Automatic detection method of cracks from concrete surface imagery using two-step light gradient boosting machine

Pang-jo Chun¹ | Shota Izumi² | Tatsuro Yamane³

¹Department of Civil Engineering, Graduate School of Engineering, The University of Tokyo, Tokyo, Japan
²Department of Civil and Environmental Engineering, Ehime University, Matsuyama, Ehime, Japan
³Department of International Studies, Graduate School of Frontier Sciences, The University of Tokyo

Abstract
Automated crack detection based on image processing is widely used when inspecting concrete structures. The existing methods for crack detection are not yet accurate enough due to the difficulty and complexity of the problem; thus, more accurate and practical methods should be developed. This paper proposes an automated crack detection method based on image processing using the light gradient boosting machine (LightGBM), one of the supervised machine learning methods. In supervised machine learning, appropriate features should be identified to obtain accurate results. In crack detection, the pixel values of the target pixels and geometric features of the cracks that occur when they are connected linearly should be considered. This paper proposes a methodology for generating features based on pixel values and geometric shapes in two stages. The accuracy of the proposed methodology is investigated using photos of concrete structures with adverse conditions, such as shadows and dirt. The proposed methodology achieves an accuracy of 99.7%, sensitivity of 75.71%, specificity of 99.9%, precision of 68.2%, and an F-measure of 0.6952. The experimental results demonstrate that the proposed method can detect cracks with higher performance than the pix2pix-based approach. Furthermore, the training time is 7.7 times shorter than that of the XGBoost and 2.3 times shorter than that of the pix2pix. The experimental results demonstrate that the proposed method can detect cracks with high accuracy.

1 | INTRODUCTION
Deterioration and damage of concrete structures, such as bridges and tunnels, have become a serious problem in many countries. To develop optimum maintenance plans for such structures in terms of cost and safety, accurately evaluating the degree of deterioration is necessary. Cracks discovered during inspections could help the person performing the bridge diagnosis to understand the damaged state of the concrete structures. For example, according to Japan’s bridge and tunnel periodic inspection guidelines, inspectors are required to visually inspect concrete structures (Ministry of Land, Infrastructure, Transport and Tourism, 2019a, 2019b). However, these inspections tend to be subjective and nonquantitative, leading to variations in damage assessments among inspectors. For example, assessment results on cracks can vary significantly depending on factors, such as the inspector’s level of knowledge. In addition, visual inspection of difficult to access locations, such as high or narrow areas, is not always possible. Thus, several methods have been recently proposed to automatically detect cracks from images of concrete surfaces. These methods are expected to be quantitative and applicable...
in evaluating areas that are difficult to access or see directly. Moreover, they offer an opportunity to drastically reduce the inspection time and labor, providing that only photography is required in the field.

Luxmoore (1973) is one of the first to discuss crack detection using holography, along with its range of application. Some years later, a drastic improvement has been observed in hardware, such as photographic instruments and computers, and software offering image processing algorithms. Abdel-Qader, Abudayyeh, and Kelly (2003) demonstrated the superiority of the fast Haar wavelet transform in detection accuracy when applied to processing images of 50 bridges compared with the fast Fourier transform, Sobel operator, and Canny algorithm. Many other studies have also performed frequency analysis. For example, Hutchinson and Chen (2006) used the Canny filter and fast Haar wavelet transform to detect cracks. In the analysis, parameters were estimated based on the receiver operating characteristic analysis. Furthermore, research projects have focused on geometrical features. For example, Iyer and Sinha (2006) used curvature feature evaluation with morphological processing in their crack detection method. Sinha and Fieguth (2006) developed an algorithm based on morphological operations to accurately segment pipe cracks, holes, joints, laterals, and collapsed surfaces and classify defects in underground pipes. Hashimoto and his research team proposed a crack detection technique using the grayscale Hough transform and percolation processing (Yamaguchi & Hashimoto, 2008, 2009, 2010; Yamaguchi, Nakamura, Saegusa, & Hashimoto, 2008). Their method was a significant accomplishment that demonstrated an improvement in detection accuracy by focusing on a crack’s linearity and continuity. Another research project focused on such geometrical features, Fujita et al. proposed an extremely robust method for detecting cracks by effectively utilizing probabilistic relaxation, line extension processing, and multiscale line enhancement processing using Hessian matrices (Fujita & Hamamoto, 2009, 2011; Fujita, Mitani, & Hamamoto, 2006). Aramisiri, Gunaratne, and Sarkar (2009) proposed a crack detection technique using the bidirectional reflection distribution function derived from a model based on the Gaussian reflectance model introduced by Ward, and the research by Lee, Kim, Yi, and Kim (2013) combining binarization with noise processing and thinning processing. They also used a method proposed by Liu, Cho, Spencer, and Fan (2014) and Liu, Nie, Fan, and Liu (2019) introducing adaptive combining crack detection by Niblack’s method and three-dimensional reconstruction from a UAV image, and a method proposed by Sohn, Lim, Yun, and Kim (2005) that employed three-dimensional analysis combined with the Hough transform and filter processing. The different methods for crack detection are summarized in a review paper by Koch, Georgieva, Kasireddy, Akinci, and Fieguth (2015).

