 Training, validation, and testing datasets

Images of road surfaces were collected from multiple sources such as manual photographing using smartphones and image mining on the Internet. A total of about 900 images were collected. Datasets for training and testing were composed of positive (pothole) and negative (background) samples extracted from collected images. Positive samples were extracted from the collected images by cropping the images around the potholes. Other objects

and the plain background produced the negative samples. All extracted datasets were divided into 75% of training data and 25% of testing data. For calibration and validation of the network, the training data was also separated into 80% of preliminary training data and 20% of validation data.

3.2 Implementation of the Deep Learning Pothole Detector

The implementation of the DLPD was in general composed of four sequential steps: dataset labeling, training parameters calibration, DLPD training, and pothole detection on testing images. Datasets were manually labeled and saved as lists. Lists of positive samples contain image filenames and locations of potholes on the images. The datasets were then randomly divided into testing datasets, and training and validation datasets. When applied to the testing data, a pothole detection result was considered true positive when the IOU between the bounding box of the detected pothole and the ground-truth is above a threshold such as 0.3. It was considered a false negative when a ground-truth pothole was missed, and a false positive when the background was detected as a pothole. The training process was iterated numerous times using multiple epochs of mini-batch of training data. All epochs together constituted the whole training datasets. The implementation of the DLPD was performed on an on-demand elastic compute cloud service. The cloud platform provides GPU-based parallel compute capabilities. The hardware system includes an NVIDIA K80 GPU with 24 GB of GDDR5, 4 vCPUs, and 61 GiB of RAM.

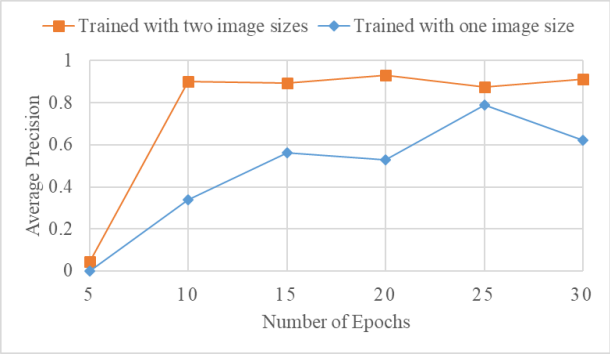

Fig. 6 Average precision versus number of epochs

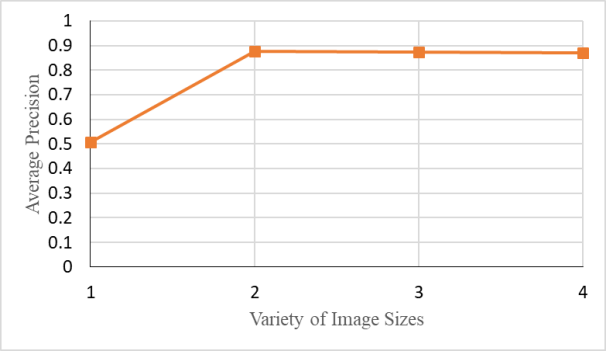

Fig. 7 Average precision versus variety of image sizes

3.3 Parameter calibration based on sensitivity tests on validation data

The DLPD' network needs to be calibrated and validated before it can be trained and used for testing new images. Therefore, the training data was segmented into 80% of preliminary training data and 20% of validation data. The detector's network was first trained using the preliminary training data, then its performance was evaluated when applied to the validation data. Sensitivity tests on major parameters in training the network were performed by evaluating the network performance in terms of average precision under different parameter values. At first, the network's sensitivity to the number of epochs was tested by increasing it from 5 to 30. The average precisions corresponding to the different number of epochs are plotted in Fig. 6. As shown in the plot, with the premise that the network was trained with one image size, the average precision jumped when the number of epochs increased from 5 to 10, then continued to climb until the number of epochs reached 25, and then started to decline slightly when the number of epochs reached 30. When the network was trained with two image sizes, similar patterns were observed: the average precision jumped when the number of epochs was increased from 5 to 10, and reached its peak when the number of epochs was increased to 20. Therefore, in the case of training the network with one image size and two image sizes, the local optimum of the number of epochs was 25 and 20 separately. When training the network, dimensions of the preliminary training images were normalized to have the same shorter dimension. Based on empirical experience, training images with a variety of sizes can improve network performance. Therefore, the training images were resized in different groups. In each group, the training images had the same shorter dimension, which was the multiple of 60. Four groups with shorter dimensions ranging from 60 to 240 were produced. The network was trained using one or more groups to test the sensitivity of the network to a variety of image sizes. As presented in Fig. 6, the network trained with two varieties of image sizes produced better results than that trained with one variety of image size. Also, when the network was trained with two image sizes, the average precision reached higher values with lower number of epochs.

