Automatic Detection of Wind Turbine Blade Surface Cracks Based on UAV-taken Images

Long Wang, *Student Member*, *IEEE*, and Zijun Zhang, *Member*, *IEEE*

***Abstract*--In this paper, a data-driven framework is proposed to automatically detect wind turbine (WT) blade surface cracks based on images taken by unmanned aerial vehicles (UAVs). Haar-like features are applied to depict crack regions and train a cascading classifier for detecting cracks. Two sets of Haar-like features, the original and extended Haar-like features, are utilized. Based on selected Haar-like features, an extended cascading classifier is developed to perform the crack detection through stage classifiers selected from a set of base models, the LogitBoost, Decision Tree (DT), and Support Vector Machine (SVM). In the detection, a scalable scanning window is applied to locate crack regions based on developed cascading classifiers using the extended feature set. The effectiveness of the proposed data-driven crack detection framework is validated by both UAV-taken images collected from a commercial wind farm and artificially generated. The extended cascading classifier is compared with a cascading classifier developed by the LogitBoost only to show its advantages in the image-based crack detection. A computational study is performed to further demonstrate the success of the proposed framework in identifying the number of cracks and locating them in original images.**

***Index Terms*--Blade image, crack detection, data-driven model, Haar-like features, wind turbine**

I. Introduction

As commercial wind turbines (WTs) are typically located in remote wilds and exposed to harsh working conditions, they suffer from highly variable loads which result in intense mechanical stresses and frequent failures [1]. A significant operations and maintenance (O&M) cost which accounts for 25% - 30% of the overall wind power generation cost has been reported [2]. A commercial wind farm can cover a broad region and contain numerous sparsely distributed large-size wind turbines. Thus, the O&M knowledge of traditional power plants is not directly applicable. Novel approaches for improving efficiencies of the wind farm O&M have attained a top priority in the recent demand of the wind energy industry.

Manuscript received September 22, 2016; revised December 12, 2016 and January 26, 2017; accepted February 22, 2017.

L. Wang and Z. Zhang are with the Department of Systems Engineering and Engineering Management, City University of Hong Kong, Kowloon, Hong Kong (E-mail: long.wang@my.cityu.edu.hk, zijzhang@cityu.edu.hk).

Due to the aging of WTs, a growing rate of wind turbine blade failures has been observed [3]. As a major component of WTs, failures of WT blades can lead to the significant capital loss and unscheduled downtime [4]. Previous studies [5], [6] reported that each blade failure could result in a downtime of more than 7 days. Pulsating wind loads and environmental factors, such as the lightning, dust, and icy weather, actually make blades fragile parts of WTs [7]. Therefore, approaches for monitoring the WT blade health are highly valuable and deserve more research efforts.

Monitoring WT blade conditions and identifying blade defects based on signals measured by various sensors including the vibration sensors, acoustic emission (AE) sensors, strain sensors, and ultrasonic sensors has been vigorously studied [2]. In studies of the WT blade defect diagnosis, AE sensors were most frequently utilized and a number of statistical methods were applied to analyze AE signals. Wei *et al.* [8] assessed damage severities as well as failure modes and locations of WT blades by comparing features of AE signals with the blade mechanical properties. Anastassopoulos *et al.* [9] utilized pattern recognition tools to construct the evaluation criteria of the integrity of blades based on AE signals so that alarms of impending failures could be issued. Menon *et al.* [10] introduced the wavelet-based AE analysis method with adaptive thresholds to evaluate WT blade conditions. Munoz *et al.* [11] applied a graphical method to locate the AE source for better detecting the blade damage. Alternative to AE sensors, the damage detection of WT blades based on ultrasonic techniques is also widely discussed [3]. Raisutis *et al.* [12] applied the ultrasonic air-coupled technique with guide waves to investigate defects in WT blades. Jasinien *et al.* [13] proposed a novel inspection method combining contact pulse-echo and immersion techniques for identifying the shape and size of blade defects. Recently, advanced sensors including macro-fiber composite (MFC) sensors [14], Fiber Bragg grating (FBG) sensors [15], and scanning laser Doppler vibrometer (SLDV) sensors [16] have also been applied to study the blade damage detection. Studies of the WT blade defect detection [8]–[16] reported powerful approaches for identifying defected WT blades. However, details of defects on WT blades still require further vision inspections.

Deploying unmanned aerial vehicles (UAVs) to inspect the surface condition of WT components, especially WT blades, has been recently tried in commercial wind farms to facilitate

the wind farm O&M efficiency [17]. UAVs equipped with digital cameras are remotely controlled to screen the surface of WT blades freely and captured images/videos can be transmitted back wirelessly. Main advantages of applying UAVs to inspect WT blades include: 1) A more frequent and periodical visit of WT blades is achievable; 2) Information of WT blade conditions is recorded as images/video clips which can be repeatedly and visually presented; 3) The risk of human injuries during the WT inspection process is dramatically reduced. The UAVs facilitate the process of collecting the information of WT blade conditions; however, analyses of collected images are typically conducted through visual inspections at the present stage. It is meaningful to build up an artificial intelligence (AI) for automatically processing UAV-taken images and generating analytical results so that labor costs can be saved and human errors can be eliminated.

Image processing approaches have been applied to the detection of cracks on public infrastructures. Tong *et al.* [18] identified the shape of the crack on the concrete bridge bottom based on binarized images. Hutchinson *et al.* [19] applied the Wavelet transformation and Bayesian decision approach to detect the crack on the concrete structure. Yamaguchi *et al.* [20] proposed a percolation-based image processing method to perform an efficient crack detection on the concrete surface. Methods reported in [18]–[20] were capable of identifying the crack from images with simple noises. However, as WT blade images taken by UAVs convey more complicated information, such as the surrounding environment, their analytics are more challenging. Powerful approaches for efficiently processing UAV-taken WT blade images and accurately identifying the crack information need to be developed.

This research presents a pioneer study of rapidly detecting WT blade cracks through analyzing images taken by UAVs. A data-driven framework for processing WT blade images and extracting details of blade cracks is developed. Haar-like features [20] are applied to depict WT blade cracks. Firstly, two feature sets, the original Haar-like feature set [21] and the extended Haar-like feature set [22], are compared in detecting blade cracks. A more effective Haar-like feature set is next selected. Based on the selected feature set, a new cascading classifier, which employs multiple base models including the LogitBoost [23], Decision Trees (DT) [24], and Support Vector Machines (SVM) [25] to develop stage classifiers, is proposed by extending a cascading classifier only developed by the LogitBoost [23]. The extended cascading classifier is compared with the LogitBoost cascading classifier to demonstrate its advantages. Based on the trained cascading classifier, a sliding window method is applied to detect cracks in an image. By iteratively sliding a scalable window over the UAV-taken image and applying the cascading classifier, regions containing cracks are detected and located. UAV-taken images collected from a commercial wind farm in China are utilized to validate the effectiveness of the proposed data-driven crack detection framework.