Although the existing image analysis techniques have their advantages, their performance depends on the quality of set parameters. To find an optimal parameter set, some studies limit the object and conditions of photography, whereas others utilize parameter optimization methods. For instance, Nguyen, Kawamura, Shiozaki, and Tarighat (2018) proposed a method for automatically detecting threshold values using an interactive genetic algorithm. Nishikawa, Yoshida, Sugiyama, and Fujino (2012) constructed an image filter with a tree structure using a genetic algorithm and evaluated the width and vector of target cracks.

Several studies have employed supervised machine learning for crack detection. For example, Prasanna et al. (2014) compared support vector machine (Tschantaridis, Joachims, Hofmann, & Altun, 2005), random forest (Breiman, 2001), and AdaBoost (Freund & Schapire, 1996). Deep learning has gained popularity in recent years owing to its ability to automatically detect features and was used in civil engineering applications (Chun, Yamane, Izumi, & Kameda, 2019; Rafiei & Adeli, 2017a, 2017b, 2018a, 2018b; Rafiei, Khoshefat, Demirboga, & Adeli, 2017). Cha, Choi, and Büyükköztürk (2017) developed a method based on a convolutional neural network (CNN). The method determines whether a crack exists in a small image window of 256 × 256 pixels. Other studies on crack detection using CNNs have also been conducted (e.g., Cha et al., 2017; Cha, Choi, Suh, Mahmoudkhan, & Büyükköztürk, 2018; Deng, Lu, & Lee, 2019; Dung, 2019; Jiang & Zhang, 2019; Li & Zhao, 2019; Silva & Lucena, 2018; Xue & Li, 2018; Yang et al., 2018; Zhang, Rajan, & Story, 2019). As extensions of CNNs, methods such as pix2pix (Kobayashi, 2018), deep convolutional encoder–decoder network (Bang, Park, Kim, & Kim, 2019), CrackNet (Zhang et al., 2018), and DeepCrack (Liu, Yao, Lu, Xie, & Li, 2019), were recently proposed. CNN-based models, however, take long to determine whether each pixel in an image represents a crack or not because many features must be considered.

In this study, we developed a method for detecting cracks based on the light gradient boosting machine (LightGBM) (Ke et al., 2017), which is a highly efficient gradient boosting decision tree. Similar to other supervised machine learning methods, the performance of LightGBM depends on the feature set. In crack detection, the target pixel values and geometric features of cracks, such as whether they are connected linearly, should be considered. In this study, we used a methodology for generating features based on pixel values and geometric shapes in two stages. In the first stage, we propose a new method for generating features that explain the crack characteristics using square and nonsquare filters, which could not be obtained in previous studies. Then, in the second stage, the geometric characteristics of connected components derived in the first stage are used as features of LightGBM. Considering the two types of features allows for achieving high crack detection accuracy under various conditions.
Concrete crack images
Conversion to grayscale images
(Section 2.1)
Correction using median filter
(Section 2.2)
Crack detection taking into account
features based on pixel values(Section 2.3)
Crack detection taking into account
features based on geometric shapes(Section 2.4)
Crack detection results

**Figure 1** Flowchart of the proposed crack detection method

**Figure 2** Example of generating a grayscale image using the maximum value of the RGB components (left: before grayscale, right: after grayscale)

## 2 | METHODS FOR DETECTING CRACKS

Figure 1 illustrates the flowchart of the crack detection method proposed in this study. Each element is explained below in turn.

