Original Article

Journal of RAIL AND RAPID TRANSIT

Proc IMechE Part F:
J Rail and Rapid Transit
0(0) 1–14
© IMechE 2015
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0954409715588119
pif.sagepub.com

Online inspection system for the automatic detection of bolt defects on a freight train

Caiqin Li, Zhenzhong Wei and Jing Xing

Abstract

Inspecting the condition of the key components of freight trains is an important task in the rail industry. Bolts on the wheel bearings are key components of a bogie, and bolt defects, such as missing or broken bolts, can lead to serious accidents. To improve the traditional manual inspection procedure, which is both laborious and inflexible, a novel method of automatic image recognition for bolt defects is proposed in this paper. The main procedures are as follows. When a freight train drives through the inspection station, images of the train's wheels are captured by cameras installed alongside the track. Based on the local binary pattern descriptor, a support vector machine classifier is trained to distinguish between bolt and non-bolt images. The classifier is then combined with a rotate-and-slide window method to localize the three bolt regions in the wheel image. Specifically, a self-updating method is proposed in the training phase to automatically capture the various different situations experienced by bolts in real-life scenarios. After localization, we distinguish defective bolts from normal bolts based on whether there is a hexagonal shape in the bolt region. As demonstrated by real-life experiments, our proposed method can guarantee to find bolt defects and further work will be devoted to reducing the false alarm rate.

Keywords

Image recognition, freight train, defect detection, LBP descriptor, support vector machine

Date received: 28 September 2014; accepted: 23 April 2015

Introduction

The inspection of freight trains is a critical task that is performed to ensure the safety of railway traffic. In contrast with track inspection, it focuses on evaluating the real-time condition of train components, such as fastening bolts, angle cocks and dust collectors. Traditionally, the inspection of train components has been laborious and inflexible; this has resulted in the maximum speed of freight trains being limited. Therefore, it is of great interest to replace the manual inspection process by an automatic inspection system using advanced computer vision technologies.

In recent years, an inspecting system called the 'Trouble of Freight Car Detection System' (TFDS) has been widely utilized on Chinese railways. The TFDS allows the online inspection of a train's key components, thereby preventing dangerous situations from arising. The TFDS consists of two major parts: a dynamic image gathering and transmitting module, and an image recognition module.

As shown in Figure 1, five high-speed cameras are used to inspect different parts of the freight train. Two are installed on the side of the tracks and acquire the profile of the freight train. The others are installed along the centreline of the track and acquire images of the underside of the train. Along the track, magnets (C0–C10) are used to trigger cameras and capture dynamic images. With the train in motion, images of all carriages are obtained and transmitted to a remote monitoring server where they are used as input to the image recognition module that detects defects in the components of the train.

A total of nearly 3000 images are taken for each freight train, depending on the number of carriages attached to the locomotive. Each image has a size of 1024×1200 pixels. Figure 2 shows 59 images of a carriage captured by TFDS. The top three rows display images captured by the bottom cameras, and the last two rows display images captured by the trackside cameras.

Key Laboratory of Precision Opto-mechatronics Technology, Beihang University, People's Republic of China

Corresponding author:
Zhenzhong Wei, Key Laboratory of Precision Opto-mechatronics Technology, Beihang University, Beijing 100191, People's Republic of China.
Email: zhenzhongwei@buaa.edu.cn

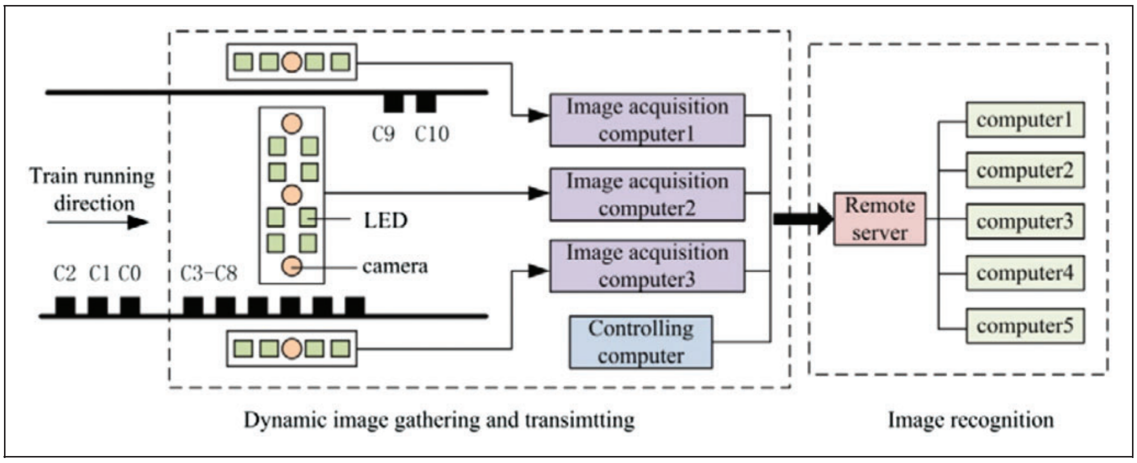

Figure 1. Sketch of the TFDS.

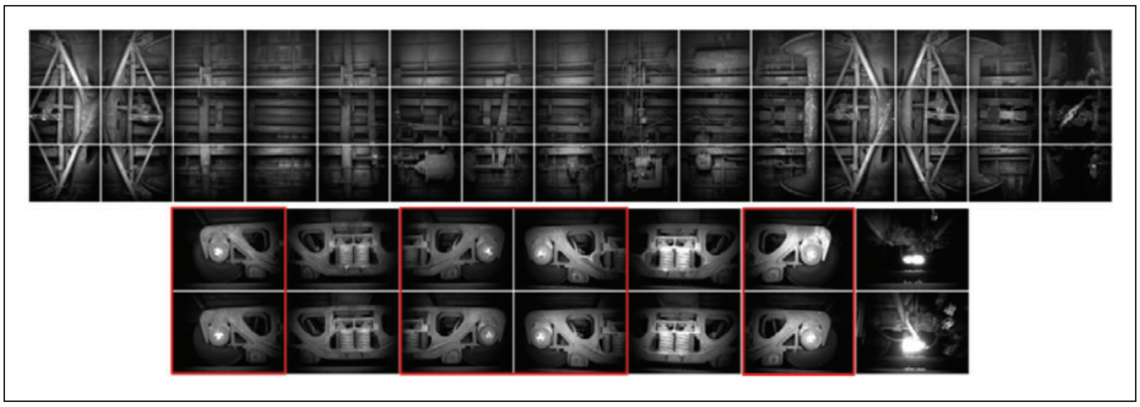

Figure 2. Fifty-nine images captured by TFDS.

As this paper is focussed on automatically detecting bolt defects, the images shown in the red boxes in Figure 2 are of key importance. Hexagonal-headed bolts play an important role in fastening the wheel bearing. The following three requirements render bolt defect detection a challenging problem.

1. TFDS images are blurry due to the high speed of freight trains. For dynamic detection, the low resolution of the image, the blurred image of the bolts, and uncertainties about the environment are difficulties that must be considered.
2. Cameras are installed outdoors, so the texture of the bolt will change due to changes in the illumination conditions. Additionally, the bolts can be contaminated by leaked grease or dust, so it is a challenging task to localize the three bolts in an image size of 1024×1200 pixels.
3. Difficulties also arise from the use of an online detection process. Generally, the shortest interval between freight trains is 5 minutes; this timeframe makes it difficult to apply complex pattern recognition algorithms.