A further sensitivity test on the varieties of image sizes was also presented in Fig. 7. It was found that training the network with more than two varieties of image sizes did not further improve the validation accuracy. Generally, training the network with two varieties of image sizes can improve the network's performance by increasing the average precisions and decreasing the optimal number of epochs. Other parameters were also tested to study their impacts on the network performance and their optimal values were chosen, although their impacts were less significant. Based on the sensitivity tests, the number of epochs was set to 20 and the network was trained using two varieties images sizes for the best performance of the DLPD. For training the final DLPD, both the preliminary training data and the validation data were used as training samples.

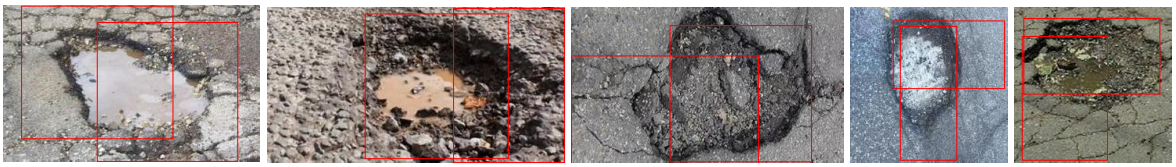

Fig. 8 Examples of one pothole detected as multiple potholes with overlapping bounding boxes

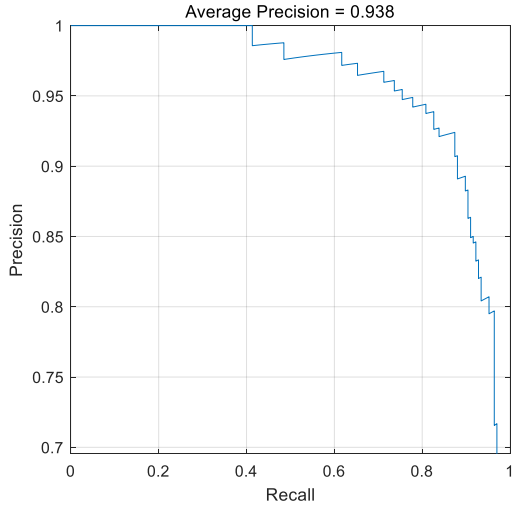

Fig. 9 Precision-recall curve of testing results

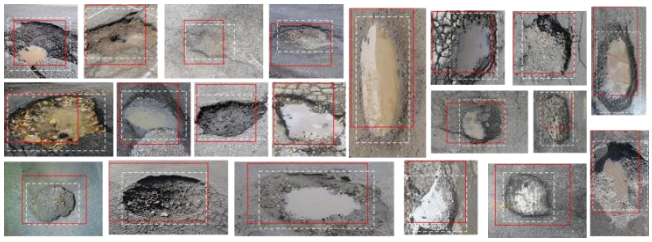

Fig.10 Examples of accurately detected potholes with different characteristics (white dash lines belong to the ground-truth bounding boxes; solid red lines belong to the bounding boxes produced by the DLPD)

It was also noticed, sometimes, one ground-truth pothole might be detected as multiple potholes with overlapping bounding boxes because all these regions encircled the same pothole as shown in Fig. 8. However, potholes are usually not adjacently placed based on field observations. Therefore, this problem can be solved by selecting the bounding box, among the overlapping bounding boxes, that was given the highest objectiveness score, because it was considered, by the detector, most likely to be the region of a pothole.