### 2.1 | Converting to grayscale image

First, we reduce the calculation time by converting the captured image to a grayscale image. For the grayscale conversion, the mean value of the RGB components or the weighted mean value using NTSC coefficients is typically used. However, in this study, we perform the grayscale conversion using the maximum value of the RGB components. By doing so, graffiti with different RGB components is relatively lightened to prevent it from being mistaken for a crack. Figure 2 shows an example of this correction; the red graffiti in the left image was removed resulting in the right image.

### 2.2 | Correction using a median filter

Median filters are often applied to images as a preprocessing step to correct contrast differences resulting from light and shadows or contamination (e.g., Fujita et al., 2006). The correction formula is defined as

\[
I_a(i, j) = \frac{I_b(i, j) \times 0.5b_m}{I_m(i, j)}
\]

where \(i\) and \(j\) denote the position of the target pixel, \(I_a\) denotes the image after the correction, \(I_b\) denotes the image before the correction, and \(I_m\) denotes the result of applying the median filter to the image before the correction. Furthermore, \(b_m\) denotes the maximum pixel value; for example, in the often used 256-gradient scale, this value will be 255. The size of the median filter affects the correction precision. On the basis of the study by Fujita et al. (2006), we set the filter size to \(41 \times 41\). However, the effect of the filter size on results is not strong. For example, using a \(21 \times 21\) median filter will result in extracted cracks being slightly thinner than those extracted using a \(41 \times 41\) filter, whereas a \(61 \times 61\) median filter will produce slightly thicker cracks. When the filter size is small, the pixel value in the denominator of Equation (1) becomes smaller near cracks, leading to larger values for \(I_a\) and creating brighter corrections. In the images used in this study, the crack width is almost less than 50 pixels. With this crack width, the parameters in this section and the subsequent sections work effectively. If the crack width is significantly larger than the filter size, the cracks become less noticeable after correction because large shadows receive a similar treatment. If such a phenomenon occurs with thick cracks, the filter size must be increased. Figure 3 illustrates an example of the correction using a \(41 \times 41\) filter. The figure also shows that the stain and shadows are properly corrected without losing the crack. As a quantitative index, we calculate the separation metrics proposed by Otsu (1979) as defined by Equation (2), which explains the degree of separation of two classes.

\[
\sigma^2_{\text{within}} = \frac{N_1 \sigma^2_1 + N_2 \sigma^2_2}{N_1 + N_2}
\]

\[
\sigma^2_{\text{between}} = \frac{N_1(m_1 - m_t)^2 + N_2(m_2 - m_t)^2}{N_1 + N_2}
\]

\[
\text{separation metrics} = \frac{\sigma^2_{\text{between}}}{\sigma^2_{\text{within}}}
\]

where \(\sigma^2_{\text{within}}\) denotes the within-class variance, \(\sigma^2_{\text{between}}\) denotes the between-class variance, \(N_i\) denotes the number of pixels of class \(i\), \(\sigma^2_i\) denotes the variance of class \(i\), \(m_i\) denotes the mean value of class \(i\), and \(m_t\) denotes the mean value of all pixel values. The separation metrics increases from 0.0147 to 0.0325 prior and subsequent to the correction. Furthermore, Figure 4 shows the change in the pixel value...
2.3 Crack detection considering features based on pixel values

2.3.1 LightGBM

This section outlines the proposed method for establishing whether each pixel in an image represents a crack or not. According to supervised machine learning, many combinations of the input and output should be prepared by a human expert in advance as a training dataset. After learning the relationships between inputs and outputs, the method can accurately estimate outputs for new inputs not included in the training dataset.

Figure 5 shows an overview of the learning process of the proposed method. As shown on the left side of Figure 5 (labeled as “training stage”), the classifier is trained using a training dataset explaining the relationship between the features (in the “original images” frame) and the cracked/not cracked categories (the “ground truth images” frame) of the target pixels. After the training process, the classifier is able to detect cracks with high precision, even in new photographic images that were not used in training (see the right side of Figure 5 labeled as “analysis stage”).