Related work in the literature

The inspection of tracks and freight trains is an important task for the rail industry. For track inspections, visual recognition of fastening bolts is often mentioned in the literature. Based on the shape, fastening bolts can be divided into several categories, such as an elastic rail clip, hook bolt and hexagonal-headed bolt. Most of the literature is based on elastic rail clips [1–4], whereas our research is about hexagonal-headed bolts. Only two studies could be found that discussed defect detection of this kind of bolt.[5,6] Marino et al.[5] proposed a fault inspection system that could be used to automatically detect the absence of hexagonal-headed bolts used to secure rails to sleepers. They used the techniques of wavelet transformation and principal component analysis (PCA) to pre-process images of the track. The converted data were fed into a multilayer perceptron neural network for classification and identification. De Ruvo et al.[6] applied an error back-propagation algorithm to model hexagonal-headed bolts. To achieve real-time performance, they implemented the detection algorithm on graphical processing units. However, these two studies are only effective for good conditions. They are not applicable to dynamic images obtained from a complex environment.

For freight train inspection, methods for diagnosing different components are discussed. Anderson[7] proposed an acoustics-based fault diagnosis approach for roller bearings. Hart et al.[8] designed a detection system for monitoring the condition of disc brakes and the performance of its bearings. Kim and Kim[9] used a binocular vision measurement system to accurately measure the thickness of a brake shoe. TFDS is a vision-based fault detection system for key components of in-service freight trains. Using images taken by TFDS, Zhou et al.[10] combined a gradient-encoding histogram and a support vector machine classifier to inspect for missing handles on an angle cock. Nan et al.[11] proposed an automatic recognition method for the loss of a bolt from the centre plate of a bogie. Liu et al.[12] used Hough transforms and symmetry validation to detect displacement faults in a bearing's weight saddle. Zhu et al.[13] proposed a novel approach to extract complex shapes from TFDS images based on discrete-point sampling and centreline grouping, however, they did not extend their method for use in fault detection. As far as we know, our method is the first to detect bolt defects in complex environments, and it has the following three advantages.

1. Bolts with various forms can be recognized. Defects such as bearing cap loss, bolt loss and broken bolts can be accurately recognized from hundreds of thousands of images.
2. A self-updating strategy is proposed to better prepare training samples.
3. The average processing time for one image is 118 ms, which meets the system's real-time requirement.

Approach overview

When a train passes through a TFDS station, images of its key components are obtained and transmitted to remote computers for use as input data to the image recognition module. As part of the image recognition module, the proposed bolt defect detection method will examine each wheel image, if any defect is found the image will be marked and the situation immediately reported.

The bolt defect detection method is composed of three parts: extraction of the region of interest (ROI), localization of the bolt regions, and identification of defects. Overviews of these procedures are now presented.

1. Extraction of the ROI. As the three bolt regions occupy a small area in the 1024×1200 pixel image, it is essential to extract the wheel bearing's ROI. Here, we use a sub-window searching method to extract the ROI.
2. Localization of the bolt region. We consider local binary pattern (LBP) features to represent the bolt and make use of a rotate-and-slide window method to localize the three bolt regions.
3. Identification of defects. The detected bolt regions are then transmitted to the identification module. In this procedure, the geometrical relationships between the three bolts are fully used, and a hexagonal pattern is proposed to further analyse the three bolt regions.

Extraction of the ROI

We define the wheel bearing region as the ROI of the image. As the distance between the freight train and the installed camera is fixed, the size of the region containing the wheel bearing is within a certain range. As shown in Figure 3, the size of the ROI is defined as 200×200 pixels. To effectively extract the ROI, an object detection approach that employs the LBP descriptor and support vector machine (SVM) is used. In the first part of this section, we give a brief introduction to LBP. We then detail our ROI extraction method, which includes ROI localization and alignment.

LBPs

Due to its grey-scale invariance and computational efficiency, the LBP operator has been widely used in various applications, for example, face recognition.[14]

The original LBP operator labels the pixels of an image by adding the value of the centre pixel to the 3×3 neighbourhood of each pixel and considering the result as a binary number. Figure 4 shows an example of an LBP calculation. The 256-bin histogram of the labels computed over a region can be used as a texture descriptor. Each bin (LBP code) can be regarded as a micro-structure. Local primitives, which are codified by these bins, include different types of curved edges, spots and flat areas. That is why the LBP descriptor is highly discriminative.

The LBP operator has been extended to consider different sizes of the neighbourhood.[15] In general, the operator $LBP_{P,R}$ refers to a neighbourhood size of P equally spaced pixels on a circle of radius R that forms a circularly symmetric neighbour set. We use the four-neighbour operator $LBP_{4,1}$ for the ROI detection.

ROI localization and alignment

To train the ROI classifier, we collected a set of ROI images as the positive training set and a set of

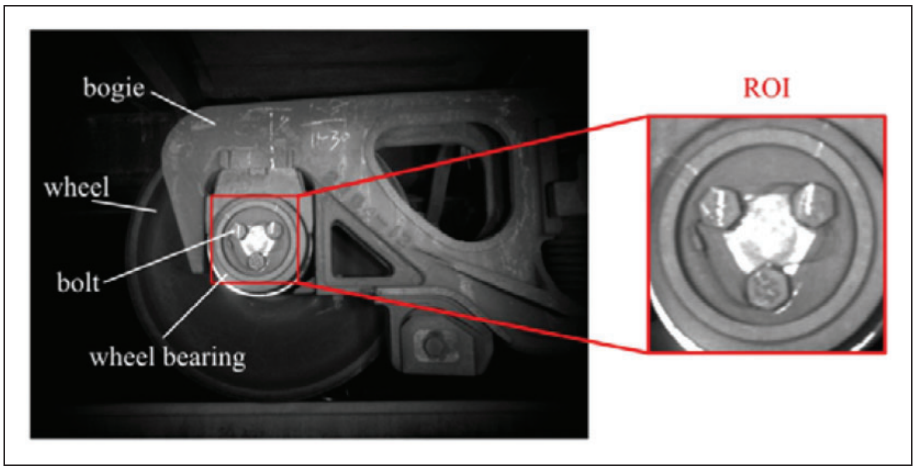

Figure 3. The ROI of the wheel image.

non-ROI images as the negative training set. The procedure for LBP feature extraction can be described as follows. Each training sample is divided into nine overlapping regions of 25×25 pixels (overlap size = 13 pixels). From each region, we compute a 16-bin histogram using the $LBP_{4,1}$ operator and concatenate the results into a single 144-bin histogram. Thus, an LBP feature vector of 144 degrees is obtained. Using the features of the positive and negative samples, we train a second-degree polynomial kernel SVM classifier.

The SVM classifier[16] is extensively studied in statistical learning theory and has been applied to various tasks involving the detection of an object. Given training samples represented by their feature vectors, an SVM classifier can find the separating hyper-plane that has the maximum distance to the closest patterns. To perform a nonlinear separation, the input space is mapped onto a higher-dimensional space using a kernel method.

Let $\mathbf{x}_i$ be the LBP representation of the ith training sample. y_i has a value of 1 or -1 depending on whether $\mathbf{x}_i$ is a positive or negative sample, and N be the number of samples. In a binary classification problem, the decision function of the SVM is

$$f(\mathbf{x}) = \mathrm{sgn}\left(\sum_{i=1}^{N} y_i \alpha_i k(\mathbf{x}_i, \mathbf{x})\right) + \mathrm{b} \tag{1}$$

where $\mathbf{x}$ is the LBP representation of the tested sample, b is a scalar (bias), and $k(\cdot,\cdot)$ the second-degree polynomial kernel function defined by

$$k(\mathbf{x}_i, \mathbf{x}_j) = [\gamma\langle \mathbf{x}_i, \mathbf{x}_j\rangle + 1]^2 \tag{2}$$

where$\langle\cdot,\cdot\rangle$ denotes the dot product in the feature space, and γ is the parameter of the polynomial kernel function, which is fixed to 0.01 in our experiment.