3.4 Detection results of the Deep Learning Pothole Detector applied on testing images

The performance of the trained DLPD was evaluated unbiasedly using the 215 testing images. The recall and the precision were calculated to produce the precision-recall curve, which is presented in Fig. 9. In general, the DLPD produced good detection results with an average precision of 0.938.

A small portion of the true positive detection results is presented in Fig. 10. The proposed DLPD accurately detected the presence and location of the potholes. The testing images included potholes in different scenarios. Potholes in the testing images had different shapes such as circles, ellipses, triangles, and irregular shapes. The contents inside the potholes were also different including gravels, broken asphalt, sands, soils, muds, and white substances likely to be ground limestones. In some of the potholes, rainwater was accumulated. The contents in the potholes had different colors ranging from bright to dark colors. The differences in the shape, texture, and color of the potholes made it difficult to accurately detect potholes using traditional image processing algorithms such as image segmentation and shape fitting. Traditional feature extraction methods also depend heavily on image intensities

Fig. 11 Examples of true negatives

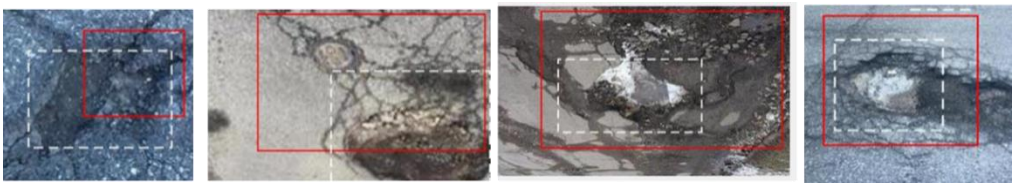

Fig. 12 Imperfect bounding boxes for the detected potholes

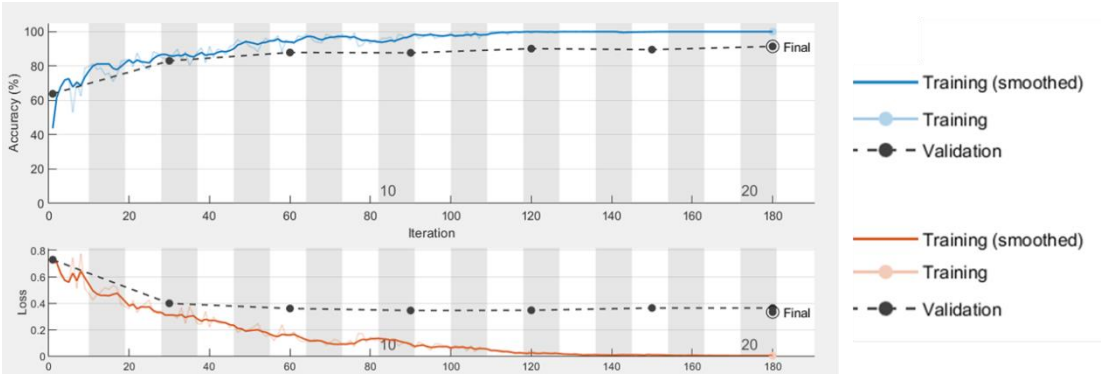

Fig. 13 Improvements of the training and validation accuracy during the classifier training

of the objects, therefore theoretically cannot robustly detect the potholes under varying conditions. The DLPD trained using the deep convolutional neural network was able to accurately detect the potholes since a large number of most significant features though cannot be intuitively understood were automatically extracted by the trained network layers. Some of the true negative detections were presented in Fig. 11. Plain road surfaces and road markings usually had obvious characteristics such as brightness, texture and shape different potholes, therefore can be easily differentiated.

There were some limitations to the DLPD mainly due to imperfect pothole bounding boxes. Such as examples shown in Fig. 12, the potholes were accurately detected, however, the bounding boxes of these potholes were not perfect and did not overlap well with the ground-truth boxes. This was because some of the potholes had ambiguous boundaries and the detector considers a broader or narrower region as the object. The imperfect boundaries boxes, however, resulted in both false negatives and false positives in the accuracy evaluation; this was because an imperfect bounding box was considered a false negative and the corresponding pothole was considered to be missed by the detector (false negative). For the purpose of detecting and locating potholes on the city streets so the maintenance team can be more informed, these imperfect bounding boxes with errors less than one meter were acceptable.