The remaining parts of this paper are organized as follows: In Section II, the development of the data-driven detection framework including the Haar-like feature set and procedures of developing cascading classifiers is introduced. Meanwhile, metrics for assessing the performance of the crack detection are defined. In Section III, a comprehensive real case study is conducted and computational results are analyzed. A conclusion of this study is presented in Section IV.

II. The Detection Framework

The image-based WT blade crack detection framework is composed of two phases, the data-driven framework development, and the data-driven framework deployment. A schematic diagram describing the procedure of developing the proposed crack detection framework is illustrated in Fig. 1.

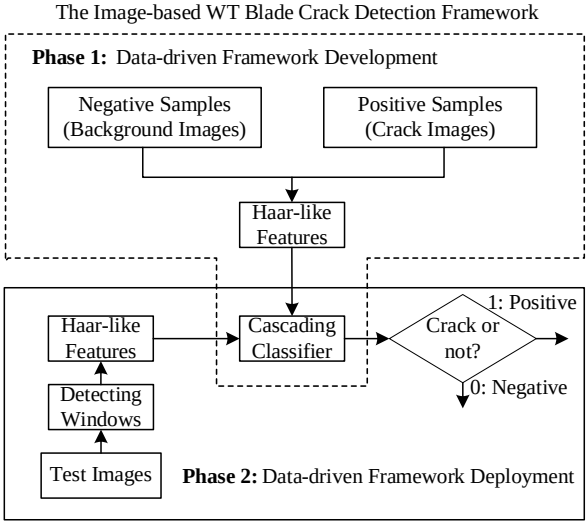


Fig. 1. The schematic diagram of the image-based WT blade crack detection framework

In Phase 1 of the schematic diagram, both original and extended Haar-like features are firstly compared to determine a more suitable set of Haar-like features for depicting blade cracks. Next, cascading classifiers are developed based on the selected feature set to detect blade cracks. Detailed procedures of constructing the data-driven crack detection framework are summarized as follows:

Step 1 Constructions of training samples
- Step 1.1 Collect WT blade images taken by UAVs
- Step 1.2 Manually extract regions containing cracks from collected images as positive samples for training; select a set of images without blade cracks while containing all possible backgrounds as negative samples for training

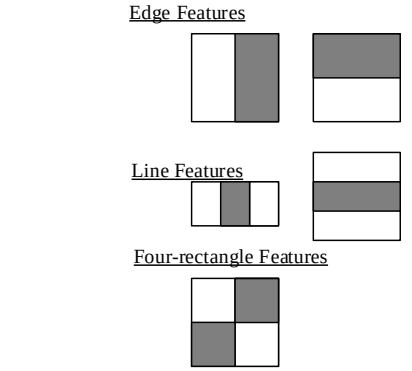
Step 2 Computation of Haar-like features
- Step 2.1 Transform training samples into grayscale images
- Step 2.2 Compute values of Haar-like features based on the original and the extended Haar-like feature sets

Step 3 Development of the cascading classifier
- Step 3.1 Set a number of stages for the cascading classifier
- Step 3.2 Train cascading classifiers based on computed two sets of Haar-like features respectively. To speed up the training process, a higher false-positive detection rate is set at each stage to ensure a 100% true-positive detection rate

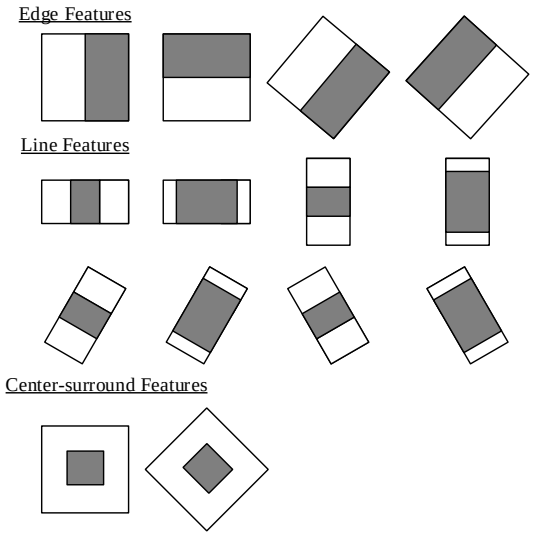
Step 4 Comparison of Haar-like features

Step 4.1 Classify the training images by trained classifiers
Step 4.2 Compare computational results and select a better feature set
Step 4.3 Apply training algorithms of cascading classifiers to develop crack detection models based on the selected feature set

In Phase 2, testing images are iteratively screened by a scalable window, which covers a sub-region of the image, and developed crack detection models (cascading classifiers) are deployed to determine sub-regions containing cracks.



(a) The original Haar-like features



(b) The extended Haar-like features

Fig. 2. Two Haar-like feature sets

A. Haar-like Features

Original Haar-like features, which were close to coefficients in the Haar wavelet transformation, were introduced in the Viola-Jones framework [20]. There are three advantages of using Haar-like features compared with methods based on pixel values. Firstly, the Haar-like features are able to encode *ad hoc* domain knowledge which is difficult to describe via a limited quantity of training samples. Compared with pixel values, the Haar-like features can reduce/increase the in-class/out-of-class variability to benefit the classification [22]. Next, detecting objects through Haar-like features is much faster than the pixel-based system. Moreover, since Haar-like features are computed to discover the difference between white and black rectangles, they are more robust to noises and lighting variations. To improve the performance of the original feature set, an extended feature set was proposed by Lienhart and Maydt [22]. Fig. 2 illustrates the original and the extended Haar-like feature set.