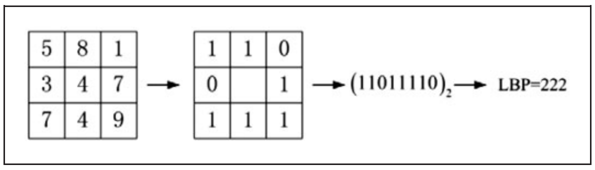

Figure 4. An example of the calculation of a LBP.

Here, α_i is a Lagrange multiplier that corresponds to the sample $\mathbf{x}_i$, recovered by solving the following quadratic programming problem

mimimize

$$W(\alpha) = -\sum_{i=1}^{N} \alpha_i + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{3}$$

subject to

$$\sum_{i=1}^{N} y_i \alpha_i = 0, \quad \forall i; 0 \leqslant \alpha_i \leqslant C \tag{4}$$

where C is the regularization parameter during the training process, which is fixed to a value of one in our experiment.

When the ROI classifier is constructed, we apply the sliding window detection approach to the tested image. The main procedure is described in Figure 5. The ROI's vertical position in the image is limited to be within a certain range, because the trackside cameras are installed at a fixed height. Therefore, we cut the image and resize it before sliding window detection. Then, the image is closely scanned from the top left to the bottom right using rectangular sliding windows. For each sliding window, an LBP feature vector is extracted and fed to the trained classifier.

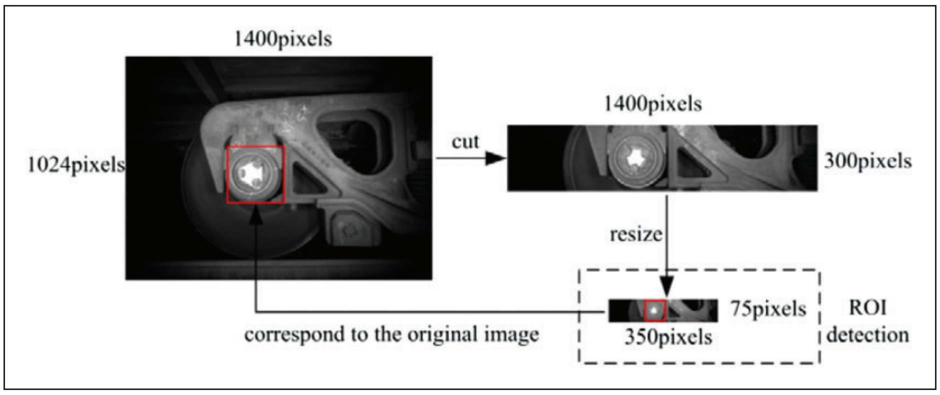

Figure 5. Main procedures of ROI localization.

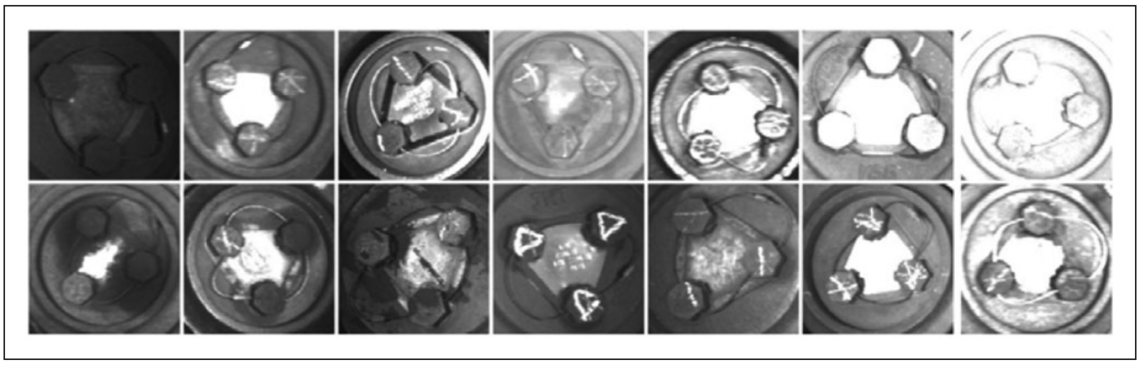

Figure 6. Bolts under different illumination conditions and contaminated by leaked grease or dirt.

This model classifies the sliding windows that bound the ROI as positive and the others as negative. Thus, the position of ROI in the original image is obtained.

After ROI localization, the region containing the wheel bearing is aligned using the ED-circle method.[17] When there is an error in the localization of the ROI, such an alignment is essential for the subsequent localization of the bolt region. This not only increases the accuracy but also saves processing time.

Localization of the bolt region

As the wheel bearing rotates during movement of the train, it is difficult to efficiently obtain accurate positions of the three bolts. In addition, the cameras are installed in outdoor conditions, and images of the bolts can occur at different textures and prototypes subjected to varying illumination conditions. In particular, some bolts are contaminated by leaked grease or dirt. Figure 6 shows some bolts under different illumination conditions and contaminated by leaked grease or dirt.

Bolt description using LBPs

We considered the LBP features as bolt representations. In addition to its grey-scale invariance, the idea of using LBP operators is motivated by the fact that bolt images have textures that can be seen as a composition of micro-patterns described by LBPs. In addition, the contours of bolts are apparent after applying the LBPs. Figure 7 shows different bolt images after the application of LBPs.

The size of the bolt region is defined as 60×60 pixels. To avoid statistical unreliability due to large histograms computed over small regions, we use overlapping regions and the $LBP_{4,1}$ operator to obtain local LBP histograms. As shown in Figure 8, each sample is divided into nine regions of 30×30 pixels (overlap size = 15 pixels). From each region, we compute a 16-bin histogram and concatenate the results. Additionally, to enhance the holistic description of a bolt, we calculate the global LBP histogram over the whole image using the $LBP_{8,1}$ operator. Thus, the length of a bolt feature vector is $16 \times 9 + 256 = 400$. Then, feature vectors of the training samples are trained by the SVM to obtain a hyper-plane to distinguish bolt and non-bolt regions in an ROI image.

Bolt localization using rotate-and-slide windows

The sliding window method is widely used in object detection; however, it takes a large time to generate sub-windows in the 200×200 pixel ROI. The distribution of three bolts is an equilateral triangle. We make full use of this information to be able to quickly locate the three bolts. Compared with the traditional method of sliding from the top left to the bottom

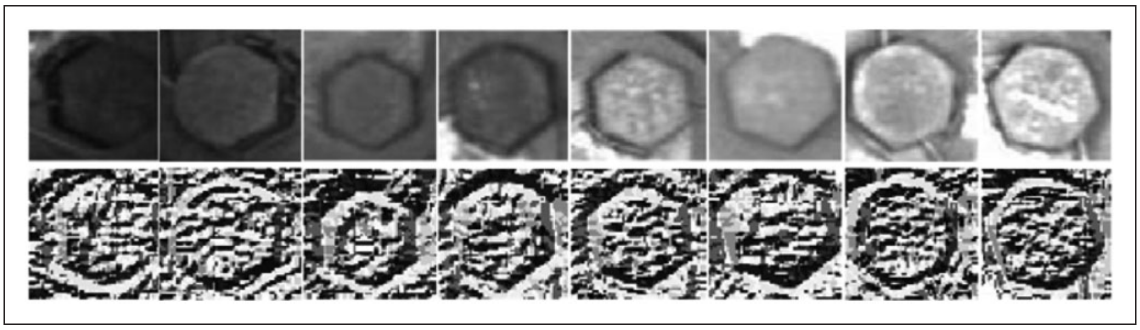

Figure 7. Bolt images after the application of LBPs.