3.5 Differentiate potholes from manholes

Manholes and potholes can be ambiguous to the DLPD sine they have similar shapes, and both of their textures stand out from the background road surface. In detecting the potholes, it was noticed that manholes were sometimes detected as potholes. To differentiate potholes from manholes, a straight-forward solution was to train a two-class manhole-pothole classifier. When a potential object was detected, the region within the bounding box was classified by the classifier to determine whether it was a pothole or a manhole. Methods based on CNN also produced good results in the object classification competitions. Therefore, the deep CNN, AlexNet, used in the pothole network were extracted to train the pothole-manhole classifier. About 860 manhole images and 900 pothole images were divided into training and testing images and were used to train the classifier. Similar to the setting for the DLPD, the number of epochs was set to 20.

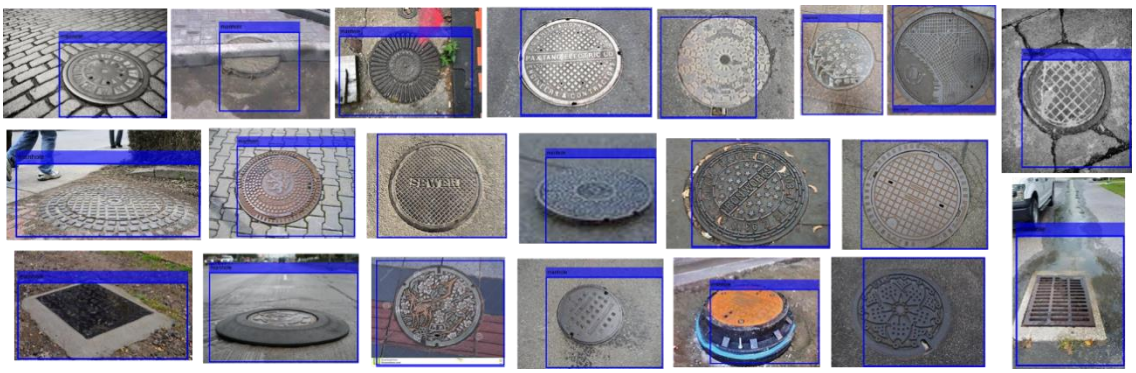

Fig. 14 Potential objects accurately classified as manholes

Fig. 15 Few manholes falsely classified as potholes

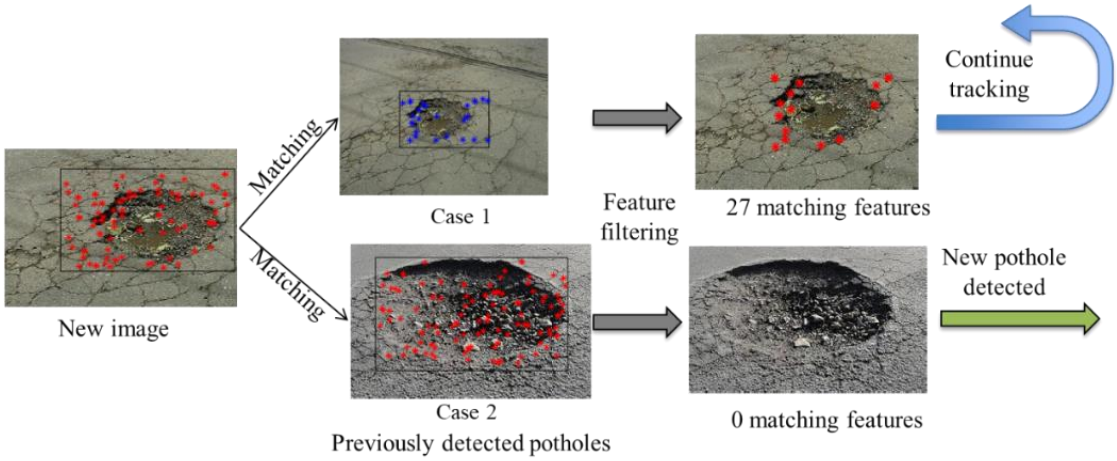

Fig. 16 Pothole matching using the feature tracking method

The training and validation results during the training process were plotted in Fig. 13. The training accuracy grew and the loss declined with the increase of iterations. About 180 iterations were performed for the 20 epochs of mini-batch of data.