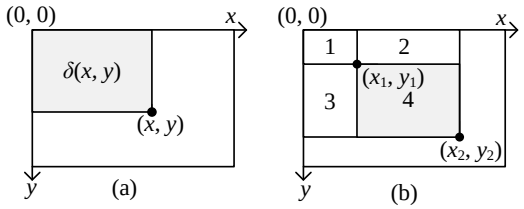


Fig. 3. The description of integral images

Based on grayscale images, the value of a single Haar-like feature, which describes the difference between pixels contained in white and gray rectangles, is computed according to (1).

$$h = \sum_{(x,'\,y')\in \text{grey region}} i(x', y') - \sum_{(x,'\,y')\in \text{white region}} i(x', y') \qquad (1)$$

where h is the value of a Haar-like feature and $i(x', y')$ is the pixel value at the location described as a coordinate, (x', y'), in an image. In real applications, all possible sizes and locations of each feature are computed. To facilitate the computation, an integral image, a rectangular region determined by an origin, (0, 0), and a coordinate, (x, y), as shown in Fig. 3(a), is firstly constructed. The value of this integral image, $\delta(x, y)$, is next computed based on (2).

$$\delta(x, y) = \sum_{x'\le x, y'\le y} i(x', y') \qquad (2)$$

Based on the integral image, Haar-like features can be promptly computed and the sum of pixel values contained in a specific rectangular region, $\delta((x_1, y_1), (x_2, y_2))$, defined by two coordinates (x_1, y_1) and (x_2, y_2) can be determined based on simple addition and subtraction operations of four integral values, such as Region 4 in Fig. 3(b). Its sum of pixel values is computed by (3).

$$\begin{aligned}\delta((x_{1,}y_1),(x_{2,}y_2)) &= \sum_{x_1\le x'\le x_2, y_1\le y'\le y_2} i(x', y') \\ &= \sum_{x'\le x_2, y'\le y_2} i(x', y') - \sum_{x'\le x_2, y'\le y_1} i(x', y') \\ &\quad - \sum_{x'\le x_1, y'\le y_2} i(x', y') + \sum_{x'\le x_1, y'\le y_1} i(x', y') \\ &= \delta(x_{2,}y_2) - \delta(x_{2,}y_1) - \delta(x_{1,}y_2) + \delta(x_{1,}y_1)\end{aligned} \qquad (3)$$

Define $S(x, y)$ as the cumulative row sum of integral values and a recursive method for computing the integral image is provided in (4).

$$\begin{aligned} S(x, y) &= S(x, y-1) + i(x, y) \\ \delta(x, y) &= \delta(x-1, y) + S(x, y)\end{aligned} \qquad (4)$$

Since x and y are both natural numbers, an initialization, $S(x, -1) = \delta(-1, y) = 0$, is applied and –1 indicates that the pixel is outside the image.

B. The Cascading Classifier

Since a large number of Haar-like features can be generated for a small image, it is computationally expensive to build a classifier based on all features. For example, over 100,000 Haar-like features can be generated for a 24×24 image [20]. To realize a prompt detection, a cascading classifier which is composed of classifiers over multiple stages is developed to identify blade cracks based on Haar-like features. The decision process of the cascading classifier is illustrated in Fig. 4. By sliding the detection window horizontally row-by-row on the target image (see Fig. 5), each window covers a specific region of the image and the cascading classifier employs classifiers of multiple stages to sequentially evaluate windows of the image. At the first stage, the classifier is trained only based on a small set of Haar-like features to achieve a pre-defined false alarm rate. The subsequent classifiers are enhanced via considering more features stage-wisely. Only if former stage classifiers determine a window as a region containing cracks, classifiers of subsequent stages will be activated. Since most windows of an image are negative (most regions do not contain cracks), the chance that a possibly positive window containing cracks can pass through all classification stages is low. Therefore, the cascading classifier rapidly filters out negative windows and focuses on analyzing possibly positive windows so that the computational time is significantly reduced.

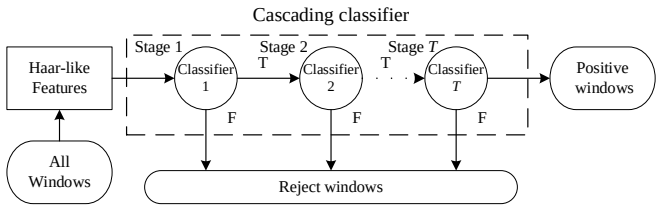


Fig. 4. The cascading classifier

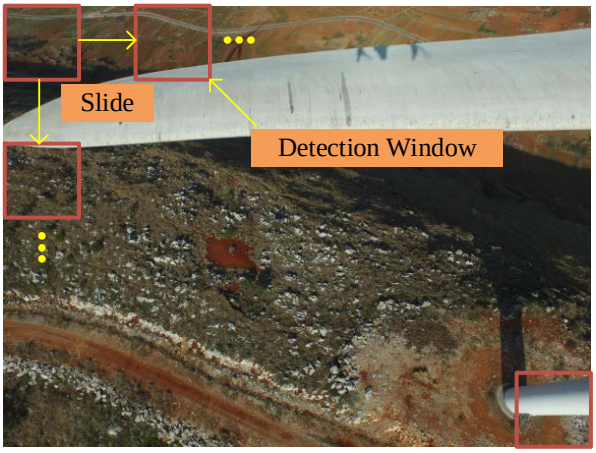


Fig. 5. A schematic diagram of the sliding window method. A window scans the whole image with a pre-defined sliding scale and the cascading classifier is applied to determine labels of its every move, positive and negative.

The LogitBoost Cascading Classifier

A simple LogitBoost cascading classifier is employed here to firstly introduce the principle of the cascading classifier. The LogitBoost cascading classifier only applies the LogitBoost to build stage classifiers by integrating weak linear classifiers. The training objective is to minimize the false-positive rate conditioned on a pre-defined value. Given h_j as one of the Haar-like feature of a window image $\mathbf{x}$, the one-feature weak classifier, c_j, is a linear classifier defined in (5).

$$c_j(\mathbf{x}) = \begin{cases} 1 \text{ if } p_j h_j < p_j \theta_j, j \in [1, N_h] \\ 0 \text{ otherwise} \end{cases} \tag{5}$$

where θ_j is a classifier parameter, N_h is the number of Haar-like features of $\mathbf{x}$, and p_j indicates the direction of the inequality sign. The LogitBoost is applied to integrate one-feature weak classifiers developed based on different Haar-like features to achieve the training objective. In the LogitBoost, the binomial log-likelihood loss function as shown in (6) is considered.

$$\mathcal{L}(\mathbf{x}, l) = \log(1 + e^{-l \cdot F(\mathbf{x})}) \tag{6}$$

In (6), the F is a function composed of one-feature weak classifiers. Two classes of images, the positive and negative, are defined as $l \in \{0,1\}$. This loss function varies linearly with the classification error and is less sensitive to noises as well as outliers [22]. To optimize the loss function, the LogitBoost builds up a strong classifier $C(\mathbf{x})$ shown in (7) by iteratively adding weak classifiers.

$$C(\mathbf{x}) = \text{sign}[F(\mathbf{x})] = \begin{cases} 1 \text{ if } F(\mathbf{x}) > 0 \\ 0 \text{ if } F(\mathbf{x}) < 0 \end{cases} \tag{7}$$

In (7), $C(\mathbf{x})$=1 indicates that $\mathbf{x}$ belongs to the positive class and $C(\mathbf{x})$=0 means that $\mathbf{x}$ belongs to the negative class. F is learned through following procedures.