In this study, we used LightGBM as a supervised machine learning algorithm. LightGBM is a gradient boosting framework that uses tree-based learning algorithms (Ke et al., 2017). LightGBM is known to have the following merits: fast training speed, high efficiency, low memory usage, high accuracy, supported parallel learning, and capability in handling large-scale data. Another popular gradient boosting algorithm is XGBoost (Chen & Guestrin, 2016). XGBoost and LightGBM are decision-tree-based ensemble methods; they differ in the tree growth procedure (Figure 6). The majority of decision-tree learning algorithms, including XGBoost, grow trees level-wise, as illustrated in the top image of Figure 6. The bottom of Figure 6, however, illustrates that the LightGBM grows trees leaf-wise by determining the branch point using a histogram-based algorithm. The leaf-wise tree growth could easily increase the complexity of the structure compared
Different ways of growing decision trees employed in decision-tree learning algorithms (Microsoft, 2019)

![Level-wise tree growth](image1)

![Leaf-wise tree growth](image2)

**Figure 6**

**Table 1** Parameters used in LightGBM

| Parameter                  | Value                |
|----------------------------|----------------------|
| Number of iterations       | 2,000                |
| Early stopping             | 10                   |
| Number of leaves           | 200                  |
| Learning rate              | 0.02                 |
| Maximum number of bins     | 100                  |
| Minimum number of data in a leaf | 20          |
| Maximum depth              | Unlimited            |
| Loss function              | Multiclass logarithmic loss |

When using LightGBM, parameters, such as the number of iterations, must be specified. In this study, we used the parameters listed in Table 1. The details of these parameters are described in Microsoft (2019). However, doubling or halving the values of these parameters, except for the number of iterations, did not significantly affect the detection results.

**2.3.2 Features**

The LightGBM model was built to classify all pixels in an image as those representing cracks and those that do not. Pixels in the areas representing cracks relative appear darker, and their values are mostly low compared with the pixels in the areas surrounding cracks. Therefore, we used the target pixel value as a feature. The use of this feature alone will lead to a high rate of false detections because photo images of concrete surfaces contain other areas with low brightness (such as those covered in dirt) in addition to cracks. Hence, we also used the values of the pixels surrounding the target pixel as features. These values will be low after applying the Gaussian and median filters to the region that includes the target pixel as cracks in images are often collections of dark pixels. The Gaussian filter can be formulated as:

$$f'(i, j) = \sum_{l=-n}^{n} \sum_{k=-m}^{m} f(i+k, j+l)h(k, l)$$

where $h(k, l) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(k^2+l^2)}{2\sigma^2}\right)$

(3)

where $f(i, j)$ denotes the pixel values of the input image, $f'(i, j)$ denotes the pixel values of the output image, $i, j$ denote the coordinates of the target pixel in the image, and $h(k, l)$ represents a Gaussian filter with a size of $(2m + 1) \times (2n + 1)$. $\sigma$ denotes the standard deviation that affects the weight of the distant pixels. In this study, $\sigma$ was set as $1/2$ of the long side of the filter. Note that we have confirmed that the change in $\sigma$ has no significant effect on the accuracy. We also used eight types of square kernels: Gaussian filters with sizes of $7 \times 7$, $31 \times 31$, $55 \times 55$, $105 \times 105$, and $205 \times 205$ and median filters with sizes $7 \times 7$, $15 \times 15$, and $21 \times 21$. The sizes of the median filters were slightly smaller than those of the Gaussian filters. This reflects that, when the filter area is large, the median value is strongly affected by a part not representing a crack rather than that obtained after applying the Gaussian filter, and hence, it cannot function as a feature for crack detection.

Furthermore, we applied nonsquare kernels to extract cracks as cracks are usually directional, that is, they stretch in the longitudinal, lateral, or diagonal direction. For example, in the case of the horizontal crack shown in Figure 7, the weighted mean pixel value gets low within a horizontal region (yellow rectangle); hence, we can consider this value to be an effective characteristic value. On the contrary, although the longitudinal region (blue rectangle) in Figure 7 is not a
dominant feature for the horizontal crack, it is effective for the longitudinal crack. For the diagonal crack illustrated in Figure 7, the weighted mean pixel value of the obliquely inclined rectangular region (green rectangle) is an effective feature. In this study, a total of eight directional filters were applied in 22.5-degree increments, as shown in Figure 8. We used six types of nonsquare filters: Gaussian filters with sizes $5 \times 5$, $5 \times 10$, $5 \times 20$, and $5 \times 40$ and median filters with sizes $5 \times 21$ and $5 \times 61$.

An example of applying these nonsquare filters is shown in Figure 9, which also shows that the horizontal crack is emphasized when the horizontal filter is used. The same case applies to the vertical and diagonal directions.