The trained classifier was then evaluated on the testing data. Again, bounding boxes around potential potholes were encircled by the DLPD, and regions within the boxes were classified by the manhole-pothole classifier. A good classification result was obtained with the accuracy of 0.9143 for about 300 testing images. As presented in Fig. 14, manholes in the example testing images were accurately classified. A few of the manholes were however falsely classified as potholes. But as shown in Fig. 15, most of the falsely classified manholes in these images looked very similar to potholes due to deteriorations of the manhole covers, deterioration of surrounding road surfaces or the shade under an upturned manhole cover. In general, accurate classifications of the potholes from the manholes was achieved by applying a manhole-pothole classifier trained on the CNN.

3.6 Pothole matching to avoid repeated pothole detections

One pothole may be repeatedly detected multiple times with different camera angles or distances in a video. To avoid repeated detection of the same pothole and to enable pothole detection on videos, a new pothole detection needed to be matched with previously detected potholes within a small region such as three meters of diameter to check if it was already recorded. Since the camera angle and distance

are likely to change, the matching method should be robust to different viewing angles and scales. Therefore, the feature extraction method based on the FAST algorithm (Rosten and Drummond 2005, Rosten *et al*. 2008), short for Features from Accelerated Segment Test, and the feature tracking method based on the Kanade-Lucas-Tomasi (KLT) algorithm (Lucas 1981, Kalal *et al*. 2010) were applied. As shown in the example in Fig. 16, FAST features were identified from the newly detected pothole and matched with previously detected images representing two different cases. Then, in the matching process, the KLT algorithm filtered the non-matching features. In case 1, the previously detected pothole had numerous matching features with the new image, thus the pothole in the new image was considered a repeated detection of the pothole and continued to be tracked in subsequent images. In case 2, the previously detected pothole had zero matching features with the new image, thus the pothole was considered a new pothole detected for the first time, and its snapshot along with the GPS location were transmitted to the cloud server.

4. Conclusions

An autonomous DLPD was proposed for detecting potholes on the roads from images and videos based on the framework of the faster-RCNN method. The DLPD was calibrated through sensitivity tests. Then the detector was unbiasedly evaluated on the testing images. Methods were implemented to differentiate potholes from manholes and to avoid repeated detection of the same potholes. Important conclusions are listed as follows.

1. From the sensitivity tests, it was found the number of epochs and the variety of image sizes had significant impacts on the validation performance of the DLPD. The average precision of the validation results can be improved by increasing the number of epochs to 20 and the variety of image sizes to 2. Therefore, parameters such as the number of epochs and the variety of image sizes were set to the optimal values for training the final DLPD.
2. Potholes were accurately detected by the proposed and trained DLPD with high average precision. The average precision of the DLPD on the testing data of 215 images was 0.938. The accurately detected potholes were encircled by bounding boxes that overlapped well the ground-truth boxes.
3. Imperfect bounding boxes with small errors in terms of the locations or dimensions were observed for potholes with ambiguous boundaries, and resulted in both false negatives and false positives. However, the small errors in the scale of a few decimeters were acceptable for the purpose of detecting and locating potholes on roads.
4. Potholes and manholes were differentiated by classifying the regions within the bounding boxes of the detected objects using a manhole-pothole classifier, which was trained on the CNN using pothole and manhole images.
5. It was demonstrated that repeated detection of the same potholes can be avoided through pothole matching to enable detections on videos by applying the FAST feature extraction and KLT tracking algorithm.

Acknowledgments

This research is partially supported by the National Natural Science Foundation of China under Grant No. U1509205 and No. U1709212. The authors would like to express the appreciation to Lijun Xie, a research assistant at Tsinghua University for her work in collecting the historical data.