Given N training samples, the training dataset is described in (8).

$$\mathcal{D} = \{(\mathbf{x}_1, l_1), (\mathbf{x}_2, l_2), \ldots, (\mathbf{x}_N, l_N)\} \tag{8}$$

Given iteration steps, t = 1, 2,…, I, the strong classifier is updated according to (9) and (10).

$$F(\mathbf{x}) = F(\mathbf{x}) + \frac{1}{2} f_t(\mathbf{x}) \tag{9}$$

$$f_t(\mathbf{x}) = \arg\min_{c_j} \frac{1}{2} \sum_{i=1}^{N} w_i [z_i - c_j(\mathbf{x}_i)]^2, j = 1, 2, \ldots, N_h \tag{10}$$

In (10), the weight, w_i, and the working response, z_i, are computed by (11) and (12).

$$w_i = p(\mathbf{x}_i)[1 - p(\mathbf{x}_i)] \tag{11}$$

$$z_i = \frac{y_i - p(\mathbf{x}_i)}{p(\mathbf{x}_i)[1 - p(\mathbf{x}_i)]} \tag{12}$$

The $p(\mathbf{x})$ is updated per iteration with the latest $F(\mathbf{x})$ according to (13).

$$p(\mathbf{x}) = \frac{e^{F(\mathbf{x})}}{e^{F(\mathbf{x})} + e^{-F(\mathbf{x})}} \tag{13}$$

The initial $F(\mathbf{x})$ is set to 0, the initial probability is set as $p(\mathbf{x})$ = $p(y=1|\mathbf{x})$ = q/N, and the initial weights are set as $w_i = \frac{1}{2m}, \frac{1}{2q}$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2017.2682037, IEEE Transactions on Industrial Electronics

for l_i = 0,1 respectively, where m and q are the number of negatives and positives.

After executing required iterations, $F(\mathbf{x})$ becomes

$$F(\mathbf{x}) = \frac{1}{2}[0 + f_1(\mathbf{x}) + f_2(\mathbf{x}) + \ldots + f_I(\mathbf{x})] \quad (14)$$

The iterative procedure of developing a cascading classifier based on stage classifiers trained by the LogitBoost is depicted in the **Algorithm 1**.

Algorithm 1: LogitBoost cascading classifier

Input: Training dataset: $\mathcal{D}$;
Number of stages: T;
Target false-positive rate: r_{target};
Output: Stage Classifier: $C(\mathbf{x})$
Let $k = 0$;
while $k < T$ **do**
  Let $k = k + 1$;
  Let $r = 1$;
  while $r > r_{target}$ **do**
    Train Classifier $C(\mathbf{x})$;
    Compute r;
  end
  Save $C(\mathbf{x})$ as classifier for Stage k;
end

In this study, the r_{target} is set to 0.3 and T is set to 25 in training a LogitBoost Cascading classifier.

Algorithm 2: Extended cascading classifier

Input: Training dataset: $\mathcal{D}$;
Number of stages: T;
Target false-positive rate: r_{target};
Maximum number of features at each stage: n_{max};
Output: Stage Classifier: $C(\mathbf{x})$;
Let $k = 0$;
while $k < T$ **do**
  Let $k = k + 1$;
  Let $r = 1$;
  Let $n_k = 0$;
  while $r > r_{target}$ *and* $n_k < n_{max}$ **do**
    Train classifier $C(\mathbf{x})$ with LogitBoost;
    Compute false-posive rate: r;
    Let $n_k \leftarrow$ the number of used features;
  end
  if $n_k = n_{max}$ **then**
    Train classifier $C(\mathbf{x})$ with DT algorithm;
    Compute r;
    if $r > r_{target}$ **then**
      Train classifier $C(\mathbf{x})$ with SVM algorithm;
    end
  end
  Save $C(\mathbf{x})$ as classifier for Stage k;
end

The Extended Cascading Classifier

In training cascading classifiers, negative samples become fewer and less different from positive samples at latter stages. To better separate negative and positive samples, more features need to be considered by stage classifiers. In addition, limited training samples and an enlarged number of features may lead to over-fitting at latter stages. To better tackle this challenge, an extended cascading classifier utilizing multiple base models, the LogitBoost, DT and SVM, to develop stage classifiers is proposed. In such classifier, the LogitBoost is employed in priority to develop stage classifiers; however, it will be substituted by DT or SVM if the pre-defined number of features is reached. Since DT and SVM are developed based on features selected by the LogitBoost, it is fast to train these two models. Meanwhile, as DT is relatively simpler than SVM, the substitution is performed sequentially from DT to SVM until a pre-defined r_{target} is achieved. Under such setting, the extended cascading classifier will maintain a high efficiency by preferring simpler models in stacking stage classifiers.

In SVM, the Gaussian kernel described in (15) is applied.

$$K(\mathbf{x}, \mathbf{x}') = \exp(\gamma \left\| \mathbf{x} - \mathbf{x}' \right\|^2) \quad (15)$$

In (15), γ is a free parameter. In this study, γ is set to the inverse of the number of features according to [26]. Besides γ, the capacity factor is set to 10 in training the SVM model. In DT model, the impurity for early stopping in the tree growth is set to 10^{-7} according to [24].

The procedure of training the extended cascading classifier is described in **Algorithm 2**. In **Algorithm 2**, the n_{max} is set to 30, which ensures r_{target} is obtained at stages.

C. Detection Process

In the detection process, testing images are scanned by the detection window and all windows are classified by the developed cascading classifiers. The detection window moves with a pre-defined number of pixels, Δ, incrementally. The Δ is set to 10 for movements in both horizontal and vertical directions. Since cracks have different sizes, it is necessary to scale the detection window to cover them. Compared with directly scaling testing images, the proposed process is more efficient because features in the classifier are evaluated with the same cost at any scale [20]. A scale parameter, s = 1.25, is employed to scale the detection window after each scan and the movement is also scaled by s.