In addition, the results of the Canny filters (Canny, 1986), which are suitable for edge extraction, are used as features. According to the Canny method, defining two main thresholds, namely, maxVal and minVal, is necessary (OpenCV, 2015). First, the edge gradient and direction for each pixel are calculated. Then, any edges with the intensity gradient more than maxVal are ascertained to be edges and those below minVal are ascertained to be nonedges and discarded. The edges lying between the two thresholds are classified as edges or nonedges based on their connectivity. This procedure can detect strong edges in images.

As the Canny filter detects only edges, its results are not suitable as features for detecting the inside of cracks (Figure 10b). Therefore, the Gaussian filter with a size of $41 \times 41$ is used after applying the Canny filter to obtain the features. An example of this application is shown in Figure 10c.

Table 2 lists the features discussed thus far. In summary, a total of 74 features were considered, including the target pixel value (one in subtotal), pixel values after applying the Gaussian filters (37 in subtotal), median filters (21 in subtotal), and Canny filters (15 in subtotal).

Figure 11b illustrates an example of the analysis of the images shown in Figure 11a using LightGBM with 74 features. The pixels highlighted in red indicate the pixels detected as those representing cracks. Although the crack was
mostly successfully detected, many misjudgment results were obtained due to noise. In the next section, we present a method that considers geometric features to remove these misjudgment results.

### 2.4 Crack detection considering features representing the geometric shape of the crack areas

We focused on the geometric configuration of cracked areas to eliminate remaining noise or false detections in the analysis results discussed in the previous section. Commonly used denoising techniques can remove the pixels detected as those representing cracks when the area of connected components, i.e., clusters of crack pixels, which are connected to each other, is smaller than a predefined threshold. However, in this study, the following geometrical characteristics were considered based on the fact that cracks are typically slender: area, perimeter, eccentricity, and length of the major and minor axes of the ellipse that has the same normalized second central moments as the connected components. If the area of the connected region is too small, it is likely to be noise. Furthermore, considering that the smaller the area/perimeter, the more likely it is to be slender, we consider the area and the perimeter as geometrical characteristics. Slenderness also influences the ratio of $f_3, f_4,$ and $f_5$. In some cases, however, one of these indices becomes large despite not being slender. Therefore, we choose all of them as geometrical characteristics.

Predefining independent thresholds for each detection criteria is not always appropriate. For example, even if the size of the area of connected components is smaller than a predefined threshold, it is still likely to be a crack if it is long and narrow. Therefore, we applied several detection criteria in combination, in addition to using LightGBM. Of note, the LightGBM in this section was different from that described in Section 2.3. In particular, the target of LightGBM presented in this section was the connected components, as opposed to pixels used as a target in LightGBM presented in the previous section.

Figure 12 illustrates a schematic diagram of the analysis presented in this section. In particular, each connected component obtained by analyzing the training image dataset using the method presented in Section 2.3 becomes the training data for the method presented in this section. All connected components have the aforementioned five features and an answer label indicating whether they represent a crack or not. However, the amount of training data is insufficient in this case. Therefore, we propose a technique to augment the data by repeatedly dividing the image randomly and obtaining data from each image, as shown in Figure 13. The connected component colored yellow on the right side of Figure 13 is a newly formed component after applying the proposed technique. The problem of insufficient training data is therefore solved by obtaining and accumulating the features of such newly generated components. Other than the developed method, data augmentation by rotation or shear deformation of the image
may improve the accuracy, but no significant effect was found in the data set of this study. However, it should be noted that it may be effective when the shooting conditions or structural conditions change.

3 | EXPERIMENTAL RESULTS USING ACTUAL IMAGES

In this section, we discuss the results of applying the proposed method to 98 digital images \((5184 \times 3456\) pixels) captured from the decks and girders of concrete bridges, and the inner and outer walls of buildings in Japan under various conditions. The data were uploaded to a website: (https://sites.google.com/view/pchun/). The specifications of the computer used for training the LightGBM were as follows: Nvidia Quadro RTX 8000, Intel® Xeon W-2195 CPU @ 2.30 GHz CPU, 512 GB RAM, and Windows 10 OS.

We evaluated the effectiveness of our method using K-fold cross-validation with K set to 10.