References

ASCE (2017), "2017 infrastructure report card", *https://www.infrastructurereportcard.org/cat-item/roads*.

Bousquet, O. and Bottou, L. (2008). "The tradeoffs of large scale learning", *Proc. of NIPS 2008*.

Cha, Y.J., Chen, J. and Büyüköztürk, O. (2017), "Output-only computer vision based damage detection using phase-based optical flow and unscented Kalman filters", *Eng. Struct*., **132**, 300-313. https://doi.org/10.1016/j.engstruct.2016.11.038.

Cha, Y.J., Choi, W., Suh, G., Mahmoudkhani, S. and Büyüköztürk, O. (2017), "Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types", *Comput-Aided Civ. Inf*., **33**, 731-747. https://doi.org/10.1111/mice.12334.

Chen, J.G., Wadhwa, N., Cha, Y.J., Durand, F., Freeman, W.T. and Buyukozturk, O. (2015), "Modal identification of simple structures with high-speed video using motion magnification", *J. Sound Vib.*, **345**, 58-71. https://doi.org/10.1016/j.jsv.2015.01.024.

Chen, Y.L., Jahanshahi, M.R., Manjunatha, P., Gan, W., Abdelbarr, M., Masri, S.F., Becerik-Gerber, B. and Caffrey, J.P. (2016), "Inexpensive multimodal sensor fusion system for autonomous data acquisition of road surface conditions", *IEEE Sen. J*., **16**(21), 7731-7743. DOI: 10.1109/JSEN.2016.2602871

Cord, A. and Chambon, S. (2012), "Automatic road defect detection by textural pattern recognition based on AdaBoost", *Comput-Aided Civ. Inf*., **27**(4), 244-259. https://doi.org/10.1111/j.1467-8667.2011.00736.x.

Debella-Gilo, M. and Kääb, A. (2011), "Sub-pixel precision image matching for measuring surface displacements on mass movements using normalized cross-correlation", *Remote Sens. Environ.*, **115**(1), 130-142. https://doi.org/10.1016/j.rse.2010.08.012.

Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S. and Balakrishnan, H. (2008), "The pothole patrol: using a mobile sensor network for road surface monitoring", *Proc. of MobiCASE 2014*.

Fernandez-Llorca, D., Minguez, R.Q., Alonso, I.P., Lopez, C.F., Daza, I.G., Sotelo, M.A. and Cordero, C.A. (2017), "Assistive intelligent transportation systems: the need for user localization and anonymous disability identification", *IEEE Intel. Transp. Sy*., **9**(2), 25-40. DOI: 10.1109/MITS.2017.2666579.

Girshick, R. (2015), "Fast R-CNN", *Proc. of IEEE ICCV 2015*.

Girshick, R., Donahue, J., Darrell, T. and Malik, J. (2014), "Rich feature hierarchies for accurate object detection and semantic segmentation", *Proc. of IEEE CVPR 2014*.

Greenbaum, R.J., Smyth, A.W. and Chatzis, M.N. (2015), "Monocular computer vision method for the experimental study of three-dimensional rocking motion", *J. Eng. Mech. - ASCE*., **142**(1), 04015062. https://doi.org/10.1061/(ASCE)EM.1943-

7889.0000972.

Jahanshahi, M.R., Chen, F.C., Ansar, A., Padgett, C.W., Clouse, D. and Bayard, D.S. (2017), "Accurate and robust scene reconstruction in the presence of misassociated features for aerial sensing", *J. Comput. Civil Eng.*, **31**(6), 04017056. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000702.

Jahanshahi, M.R., Jazizadeh, F., Masri, S.F. and Becerik-Gerber, B. (2012), "Unsupervised approach for autonomous pavement-defect detection and quantification using an inexpensive depth sensor", *J. Comput. Civil Eng.*, **27**(6), 743-754. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000245.

Jang, J., Yang, Y., Smyth, A.W., Cavalcanti, D. and Kumar, R. (2016), "Framework of data acquisition and integration for the detection of pavement distress via multiple vehicles", *J. Comput. Civil Eng.*, **31**(2), 04016052. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000618.