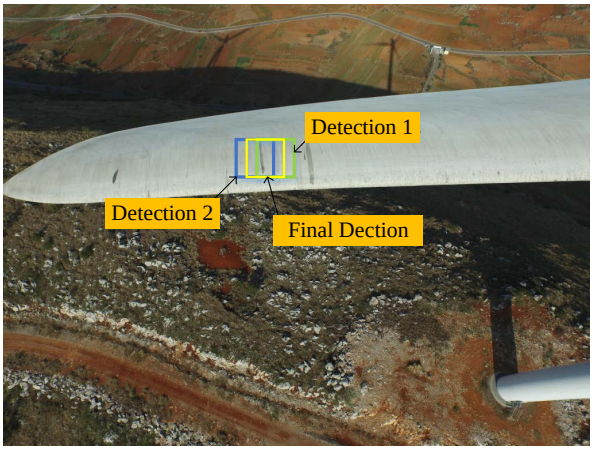


Fig. 6. An illustrative example of the post-processing of detection results.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2017.2682037, IEEE Transactions on Industrial Electronics

As the same crack can be detected by multiple windows, a post-processing is implemented to merge repeated detections. An illustrative example of merging two overlapping detections is demonstrated in Fig. 6. The merge process is composed of following two major steps: 1) Partition a set of detections into disjoint subsets. Thus, two detections are included into the same subset if there exists an overlap in their bounding regions. 2) The corner of each partition is set as the average of corners of all detections within the same partition.

D. Performance Assessment

To assess the effectiveness of the proposed framework, three metrics, the detection rate (*DR*), false alarm rate (*FR*), and accuracy (*AC*), defined in (16) - (18) are applied.

$$DR = \frac{TP}{TP + FN} \tag{16}$$

$$FR = \frac{FP}{TN + FP} \tag{17}$$

$$AC = \frac{TP + TN}{TP + FN + TN + FP} \tag{18}$$

In (16)-(18), *TP* is the number of true positives, *FN* is the number of false negatives, *FP* is the number of false positives, and *TN* is the number of true negatives.

III. Case Studies

To evaluate the effectiveness of the proposed data-driven framework, UAV-taken images provided by a commercial wind farm in China are utilized. In this wind farm, each wind turbine has three blades. The length and the maximal width of each blade are 37.5m and 3.2m respectively. Based on UAV-taken images, cascading classifiers are trained by using both original and extended Haar-like features. Their performance is compared and a more effective feature set is selected. The extended cascading classifier is next validated based on the selected set of Haar-like features.

A. Image Description and Pre-Processing

The UAV for taking WT blade images is manufactured by DJI Co., Ltd. and its model is DJI FC-350. The commercial wind farm containing 33 WTs located in a mountainous region of China is considered in this study. Images including WT blade images, WT images, and wind farm images are taken from various angles and positions in low wind speed days without shutting down wind turbines. Illustrative examples of UAV-taken images are provided in Fig. 7.

It is observable that various backgrounds and lighting conditions are included in UAV-taken images. To minimize impacts of lighting conditions, an image pre-processing [21], [22] is conducted. The following normalization (19) is firstly applied.

$$i'(x, y) = \frac{i(x, y) - \mu}{\sigma} \tag{19}$$

In (19), μ and σ are the mean and standard deviation of pixel values in the image. By using the integral image, μ and σ can be fast computed.

Fig. 7. UAV-taken image examples

Fig. 8 A UAV-taken blade image with an enlarged crack

After the normalization, WT blade crack regions are manually cropped from UAV-taken images and defined as positive samples in training after confirming with wind farm field experts. Besides, negative samples are generated by extracting regions describing the surrounding environment including the ground, sky, and WT blades without cracks. Totally 50 positive samples and 200 negative samples are generated from raw images. In this study, the length and width of the smallest crack among 50 positive samples are 6.54 cm and 1.55 cm separately. Since cracks can display differently according to angles and distances of taking images, resolutions are considered as criteria of detectable cracks. Based on the set of cropped cracks, cracks with length > 40 pixels and width > 10 pixels in a raw image with a size = 4000×3000 can be effectively detected in this study. In Fig. 8, a region containing

a crack in a UAV-taken image is enlarged to offer an illustrative example of the crack failure.

To enrich positive samples and improve the training of the cascading classifier, a denoising process is performed to generate denoised samples. Both of original and denoised samples are considered as training samples. Applied denoising filters include the bilateral filter, median filter, and Laplacian of Gaussian operator [27].

A bilateral filter is a non-linear, edge-preserving, and noise-reducing smoothing filter for images. Each pixel value in an image is replaced by a weighted average of values from nearby pixels. This preserves sharp edges by looping through each pixel and adjusting weights to the adjacent pixels accordingly. The filter is defined as

$$i^b(x,y)=\frac{1}{W_p}\sum_{(x_i,y_i)\in\Omega} i(x_i,y_i)r(\|i(x_i,y_i)-i(x,y)\|)g_s(\|(x_i,y_i)-(x,y)\|) \quad (20)$$

$$W_p=\sum_{(x_i,y_i)\in\Omega} r(\|i(x_i,y_i)-i(x,y)\|)g_s(\|(x_i,y_i)-(x,y)\|) \quad (21)$$

where i^b is a pixel value of the filtered image, Ω is the window centered in (x, y), r is the range kernel for smoothing differences in intensities, and g_s is the spatial kernel for smoothing differences in coordinates.

The Median filter is a non-linear filter and the central pixel value is replaced with this median value of all pixels under same kernel windows. This filter is effective in removing the salt-and-pepper noise. The smoothed pixel value i^m is obtained according to (22)

$$i^m(x,y)=median[i^b(x_i,y_i)],(x_i,y_i)\in R[(x_i,y_i)] \quad (22)$$

where R is a region, a 5×5 rectangle, around the pixel (x, y) in this study.

As crack regions are darker than surrounding areas, a Laplacian of Gaussian (LoG) operator is employed to further distinguish crack regions from the image. The LoG operator is presented in (23).

$$i^{LoG}(x,y)=-\frac{1}{\pi\sigma^4}[1-\frac{x^2+y^2}{2\sigma^2}]e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (23)$$

Since the size of raw images is large, a resizing process is applied to improve the detection efficiency. A sensitivity analysis is conducted to determine the most suitable scale factor in resizing from a set {1/2,1/3,…,1/10}. A widely considered detection window size, 24×24 [28], is applied. An ad-hoc training set is developed by incorporating 20 positive and 50 negative samples randomly selected from the training image set and scaled to 24×24 to facilitate the sensitivity analysis. Based on the ad-hoc training set, a LogitBoost cascading classifier is trained with the original Haar-like feature set. Next, the developed classifier is applied to images resized according to scale factors. Detection results show that a high detection performance (19 cracks detected) is achieved with images whose sizes are bigger than 1000×750 and a degradation of the detection performance is observed with images resized by scale factors smaller than 1/4. Therefore, all raw images are resized to 1000×750 to expedite the detection process. To further tune detection window sizes, three window sizes, 20×20, 24×24, and 30×30, are utilized. Detection results based on three windows sizes are provided in Table I.