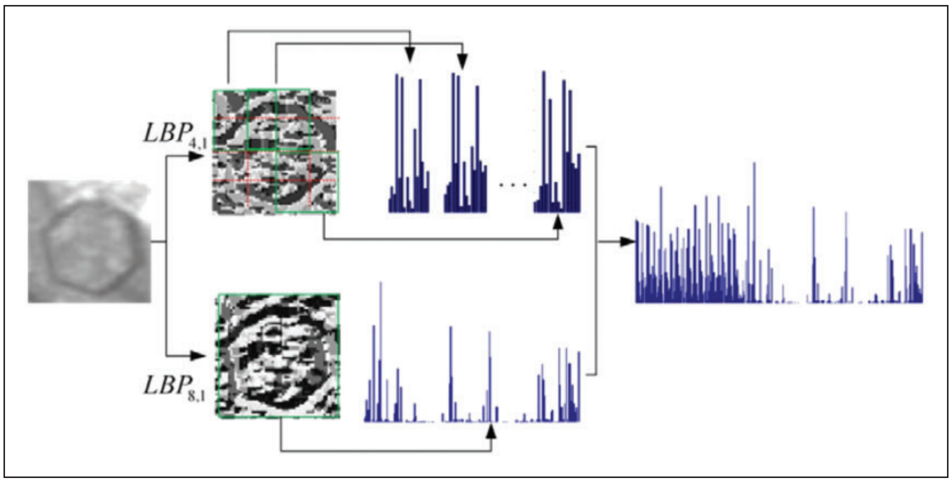

Figure 8. Calculation of the LBP feature vector in a bolt region.

right, we propose a rotate-and-slide window method. The 60×60 pixel sub-window slides around the centre of the wheel bearing, and the incremental steps in the angle and radius are set appropriately in advance. In pictures obtained by TFDS, there are two types of wheel bearings based on the size of the axis. Based on empirical evidence, we set the range of the sliding radius to be 40–65 pixels and the incremental step in the angle and radius to 0.3 rad and 4 pixels, respectively.

The procedure of the rotate- and-slide window can be described as follows.

1. Let the centre of the ROI be the rotation centre (x_c, y_c). With the radius ranging from 40 to 60 pixels, the sub-window rotates around (x_c, y_c) from 0° to 180° in steps of $\Delta\theta$ in a clockwise direction until the prediction of the classifier has a value of one. The current position A (the centre of the detected window) and radius, r, are then recorded.
2. For the given detected position A, we use the classifier to give predictions every 4 pixels and write down the detected windows (represented by its centre) within a neighbourhood of 16×16 pixels. Then, the centroid of the result is the position of the first bolt.
3. Let the radius be r and the sub-window rotate around (x_c, y_c) from 0° to 180° in steps of $\Delta\theta$ in an anticlockwise direction until the prediction of classifier has a value of one. Record the current position B.
4. For the given detected position B, we use the classifier to give predictions every 4 pixels and record the detected windows within a neighbourhood of 16×16 pixels. Then, the centroid of the result is the position of the second bolt.
5. The rough position of the third bolt is calculated through an equilateral triangle constraint. For the rough position C, we use the classifier to give predictions every 4 pixels and write down the detected windows within a neighbourhood of 16×16 pixels. Then, the centroid of the result is the position of the third bolt.

As shown in Figure 9, the green point denotes the rotation centre of the sliding window. The red and blue points form the neighbourhood of 16×16 pixels. Red indicates that the prediction of the classifier has a value of one, whereas blue means the prediction has a value of zero. The bolt's exact position is the centroid of the region formed by the red points.

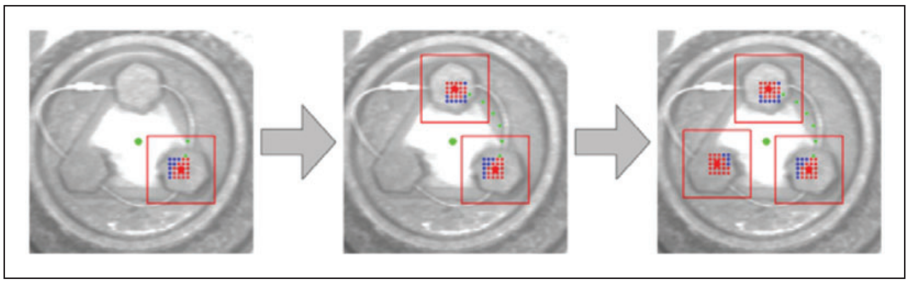

Figure 9. The application of the rotate-and-slide window strategy to the ROI.

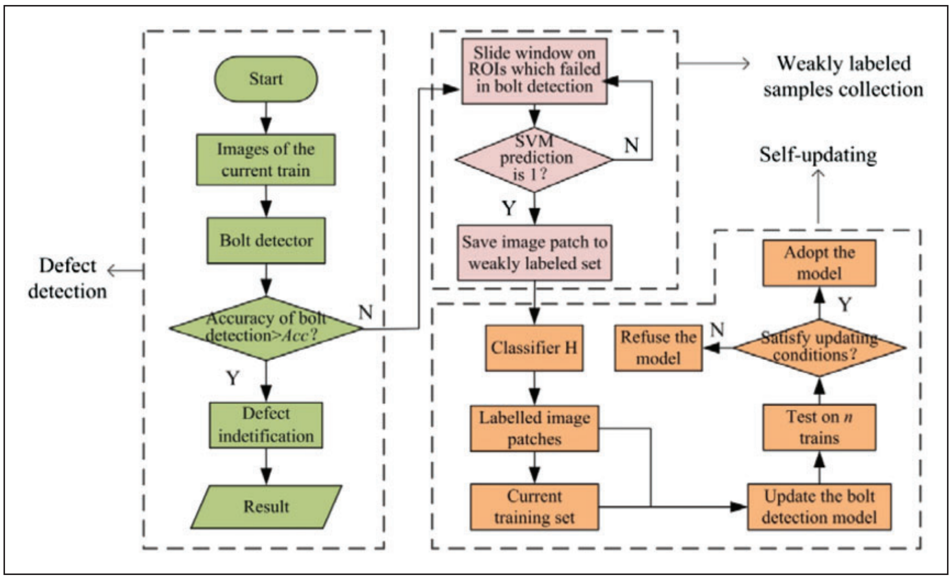
Start
Images of the current train
Bolt detector
Accuracy of bolt detection>Acc?
N
Y
Defect indetification
Result
Defect detection
Slide window on ROIs which failed in bolt detection
SVM prediction is 1?
N
Y
Save image patch to weakly labeled set
Weakly labeled samples collection
Self-updating
Adopt the model
Y
Classifier H
Refuse the model
N
Satisfy updating conditions?
Labelled image patches
Test on n trains
Current training set
Update the bolt detection model

Figure 10. Schematic representation of the self-updating procedure.