Jog, G., Koch, C., Golparvar-Fard, M. and Brilakis, I. (2012), "Pothole properties measurement through visual 2D recognition and 3D reconstruction", *Proc. of ASCE 2012*.

Kalal, Z., Mikolajczyk, K. and Matas, J. (2010). "Forward-backward error: Automatic detection of tracking failures", *Proc. of ICPR 2010*.

Kim, S.W., Jeon, B.-G., Kim, N.-S. and Park, J.-C. (2013), "Vision-based monitoring system for evaluating cable tensile forces on a cable-stayed bridge", *Struct. Health Monit.*, **12**(5-6), 440-456. https://doi.org/10.1177/1475921713500513.

Koch, C. and Brilakis, I. (2011), "Pothole detection in asphalt pavement images", *Adv. Eng. Inform.*, **25**(3), 507-515. https://doi.org/10.1016/j.aei.2011.01.002.

Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012). "Imagenet classification with deep convolutional neural networks", *Proc. of ICONIP 2012*.

Lucas, B.D. and Kanade, T. (1981), "An iterative image registration technique with an application to stereo vision", *Proc. of IJCAI 1981*.

Luo, L., Feng, M., Fukuda, Y. and Zhang, C. (2015), "Micro displacement and strain detection for crack prediction on concrete surface using optical nondestructive evaluation methods", *Int. J. Progn. Health Manag.*, **6**(SP3), 1-12.

Luo, L., Feng, M.Q. and Wu, Z.Y. (2018.), "Robust vision sensor for multi-point displacement monitoring of bridges in the field", *Eng. Struct.*, **163**, 255-266. https://doi.org/10.1016/j.engstruct.2018.02.014.

Mednis, A., Strazdins, G., Zviedris, R., Kanonirs, G. and Selavo, L. (2011). "Real time pothole detection using android smartphones with accelerometers", *Proc. of DCOSS 2011*.

Park, K., Torbol, M. and Kim, S. (2018), "Vision-based natural frequency identification using laser speckle imaging and parallel computing", *Comput-Aided Civ. Inf.*, **33**(1), 51-63. https://doi.org/10.1111/mice.12312.

Ren, S., He, K., Girshick, R. and Sun, J. (2015), "Faster r-cnn: Towards real-time object detection with region proposal networks", *Proc. of ICONIP 2015*.

Rosten, E. and Drummond, T. (2005), "Fusing points and lines for high performance tracking", *Proc. of ICCV 2005*.

Rosten, E., Porter, R. and Drummond, T. (2008), "Faster and better: A machine learning approach to corner detection", *IEEE T. Pattern Anal.*, **32**(1), 105-119. DOI: 10.1109/TPAMI.2008.275.

Uijlings, J.R., Van De Sande, K.E., Gevers, T. and Smeulders, A.W. (2013), "Selective search for object recognition", *Int. J. Comput. Vision.*, **104**(2), 154-171. https://doi.org/10.1007/s11263-013-0620-5.

Ye, X.W., Ni, Y.Q., Wai, T.T., Wong, K.Y., Zhang, X.M. and Xu, F. (2013), "A vision-based system for dynamic displacement measurement of long-span bridges: algorithm and verification", *Smart Struct. Syst.*, **12**(3-4), 363-379. http://dx.doi.org/10.12989/sss.2013.12.3_4.363.

Ye, X.W., Dong, C.Z. and Liu, T. (2016a), "Force monitoring of steel cables using vision-based sensing technology: methodology and experimental verification", *Smart Struct. Syst.*, **18**(3), 585-599. http://dx.doi.org/10.12989/sss.2016.18.3.585

Ye, X.W., Dong, C.Z. and Liu, T. (2016b), "Image-based structural dynamic displacement measurement using different multi-object tracking algorithms", *Smart Struct. Syst.*, **17**(6), 935-956. http://dx.doi.org/10.12989/sss.2016.17.6.935.

Yoon, H., Shin, J. and Spencer, B.F. (2018), "Structural displacement measurement using an unmanned aerial system", *Comput-Aided Civ. Inf.*, **33**(3), 183-192.