TABLE I
DETECTION RESULTS BASED ON THREE WINDOW SIZES

Window	Detected Cracks	False Alarms
20×20	18	2
24×24	19	3
30×30	19	2

According to Table I, the detection window size is set to 30×30 as it provides better detection performance. Based on results of sensitivity analyses, the 50 original and 20 filtered positive samples as well as 200 negative samples are resized into 30×30 to develop the training dataset for developing the cascading classifier.

B. Comparative Analysis of Haar-like Feature Sets

The LogitBoost cascading classifier is developed based on the original and extended Haar-like feature sets applied into generated training samples. The training is performed on a computer with Intel Core i5-4570 @ 3.2GHz CPU and 8GB memory. The implementation of training is based on Windows 7, C++ and OpenCV package. Tables II and III list the performance of two classifiers.

TABLE II
TRAINING RESULTS OF LOGITBOOST CASCADING CLASSIFIERS BASED ON TWO HAAR-LIKE FEATURE SETS

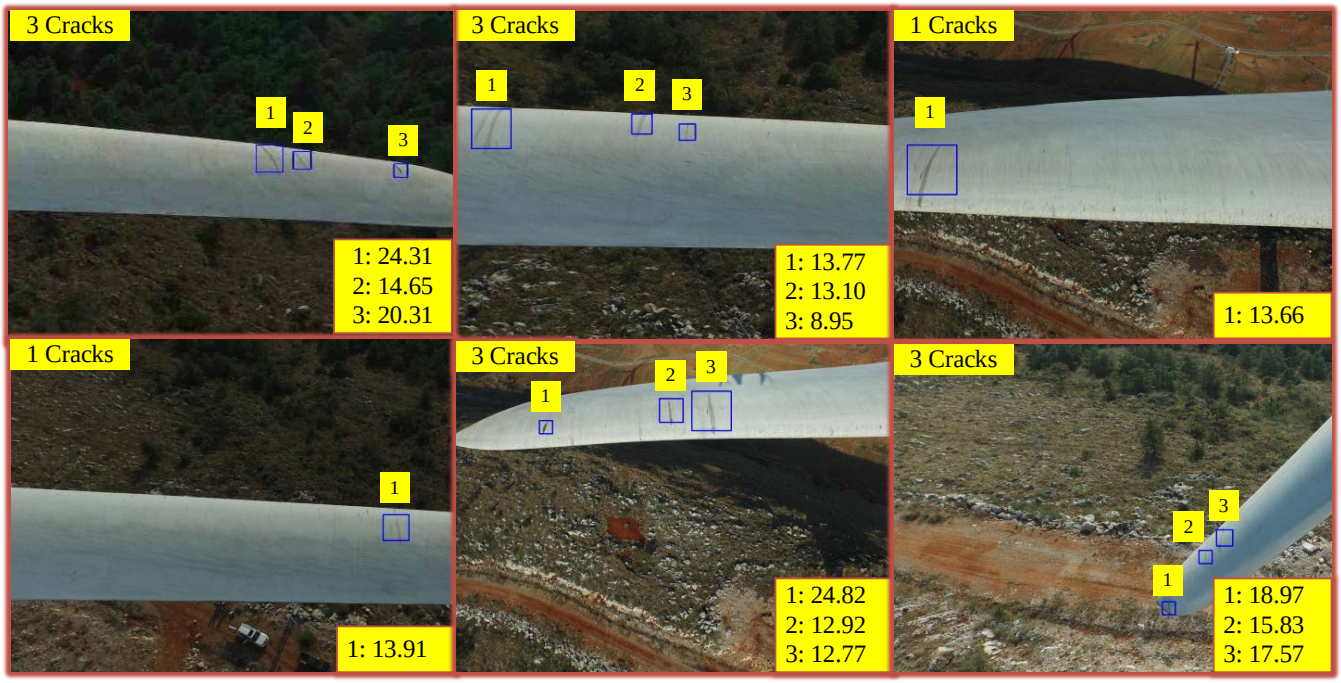
	Detected			
Actual	Original Haar-like Features		Extended Haar-like Features	
	Positive	Negative	Positive	Negative
Positive	56	14	68	2
Negative	10	190	2	198

TABLE III
COMPARISON OF LOGITBOOST CASCADING CLASSIFIERS BASED ON TWO HAAR-LIKE FEATURE SETS

Feature Set	DR	FR	No. of Features	Training Time
Original	0.800	0.050	1112	1531s
Extended	0.971	0.010	857	1573s

According to Tables II, most negative samples are rejected by two LogitBoost cascading classifiers. The returned accuracies of two classifiers are 0.911 and 0.985 respectively. The *DR* and *FR* associated with the utilized number of Haar-like features and training time of two classifiers are computed and listed in Table III. It is observable that the cascading classifier based on the extended Haar-like features has a better performance than that using the original Haar-like features in terms of *DR*, *FR*, and the number of features used. Although the original feature set contains fewer feature types, a classifier needs a larger number of features derived from them to achieve a required *FR* at each stage. Meanwhile, the training time differs slightly between two classifiers. Thus, the result supports that it is more effective and efficient to select the extended Haar-like feature set in the development of the data-driven framework.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2017.2682037, IEEE Transactions on Industrial Electronics



(a)

(b)

Fig. 9. Detection results of UAV-taken blade images. (a) Blade images with cracks (b) Blade images without cracks

C. Comparison of Two Cascading Classifiers

Based on the extended Haar-like feature set, the extended cascading classifier is benchmarked with the LogitBoost cascading classifier to demonstrate its advantage. The performance of two classifiers is listed in Tables IV and V.

TABLE IV
TRAINING RESULTS OF THE EXTENDED CASCADING CLASSIFIER

Actual \ Detected	Positive	Negative
Positive	69	1
Negative	1	199

TABLE V
COMPARISON OF TWO CASCADING CLASSIFIERS

Classifier	DR	FR	No. of Features	Training Time
LogitBoost	0.971	0.010	857	1573s
Extended	0.985	0.005	417	1346s

As shown in Tables IV and V, a further improvement in the crack detection is achieved with the extended cascading classifier. Moreover, the number of features utilized for its training is reduced compared with the LogitBoost cascading classifier. Therefore, less training time is required by the extended cascading classifier and it is more efficient in the crack detection.

D. Validation Results

To validate the effectiveness of the proposed framework, more positive images are produced by embedding randomly rotated crack samples into random backgrounds. Simultaneously, more negatives images are obtained through the Internet. Finally, 200 positive images and 570 negative images are utilized to validate the proposed WT blade crack detection framework. Examples of images utilized in the validation and detection results are presented in Fig. 9.