Figure 14 illustrates some examples of the detection results obtained by applying the proposed method and pix2pix to six images (Isola, Zhu, Zhou, & Efros, 2017). Pix2pix is an image generation algorithm using GAN that learns the mapping from an input image to an output image. With the pix2pix, we generated images showing the segmentation results of cracks from the images of concrete surfaces. We tried various semantic segmentation methods, including the Mask R-CNN (He, Gkioxari, Dollár, & Girshick, 2017), fully convolutional network (Long, Shelhamer, & Darrell, 2015), and CrackNet (Zhang et al., 2018) but pix2pix was the most accurate for the data set used in this study. In this paper, since the accuracy almost converged at 400 epochs, the results at that time were used. In Figure 14, we colored TP red, FP blue, and FN green, and did not color TN. TP, TN, FP, and FN stand for true positive, true negative, false positive, and false negative, respectively (Table 3). Note that the results shown in Figure 14 are not necessarily obtained from the same model because the K-fold cross-validation produced 10 different models.

Figure 14 shows photos of concrete structures with adverse conditions. In particular, the first image contains relatively large areas of shadows, the second and third images contain large stains, the fourth image contains both large areas of shadow and large stains, the fifth image contains many small thin cracks, and the sixth image contains formwork marks.

The proposed method achieved a satisfactory level of detection accuracy from the first to the fourth image, whereas pix2pix demonstrated less accurate results. In particular, pix2pix falsely detected darker areas with low pixel values, such as stains. However, there are some cases where the proposed method caused a detection error, and the typical examples are the fifth image and sixth image. As this method uses Gaussian and median filters, as described in Section 2.3.2, it is difficult to recognize the exact boundaries of crack and non-crack. Consequently, there is a drawback that the detection accuracy of fine cracks becomes low, especially the finest part \((1–2\) pixels width) in the fifth image is FN (green). The sixth image shows FP (blue) in cone holes and formwork marks with similar features to cracks. The detection error caused by these fine cracks and plastic corn and formwork, however, is the same for the detection by pix2pix. Such complicated cases would be addressed in our future work.

To further evaluate the crack detection performance of the proposed method, we calculated the accuracy rate, sensitivity, specificity, precision, and F-measure as per with the following equations:

\[\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad (4)\]

\[\text{Sensitivity} = \frac{TP}{TP + FN} \quad (5)\]

\[\text{Specificity} = \frac{TN}{FP + TN} \quad (6)\]

\[\text{Precision} = \frac{TP}{TP + FP} \quad (7)\]

\[F\text{-measure} = \frac{2 \times \text{Sensitivity} \times \text{Precision}}{\text{Sensitivity} + \text{Precision}} \quad (8)\]

Table 4 shows the average values of the considered indices and their standard deviations based on the 98 images used in this study for the following methods: proposed method considering features derived from the pixel values and geometric shapes (Section 2.1–2.4), methods considering only features derived from the pixel values (Section 2.1–2.3), and the pix2pix.

The accuracy and specificity of the proposed method and pix2pix are relatively high because many pixels correspond to TN in Table 3. Thus, it cannot be said that the crack detection performance can be evaluated only using these indices. In other words, a comprehensive examination of the five indices in Table 4 is necessary, and all indices for the proposed method are higher compared with that achieved by the other two methods. The pix2pix has especially low precision. This is caused by a large number of FP pixels that are not
| Original image | Proposed method | pix2pix |
|---------------|----------------|--------|
| ![Original image](image1) | ![Proposed method](image2) | ![pix2pix](image3) |
| ![Original image](image4) | ![Proposed method](image5) | ![pix2pix](image6) |
| ![Original image](image7) | ![Proposed method](image8) | ![pix2pix](image9) |
| ![Original image](image10) | ![Proposed method](image11) | ![pix2pix](image12) |
| ![Original image](image13) | ![Proposed method](image14) | ![pix2pix](image15) |
| ![Original image](image16) | ![Proposed method](image17) | ![pix2pix](image18) |
| ![Original image](image19) | ![Proposed method](image20) | ![pix2pix](image21) |

**Figure 14** Crack detection results by the proposed method and pix2pix
original cracks but are erroneously detected as cracks. Sensitivity is also low, and consequently, the F-measure, obtained as the harmonic mean of precision and sensitivity, which is often used in the comprehensive evaluation of accuracy and completeness, also indicates a low detection performance of pix2pix. By contrast, the proposed method has higher values of sensitivity, specificity, precision, and F-measure, which is reflected in the visual accuracy of the detection results. The standard deviation of the F-measure also indicates that the proposed method is significantly better than the other two methods.