Self-updating using a nested classifier

The bolt region detector sometimes fails to locate the three bolts for the following two reasons. One is that the prototype of a normal bolt cannot be recognized by the classifier. The other is that there is at least one defective bolt that is classified as a non-bolt. In the case of a defective bolt, we generate an alarm when the bolt detector cannot locate the three bolts. However, this behaviour may increase the false alarm rate. In other words, a large number of normal images are considered to be defective due to the failure in bolt region localization. Therefore, it is essential to improve the classifier's performance.

In addition to the selection of discriminative features, effective training is also critical to ensure a good performance of the classifier. The bootstrap method [18] is a typical method for effective training; however, it requires the wrongly classified patterns to be found manually. Because bootstrap is an iterative process, the preparation of a training set that covers sufficient variances in bolt appearance is time-consuming. Therefore, it is preferable to develop an approach that automatically updates the classifier based on new samples.

The procedure of self-updating. A schematic representation of the training procedure, termed 'self-updating', is presented in Figure 10. In practice, when a freight train drives through a TFDS station, the image recognition module is called to find defective components on that train. The bolt detector is composed of a classifier L based on the LBP descriptor and the rotate-and-slide window method. If the detection rate of bolts is low, the proposed self-updating procedure is called during the interval before the next train arrives. A nested classifier H is used to continuously classify incoming weakly labelled samples.

The self-updating procedure can be summarized as follows. Set_0 is the initial training set of the labelled data, $wSet$ is the weakly labelled training set. T_1, T_2 are the thresholds related to the updating efficiency. Acc is the bolt detection rate. L and H are classifiers based on the LBP and histogram of the oriented gradient (HOG)[19] descriptors, respectively.

1. Construct L_0 and H_0 based on Set_0, set values for Acc, T_1, T_2.
2. For each image of the kth train ($k \geqslant 1$), locate the three bolt regions using L_{k-1}.

3. If the bolt detection rate $< Acc$, then slide the window from the top left to the bottom right of any ROI that failed to localize the three bolts and collect image patches to $wSet_k$ when the prediction of L_{k-1} has a value of one.
4. Use H_0 to predict samples in $wSet_k$. If H_0 predicts a label for a certain example with a high confidence, then that labelled example is added to the training set Set_{k-1} to form Set_k, otherwise the example is ignored.
5. Construct L_k based on Set_k and use it to localize bolts for the previous n trains (if $k \leqslant n$, let $n = k$). Count the number of ROIs that failed to localize the three bolts. $d_{k-n}, d_{k-n+1}, \ldots, d_k$ are the numbers of reductions of the failed ROI when using L_k for the previous n trains, respectively.
6. If $d_i > T_1 (i = k - n, k - n + 1, \ldots, k)$ and $d_{k-n} + d_{k-n+1} + \cdots + d_k > T_2$, we adopt classifier L_k, otherwise we refuse it.
7. Let $k = k + 1$, and go back to step 2.

Construction of the nested classifier. Generally, the representative capability of the nested classifier H is higher than classifier L, but its processing time is longer. We combined these two classifiers to achieve a better performance in both representative capability and processing time.

The HOG descriptor is widely accepted as one of the best methods to capture an edge or local shape information. Classifier H is constructed based on the HOG descriptor. We extract the HOG feature using the procedure described in Yuen et al.[20] Figure 11 shows the local region segmentation method for HOG. The detection window (60×60 pixels) is divided into 6×6 cells (100 pixels for each cell). Then, we group 2×2 adjacent cells into larger spatial blocks and normalize each block separately. Because the stride of each block is 10 pixels, there are 5×5 blocks in total. The HOG descriptor is a concatenated vector extracted from all of the blocks in the detection window. Thus, the dimensions of the feature vector are $36 \times 25 = 900$. Finally, we apply PCA to decrease the dimensions to 100.

Limited by the performance of the nested classifier, it is possible for incorrect samples to be collected in the training set. A few incorrect samples cannot ‘corrupt’ the model because SVMs are highly tolerant. However, if the model is corrupted, we need to be able to quickly realize that corruption has occurred, so updating conditions are added. On the other hand, we can make use of the structural constraint of the three bolts to control the number of incorrect samples in the training set.

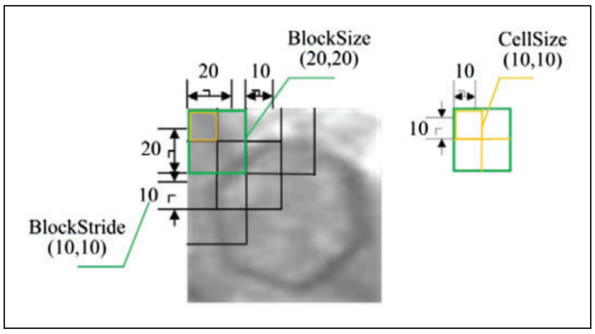

Figure 11. Local region segmentation method for HOG.

Defect identification

After obtaining the three bolt regions, the next problem is to find defective bolts. Applying training-based methods is a difficult task to perform due to an insufficient number of defective samples. We only obtained 28 defective images from hundreds of thousands of TFDS images. Figure 12 shows some ROIs with defective bolts. Compared with normal bolts, they have no hexagonal pattern. Moreover, the differences among the three bolts are apparent when there are bolt defects.

We first use the Hough transform method[19] to extract a circle from the bolt regions. If a circle with a radius smaller than 14 pixels is detected, there is a missing bolt on the wheel bearing. Otherwise, we calculate six features in the circle: radius, maximum intensity, minimum intensity, difference between the maximum and minimum intensity, average intensity, and standard deviation. Then, a vector $\mathbf{v}_i (i = 1, 2, 3)$ representing the bolt region is formed after weighting and normalization. We use the Euclidean distance $d_{ij} = \ ||\mathbf{v}_i - \mathbf{v}_j\ ||_2 (i, j = 1, 2, 3, i \neq j)$ to describe the difference between the two vectors. There may be a defective bolt if the difference between every two Euclidean distances $\Delta d = \max\{d_{12} - d_{13},$

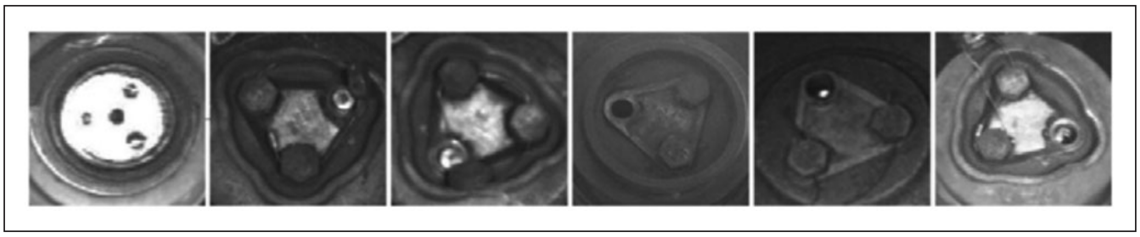

Figure 12. Examples of defective bolts.

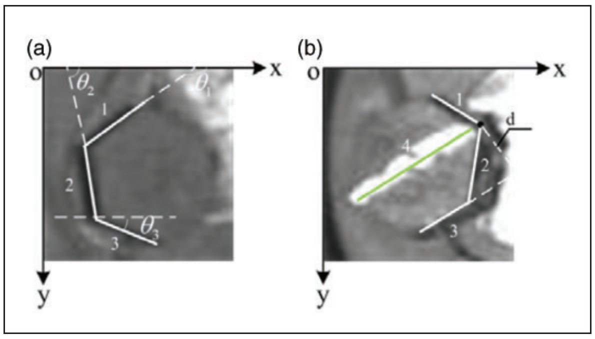

Figure 13. The hexagon pattern observed for normal bolts (a) three edges satisfying the hexagonal pattern, and (b) four edges satisfying the hexagonal pattern but the fourth one is not the bolt edge.