Cracks in Fig. 9 are all correctly detected and their locations are clearly marked with blue rectangles. The number of cracks

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2017.2682037, IEEE Transactions on Industrial Electronics

in each image is counted and shown at the top-left corner of each image. Meanwhile, to describe the degree of cracks, the standard deviation of grayscale values (SDGVs) in each crack window is computed and listed in each image. To obtain a baseline value representing SDGVs of regions having cracks, the average of SDGVs of all positive samples, 13.37, is applied. Meanwhile, 100 sub-regions with the same size, 30×30, are randomly cropped from blade regions without cracks in images. An average of SDGVs in cropped images, 4.38, is treated as a baseline value representing SDGVs of negative samples.

As shown in Fig. 9, the proposed detection framework can detect cracks as well as provide the location, the degree, and the number of cracks based on UAV-taken images.

Tables VI and VII summarize the performance of the extended cascading classifier and the LogitBoost cascading classifier over 770 samples.

TABLE VI
CONFUSION MATRIX OF THE EXTENDED CLASSIFIER

Actual \ Detected	Positive	Negative
Positive	194	6
Negative	5	565

TABLE VII
PERFORMANCE OF THE DATA-DRIVEN CRACK DETECTION FRAMEWORK

Classifier	DR	FR	Detection Time (per image)
LogitBoost	0.950	0.012	0.097s
Extended	0.970	0.008	0.083s

The confusion matrix of the proposed framework with the extended cascading classifier is presented in Table VI and its accuracy is 0.986. Since the objective of training the cascading classifier is to minimize the *FR* at each stage, results in Table VII are consistent with the objective and only a few negative samples are classified as positive. To compare the performance of the extended cascading classifier and LogitBoost cascading classifier based on test samples, the *DR* and *FR* are computed and listed in Table VII. Table VII shows that the extended cascading classifier outperforms the LogitBoost cascading classifier in terms of *DR*, *FR*, and processing time. Moreover, both classifiers have a stable performance on both training and testing. Since random noises are introduced into test samples and prior knowledge of these unseen noises is unavailable, results confirm that Haar-like features are robust to these unseen noises.

Meanwhile, the detection for an image can be completed with less than 0.09s using the extended cascading classifier. It indicates a great potential of the real-time video processing as a regular camera takes videos at a speed of 15 or 30 frames/s. To further examine the real-time performance of the proposed framework, a video clip is created by continuously rotating and shifting an image for a period of time. Based on this video, the proposed framework is applied to detect cracks in each frame of the video. Fig. 10 illustrates the detection results of three selected frames in a video.

According to Fig. 10, the proposed framework is able to detect cracks no matter the crack rotates or flips. These characteristics of the proposed framework support its capability of the real-time detection. However, the proposed framework needs to be tested on real videos taken by UAVs in the future.

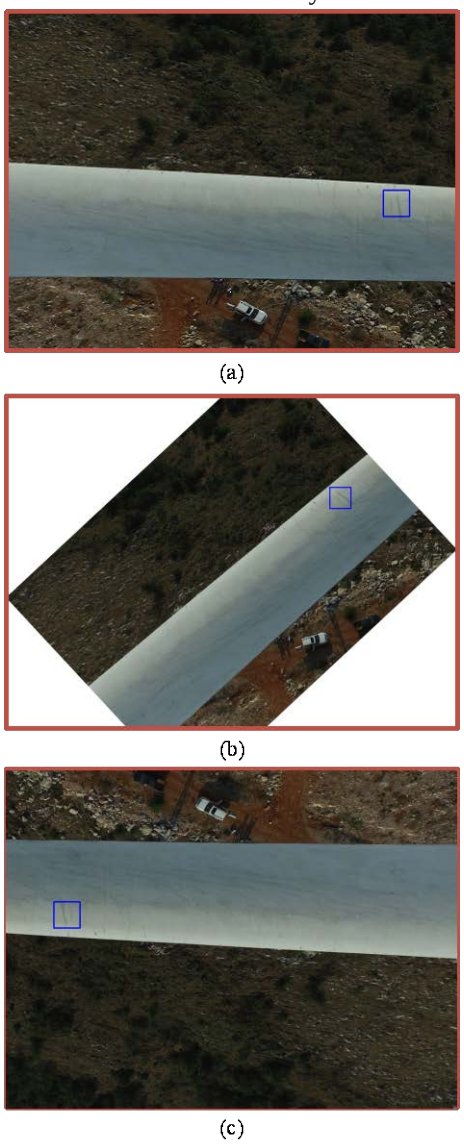


Fig. 10. Detection results of three video frames. (a) the original image (b) the rotated image (c) the flipped image

IV. CONCLUSION

A data-driven framework for automatically detecting cracks on the WT blade surface and depicting their locations based on UAV-taken images was developed. The framework included two phases, its development and deployment. The first phase developed a cascading classifier based on Haar-like features to identify images containing blade cracks. Two sets of Haar-like features, the original and extended Haar-like features, were compared in the development of the cascading classifier. An extended cascading classifier was proposed and compared with the LogitBoost cascading classifier. In the second phase, a mechanism of applying scalable windows to screen testing images and locate regions of WT blades containing cracks was presented. The proposed framework was validated with UAV-taken images collected from a commercial wind farm in China.

In the comparative analysis of two Haar-like feature sets, the classifier developed based on the extended Haar-like feature set offered the better performance than that provided by the classifier developed based on the original Haar-like feature set. Thus, the set of extended Haar-like features was considered as more suitable in developing the cascading classifier for identifying blade cracks. Meanwhile, the extended classifier offered the better performance than the LogitBoost classifier in terms of *DR*, *FR*, and processing time.

To further prove the effectiveness of the data-driven framework, a set of WT blade images was artificially generated and incorporated into the set of UAV-taken images to form a validation dataset. A promising result was obtained through the proposed framework. Meanwhile, the detection process was completed rapidly and this advantage allowed the potential application of the proposed framework in detecting WT blade cracks from video streams.

ACKNOWLEDGMENT

We would like to acknowledge the contribution of Jia Xu, Shaowu Li, and Ruihua Liu from China Longyuan Power Group Ltd. in the provision of UAV-taken blade images in this research.

REFERENCES

[1] J. Ribrant and L. M. Bertling, "Survey of Failures in Wind Power Systems with Focus on Swedish Wind Power Plants During 1997 ndash;2005," *IEEE Trans. Energy Convers.*, vol. 22, no. 1, pp. 167–173, Mar. 2007.