Accuracy and learning time were compared. The training time of 400 epochs of pix2pix is 41,321 seconds, while that of our method is 17,941 seconds, which is 2.3 times shorter, showing the superiority of LightGBM as a leaf-wise tree growth method.

If we use an extremely small number of epochs for pix2pix training or an extremely small number of CNNs with a small number of layers, the learning and analysis time can be significantly reduced. However, because the accuracy and speed have trade-off relationships, it is insignificant to compare the computational speed of the two methods unless the accuracy is comparable.

### 4 | CONCLUSIONS

This paper proposed a method for detecting cracks on concrete surfaces in two stages. The method employs LightGBM and considers features derived from pixel values and geometric shapes of cracks. The paper proposed a simple way of deriving features from pixel values by using square, nonsquare, and Canny filters. The results show that the pixel values and geometric shapes of cracks, including their size and slenderness, should be considered. Moreover, cracks have been proven to be detected very well even in images containing shadows and dirt. The proposed methodology achieves an accuracy of 99.7%, sensitivity of 75.71%, specificity of 99.9%, precision of 68.2%, and F-measure of 0.6952.

As many features were employed as inputs to LightGBM, some of them inevitably strongly relate to each other. Although a larger number of features may lead to a high accuracy, the calculation time may also increase. In the future, we plan to investigate the application of principal component analysis and stepwise forward selection method to reduce the number of features and shorten the calculation time. Within the scope of this study, the adverse effect of multicollinearity could not be confirmed from the viewpoint of accuracy.

Furthermore, traces of formwork and plastic cone holes that have similar characteristics to cracks were erroneously recognized as cracks. This problem can be solved by adding geometric features, such as those representing formwork traces with a very high linearity and plastic cone-hole traces with a circular shape. Another promising approach is to use the Mask R-CNN method to detect the traces of formwork and plastic cone holes.

### REFERENCES

Abdel-Qader, I., Abudayyeh, O., & Kelly, M. (2003). Analysis of edge-detection techniques for crack identification in bridges. *Journal of Computing in Civil Engineering, 17*(4), 255–263.

Amarasiri, S., Gunaratne, M., & Sarkar, S. (2009). Modeling of crack depths in digital images of concrete pavements using optical reflection properties. *Journal of Transportation Engineering, 136*(6), 489–499.

Bang, S., Park, S., Kim, H., & Kim, H. (2019). Encoder–decoder network for pixel-level road crack detection in black-box images. *Computer-Aided Civil and Infrastructure Engineering, 34*(8), 713–727.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 8*(6), 679–698.

Cha, Y. J., Choi, W., & Büyükoztürk, O. (2017). Deep learning-based crack damage detection using convolutional neural networks.
Computer-Aided Civil and Infrastructure Engineering, 32(5), 361–378.

Cha, Y. J., Choi, W., Suh, G., Mahmoudkhani, S., & Büyüköztürk, O. (2018). Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. Computer-Aided Civil and Infrastructure Engineering, 33(9), 731–747.

Chen, T., & Guerstrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA (pp. 785–794). New York: Association for Computing Machinery.

Chun, P. J., Yamane, T., Izumi, S., & Kameda, T. (2019). Evaluation of tensile performance of steel members by analysis of corroded steel surface using deep learning. Metals, 9(12), 1259.

Deng, J., Lu, Y., & Lee, V. C. S. (2019). Concrete crack detection with handwriting script interferences using faster region-based convolutional neural network. Computer-Aided Civil and Infrastructure Engineering, 35(4), 373–388.

Dung, C. V. (2019). Autonomous concrete crack detection using deep fully convolutional neural network. Automation in Construction, 99, 52–58.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. Proceedings of the Thirteenth International Conference on International Conference on Machine Learning, July 1996 (Vol. 96, pp. 148–156). San Francisco, CA: Morgan Kaufmann Publishers.

Fujita, Y., & Hamamoto, Y. (2009). A robust method for automatically detecting cracks on noisy concrete surfaces. Paper presented at Next-Generation Applied Intelligence, 22nd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Tainan, Taiwan, June 20–24, pp. 76–85.

Fujita, Y., & Hamamoto, Y. (2011). A robust automatic crack detection method from noisy concrete surfaces. Machine Vision and Applications, 22(2), 245–254.

Fujita, Y., Mitani, Y., & Hamamoto, Y. (2006). A method for crack detection on a concrete structure. Proceeding of the 18th International Conference on Pattern Recognition, 3, 901–904.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, pp. 2961–2969.