$d_{12} - d_{23}, d_{13} - d_{23}\}$ is bigger than the threshold. Then, it is necessary to go to the next step.

Based on the hexagonal pattern, we can efficiently separate defective bolts from normal bolts. A hexagonal pattern means that the edges of a normal bolt form a hexagon. In real-world situations, some bolt images have out-of-plane rotation due to the angle of the camera. Therefore, if there are three edges satisfying the hexagonal pattern shown in Figure 13, the bolt is normal. To prevent the situation described in Figure 13(b), we add a distance constraint. The pseudo-code of the algorithm is as follows.

```
Algorithm: search for line groups satisfying the hexagonal pattern
Input: a list Line which include n lines, $l_i = (x_{i1}, y_{i1}, x_{i2}, y_{i2}, \theta_i)$
output: line groups L
for each line $l_i, l_j\ (i \neq j)$ in Line do
  If $50^\circ < |\theta_i - \theta_j| < 70^\circ$
    for each line $l_k (k \neq j, i)$ in Line do
      $(x, y) \Leftarrow$ intersection of $l_i$ and $l_j$
      $d \Leftarrow$ distance between $l_k$ and $(x, y)$;
      If $50^\circ < |\theta_j - \theta_k| < 70^\circ, 110^\circ < |\theta_i - \theta_k| < 120^\circ, 19 < d < 21$
        $L \Leftarrow [L; l_i, l_j, l_k]$;
      end
    end
  end
end
```

The list of lines is obtained by a line segment detector (LSD) [21] that consists of three steps: finding the line-support regions, taking a rectangular approximation of the regions, and line segment validation. Although LSD gives accurate results and requires no parameter-tuning, it is necessary to adjust the parameter ε(threshold of the number of false alarms) to obtain clear edges for the bolt. We remove some short lines based on the size of the bolt. Let HP be the number of bolt regions that detected a hexagon pattern. If $HP = 3$, the tested image is considered normal; otherwise, it is considered to be defective.

Experimental results

Experiments on a larger number of images were conducted to validate our proposed method. The training set Img_{train} contains a total of 2872 original images collected from six freight trains; each image in Img_{train} has three normal bolts. The test set Img_{test} is composed of 10,248 original images from 26 freight trains and 28 images with bolt defects.

Classifier training

For ROI extraction, we used the approach described in the section 'Extraction of the ROI'. We collected a set of 2397 ROI images from Img_{train} as the positive training set. The negative training set, which included 2500 non-ROI images, was extracted from images without a ROI. All of the 200×200 pixel samples were resized to 50×50 pixels. Based on the LBP descriptor, we trained a second-degree polynomial kernel SVM classifier. Then, we used the bootstrap method, which was performed five times, until the detection rate of the ROI was 100%. The final negative training set was composed of 2797 non-ROI images. To decrease the localization error in ROI detection, we applied the ED-circle method to align the position of the wheel bearing. As shown in Figure 14, the red rectangle indicates the initial position of the ROI, and the green indicates the position after alignment.

To reduce the effort required to prepare the training set, we implemented the self-updating approach to construct the bolt detector. Initially, 2448 bolt samples and 2700 non-bolt samples were cut from images of two trains in Img_{train}. Based on these samples, we constructed SVMs classified as L_0 and H_0 as the starting point of our procedure. Then, we ran the bolt detector combined with the self-updating strategy. When a freight train drives through the TFDS station, the bolt detector is called and outputs the detection rate. In the experiment, we set $Acc = 93\%$, $T_1 = -5, T_2 = 4$, and acquired eight SVM models $L_1, L_2, L_3, L_4, L_5, L_6, L_7, L_8$ by the self-updating approach. Figure 15 shows the performance of these models. A model trained in this manner can achieve robust results for a large variety of trains.

Receiver operating characteristic (ROC) graphs are useful for organizing classifiers and visualizing their performance.[22] We collected 10,000 bolts and 10,000 non-bolts from Img_{test} as another test set. The ROC curves for classifier L_0 (blue) and L_8 (red) in the test set are shown in Figure 16. The figure shows the improvement in performance and demonstrates the effectiveness of the self-updating approach.

Defect detection results

After self-updating, the number of positive and negative samples reached 3918 and 3992, respectively. Classifier L_8 was constructed based on these samples.

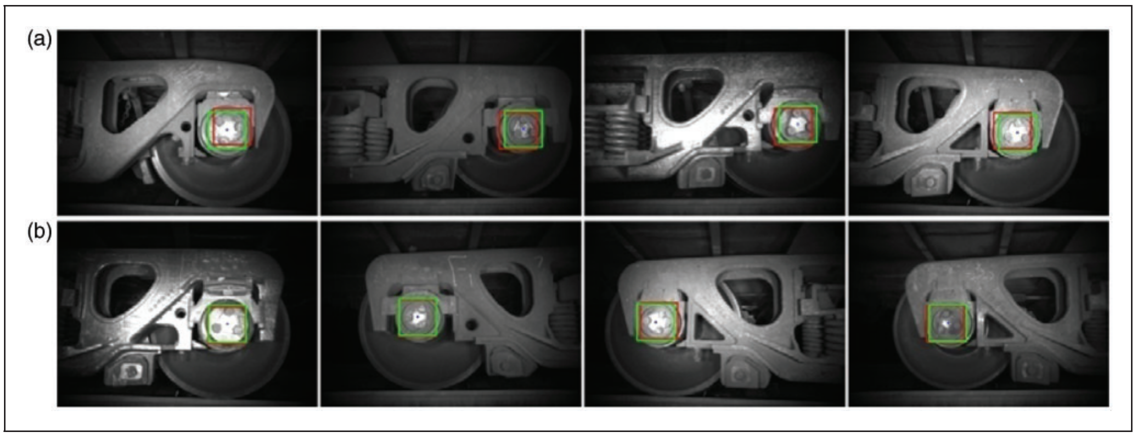

Figure 14. Some results after ROI extraction (a) large alignment by the ED-circle method, and (b) small alignment by the ED-circle method.

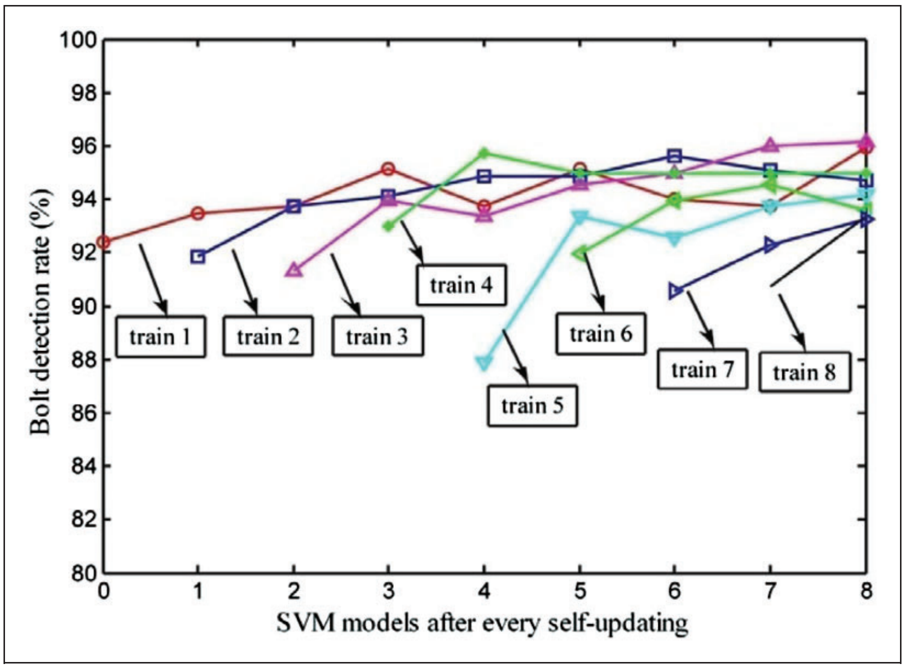

Figure 15. Bolt detection rate for passing trains.