[2] F. P. García Márquez, A. M. Tobias, J. M. Pinar Pérez, and M. Papaelias, "Condition monitoring of wind turbines: Techniques and methods," *Renew. Energy*, vol. 46, pp. 169–178, Oct. 2012.

[3] R. Raišutis, E. Jasiūnienė, R. Šliteris, and A. Vladišauskas, "The review of non-destructive testing techniques suitable for inspection of the wind turbine blades," *Ultragarsas Ultrasound*, vol. 63, no. 1, pp. 26–30, 2008.

[4] P. Tchakoua, R. Wamkeue, M. Ouhrouche, F. Slaoui-Hasnaoui, T. A. Tameghe, and G. Ekemb, "Wind Turbine Condition Monitoring: State-of-the-Art Review, New Trends, and Future Challenges," *Energies*, vol. 7, no. 4, pp. 2595–2630, Apr. 2014.

[5] W. Qiao and D. Lu, "A survey on wind turbine condition monitoring and fault Diagnosis—Part I: Components and subsystems," *IEEE Trans. Ind. Electron.*, vol. 62, no. 10, pp. 6536–6545, Oct. 2015.

[6] L. Wang, Z. Zhang, J. Xu, and R. Liu, "Wind Turbine Blade Breakage Monitoring with Deep Autoencoders," *IEEE Trans. Smart Grid*, vol. PP, no. 99, pp. 1–1, 2016.

[7] D. Li, S.-C. M. Ho, G. Song, L. Ren, and H. Li, "A review of damage detection methods for wind turbine blades," *Smart Mater. Struct.*, vol. 24, no. 3, p. 33001, 2015.

[8] J. Wei and J. McCarty, "Acoustic emission evaluation of composite wind turbine blades during fatigue testing," *NDT E Int.*, vol. 2, no. 30, p. 111, 1997.

[9] A. Anastassopoulos *et al.*, "Structural integrity evaluation of wind turbine blades using pattern recognition analysis on acoustic emission data.," *J. Acoust. Emiss.*, vol. 20, pp. 229–237, 2003.

[10] S. Menon, J. N. Schoess, R. Hamza, and D. Busch, "Wavelet-based acoustic emission detection method with adaptive thresholding," in *SPIE Proceedings*, 2000, vol. 3986, pp. 71–78.

[11] C. Q. Gómez Muñoz and F. P. García Márquez, "A New Fault Location Approach for Acoustic Emission Techniques in Wind Turbines," *Energies*, vol. 9, no. 1, p. 40, Jan. 2016.

[12] R. Raišutis, E. Jasiūnienė, and E. Žukauskas, "Ultrasonic NDT of wind turbine blades using guided waves," *Ultragarsas*, vol. 63, no. 1, pp. 7–11, 2008.

[13] E. Jasinien *et al.*, "NDT of wind turbine blades using adapted ultrasonic and radiographic techniques," *Insight - Non-Destr. Test. Cond. Monit.*, vol. 51, no. 9, pp. 477–483, Sep. 2009.

[14] C. Pitchford, B. L. Grisso, and D. J. Inman, "Impedance-based structural health monitoring of wind turbine blades," in *SPIE Proceedings*, 2007, p. 65321I.

[15] J.-K. Lee, J.-Y. Park, K.-Y. Oh, S.-H. Ju, and J.-S. Lee, "Transformation algorithm of wind turbine blade moment signals for blade condition monitoring," *Renew. Energy*, vol. 79, pp. 209–218, Jul. 2015.

[16] M. Ozbek, F. Meng, and D. J. Rixen, "Challenges in testing and monitoring the in-operation vibration characteristics of wind turbines," *Mech. Syst. Signal Process.*, vol. 41, no. 1–2, pp. 649–666, Dec. 2013.

[17] "Cyberhawk launches commercial ROAV wind turbine inspection service | Cyberhawk remote aerial inspections and surveys." [Online]. Available: http://www.thecyberhawk.com/2015/10/cyberhawk-launches-commercial-roav-wind-turbine-inspection-service/. [Accessed: 22-Nov-2016].

[18] X. Tong, J. Guo, Y. Ling, and Z. Yin, "A new image-based method for concrete bridge bottom crack detection," in *2011 International Conference on Image Analysis and Signal Processing*, 2011, pp. 568–571.

[19] T. C. Hutchinson and Z. Chen, "Improved Image Analysis for Evaluating Concrete Damage," *J. Comput. Civ. Eng.*, vol. 20, no. 3, pp. 210–216, 2006.

[20] T. Yamaguchi, S. Nakamura, and S. Hashimoto, "An efficient crack detection method using percolation-based image processing," in *2008 3rd IEEE Conference on Industrial Electronics and Applications*, 2008, pp. 1875–1880.

[21] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001, vol. 1, p. I-511-I-518 vol.1.

[22] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," in *Proceedings. International Conference on Image Processing*, 2002, vol. 1, p. I-900-I-903 vol.1.

[23] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *Ann. Stat.*, vol. 28, no. 2, pp. 337–407, Apr. 2000.

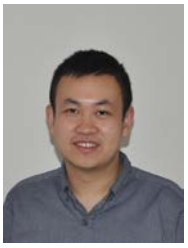
[24] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*, 1 edition. New York: Chapman and Hall/CRC, 1984.

[25] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297.

[26] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," *ACM Trans Intell Syst Technol*, vol. 2, no. 3, p. 27:1–27:27, May 2011.

[27] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*. Toronto: CL Engineering, 2007.

[28] R. Lienhart, A. Kuranov, and V. Pisarevsky, "Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection," in *Pattern Recognition*, Springer Berlin Heidelberg, 2003, pp. 297–304.



Long Wang (S'16) received his M.Sc. degree in Computer Science with distinction from University College London, London, U.K., in 2014. He is currently pursuing the Ph.D. degree in the Department of Systems Engineering and Engineering Management, City University of Hong Kong, Hong Kong.

His research interests include electricity price forecasting, wind turbine condition monitoring, computer vision, and parallel computing.

Zijun Zhang (M'12) received his Ph.D. and M.S. degrees in Industrial Engineering from the University of Iowa, Iowa City, IA, USA, in 2012 and 2009, respectively, and B.Eng. degree in Systems Engineering and Engineering Management from the Chinese University of Hong Kong, Hong Kong, China, in 2008.

Currently, he is an Assistant Professor in the Department of Systems Engineering and Engineering Management at the City University of Hong Kong, Hong Kong, China. His research focuses on data mining and computational intelligence with applications in wind energy, HVAC and wastewater processing domains.