Hutchinson, T. C., & Chen, Z. (2006). Improved image analysis for evaluating concrete damage. Journal of Computing in Civil Engineering, 20(3), 210–216.

Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 21–26 July, Honolulu, HI, pp. 1125–1134.

Iyer, S., & Sinha, S. K. (2006). Segmentation of pipe images for crack detection in buried sewers. Computer-Aided Civil and Infrastructure Engineering, 21(6), 395–410.

Jiang, S., & Zhang, J. (2019). Real-time crack assessment using deep neural networks with wall-climbing unmanned aerial system. Computer-Aided Civil and Infrastructure Engineering, https://doi.org/10.1111/mice.12519

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., … Liu, T. Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. Advances in Neural Information Processing Systems, 3146–3154. Retrieved from http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision
Rafiei, M. H., & Adeli, H. (2017b). A new neural dynamic classification algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 28(12), 3074–3083.

Rafiei, M. H., & Adeli, H. (2018a). A novel unsupervised deep learning model for global and local health condition assessment of structures. *Engineering Structures*, 156, 598–607.

Rafiei, M. H., & Adeli, H. (2018b). Novel machine-learning model for estimating construction costs considering economic variables and indexes. *Journal of Construction Engineering and Management*, 144(12), 04018106.

Rafiei, M. H., Khushefati, W. H., Demirboga, R., & Adeli, H. (2017). Supervised deep restricted Boltzmann machine for estimation of concrete. *ACI Materials Journal*, 114(2), 237–244.

Silva, W. R. L. D., & Lucena, D. S. D. (2018). Concrete cracks detection based on deep learning image classification. Presented at the 18th International Conference on Experimental Mechanics (ICEM18), Brussels, Belgium, 1–5 July 2018.

Sinha, S. K., & Fieguth, P. W. (2006). Segmentation of buried concrete pipe images. *Automation in Construction*, 15(1), 47–57.

Sohn, H. G., Lim, Y. M., Yun, K. H., & Kim, G. H. (2005). Monitoring crack changes in concrete structures. *Computer-Aided Civil and Infrastructure Engineering*, 20(1), 52–61.

Tschantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6, 1453–1484.

Xue, Y., & Li, Y. (2018). A fast detection method via region-based fully convolutional neural networks for shield tunnel lining defects. *Computer-Aided Civil and Infrastructure Engineering*, 33(8), 638–654.

Yamaguchi, T., & Hashimoto, S. (2008). Improved percolation-based method for crack detection in concrete surface images. Proceeding of 2008 19th International Conference on Pattern Recognition, 8–1 December, Tampa, FL, pp. 1–4.

Yamaguchi, T., & Hashimoto, S. (2009). Practical image measurement of crack width for real concrete structure. *Electronics and Communications in Japan*, 92(10), 1–12.

Yamaguchi, T., & Hashimoto, S. (2010). Fast crack detection method for large-size concrete surface images using percolation-based image processing. *Machine Vision and Applications*, 21(5), 797–809.

Yamaguchi, T., Nakamura, S., Saegusa, R., & Hashimoto, S. (2008). Image-based crack detection for real concrete surfaces. *IEEE Transactions on Electrical and Electronic Engineering*, 3(1), 128–135.

Yang, X., Li, H., Yu, Y., Luo, X., Huang, T., & Yang, X. (2018). Automatic pixel-level crack detection and measurement using fully convolutional network. *Computer-Aided Civil and Infrastructure Engineering*, 33(12), 1090–1109.

Zhang, X., Rajan, D., & Story, B. (2019). Concrete crack detection using context aware deep semantic segmentation network. *Computer-Aided Civil and Infrastructure Engineering*, 34(11), 1–21.

Zhang, A., Wang, K. C., Fei, Y., Liu, Y., Tao, S., Chen, C., … Li, B. (2018). Deep learning-based fully automated pavement crack detection on 3D asphalt surfaces with an improved CrackNet. *Journal of Computing in Civil Engineering*, 32(5), 04018041.

---

**How to cite this article:** Chun P-j, Izumi S, Yamane T. Automatic detection method of cracks from concrete surface imagery using two-step light gradient boosting machine. *Comput Aided Civ Inf*. 2021;36:61–72. [https://doi.org/10.1111/mice.12564](https://doi.org/10.1111/mice.12564)