Figure 17 shows the experimental results of the rotate-and-slide window. Failures in bolt localization are due to bad image quality and overexposure.

In the three detected bolt regions, the Euclidean distance and hexagonal pattern were used to exclude normal bolts. As show in Figure 18(a), a normal bolt was judged immediately by$\Delta d \leqslant 0.2$. If $\Delta d > 0.2$, the next step was to find a hexagonal pattern in each bolt region. As show in Figure 18(b), blue lines were detected by LSD, and red lines created the hexagonal pattern. The image was considered normal only when $HP = 3$. As shown in Figure 18(c), the above principle results in false alarms if the edge of the bolt appears to be blurred, this is inevitable due to the image quality.

The whole recognition method was conducted on a test set including 10,276 images. Figure 19 shows some experimental results for images with bolt defects. There are two situations that result in an image being considered defective. One is a failure to localize the three bolts in the ROI. The other is a failure to find a hexagonal pattern in each bolt region. This scheme results in false alarms. The test results are listed in Table 1, where total images means all of the tested images, including the normal ones and defective ones; defective images means images with bolt defects; total alarm is the number of images predicted as being defective by our proposed method; correct alarm is the number of images correctly predicted as being defective. The value of defective images is equal to correct alarm. This means that the number of images predicted as being defective is equal to the number of actual defective images. In other words, there are no missing bolt defect images. Defect detection rate and false alarm rate

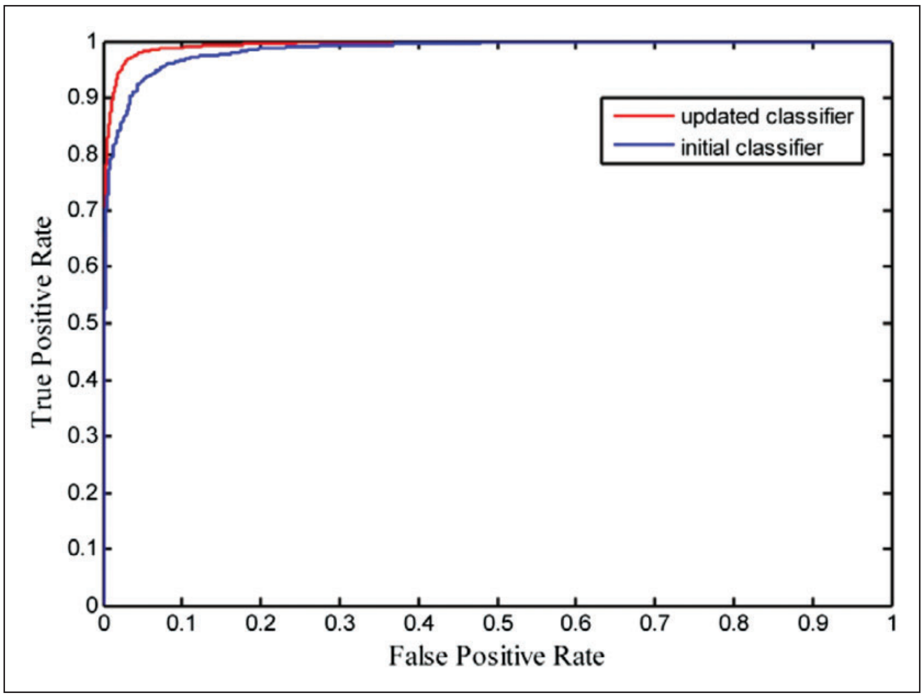

Figure 16. ROC curve for the bolt localization classifier *L*.

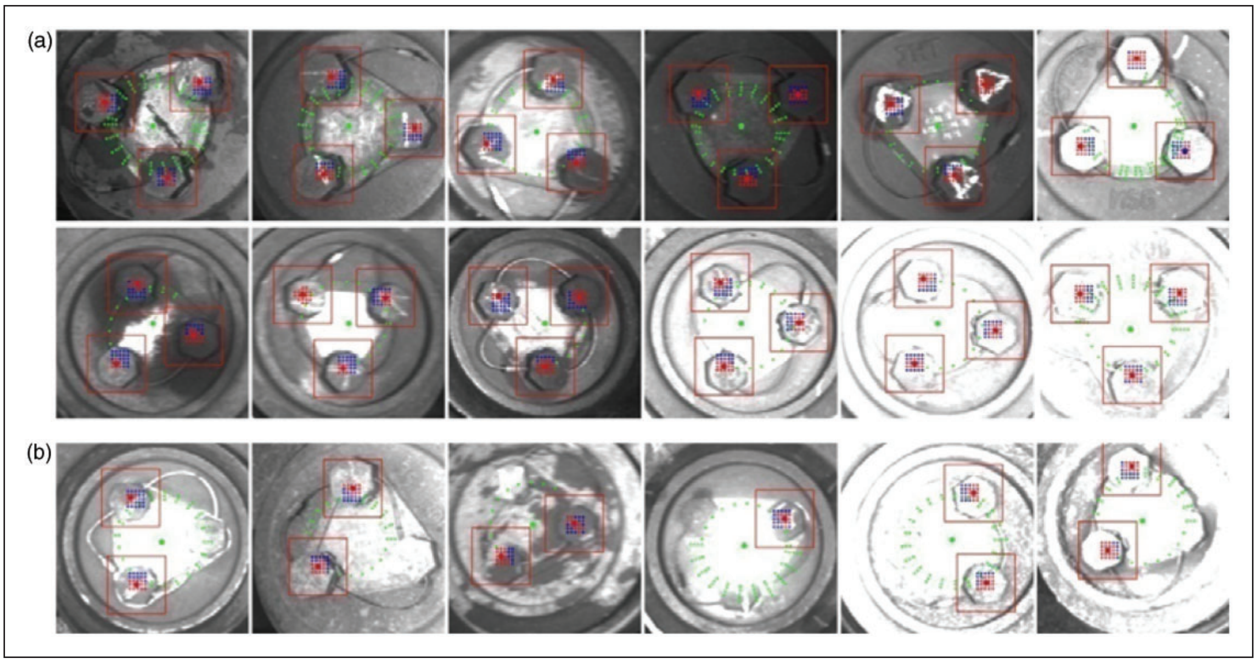

Figure 17. Localization of the three bolts (a) results of correct localization, and (b) results of failed localization.

are defined as follows. Table 2 shows the results of the three main modules.

$$\textit{defect detection rate} = \frac{\textit{correct alarm}}{\textit{actually defective images}} \tag{5}$$

$$\textit{false alarm rate} = \frac{\textit{total alarm} - \textit{correct alarm}}{\textit{actually normal images}} \tag{6}$$

Real-time processing is a necessary property in TFDS. For 1400 × 1024 pixel images, the average processing time of our method is 118 ms using the

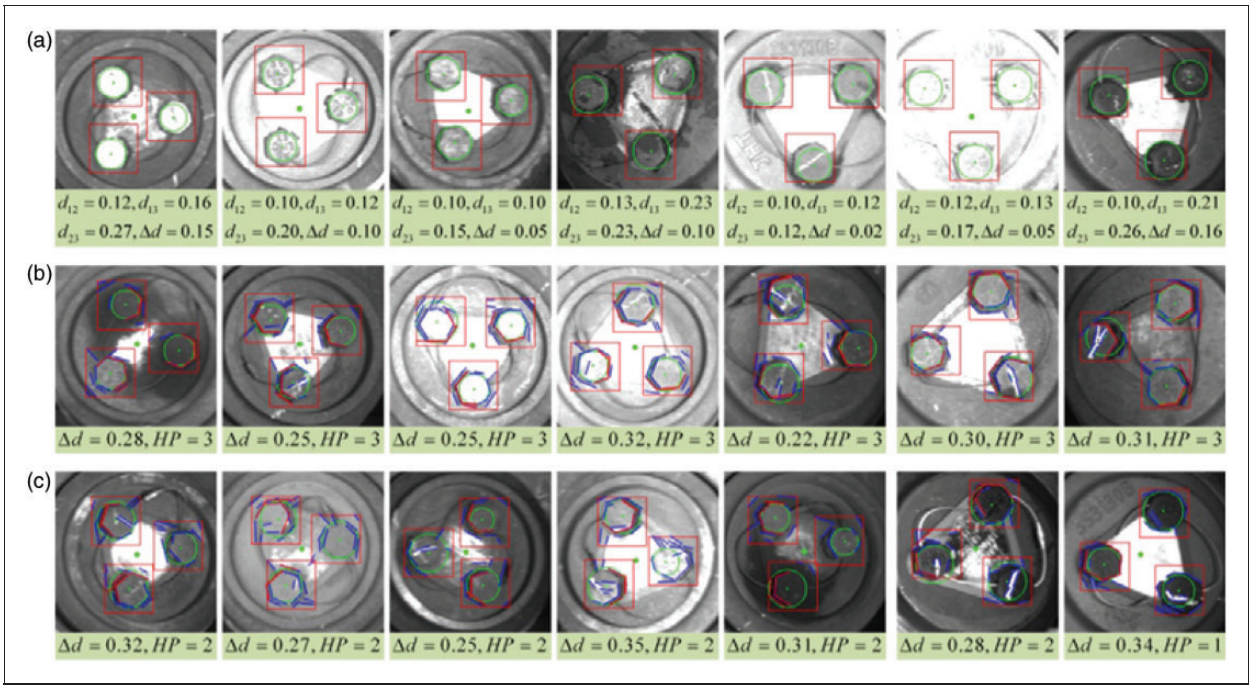
(a)
$d_{12}=0.12, d_{13}=0.16$ $d_{23}=0.27, \Delta d=0.15$
$d_{12}=0.10, d_{13}=0.12$ $d_{23}=0.20, \Delta d=0.10$
$d_{12}=0.10, d_{13}=0.10$ $d_{23}=0.15, \Delta d=0.05$
$d_{12}=0.13, d_{13}=0.23$ $d_{23}=0.23, \Delta d=0.10$
$d_{12}=0.10, d_{13}=0.12$ $d_{23}=0.12, \Delta d=0.02$
$d_{12}=0.12, d_{13}=0.13$ $d_{23}=0.17, \Delta d=0.05$
$d_{12}=0.10, d_{13}=0.21$ $d_{23}=0.26, \Delta d=0.16$
(b)
$\Delta d=0.28, HP=3$
$\Delta d=0.25, HP=3$
$\Delta d=0.25, HP=3$
$\Delta d=0.32, HP=3$
$\Delta d=0.22, HP=3$
$\Delta d=0.30, HP=3$
$\Delta d=0.31, HP=3$
(c)
$\Delta d=0.32, HP=2$
$\Delta d=0.27, HP=2$
$\Delta d=0.25, HP=2$
$\Delta d=0.35, HP=2$
$\Delta d=0.31, HP=2$
$\Delta d=0.28, HP=2$
$\Delta d=0.34, HP=1$

Figure 18. Results of the defect identification module: (a) normal ROI determined by $\Delta d \leqslant 0.2$; (b) normal ROI determined by $HP = 3$; and (c) false alarms determined by $HP < 3$ due to blurry edges.

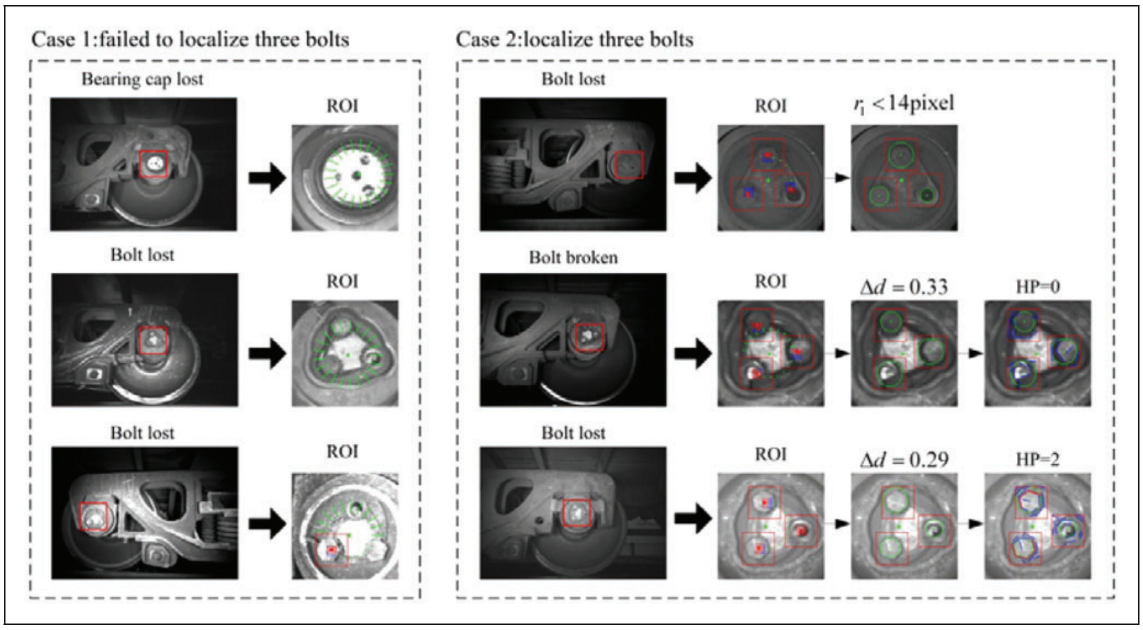

Figure 19. Experimental results for images with bolt defects.

Table 1. Test results of the proposed recognition method.

Total images	Defective images	Total alarm	Correct alarm	Defect detection rate (%)	False alarm rate (%)	Average time (ms)
10,276	28	860	28	100	8.1	106

Table 2. Test results of the three main modules.

Total images	ROI detection rate (%)	Bolt detection rate (%)	False alarms in defect identification module
10,276	100	95.2	366

Table 3. Average processing time of each module.

Main module	Average time (ms)	Percentage (%)
Localization of ROI	35	29.7
Localization of bolt	71	60.2
Confirmation of defective bolt	12	10.1

following configuration: 3.6 GHz Intel Xeon E5-1620 processor (four cores and eight threads), 8 GB RAM, and Win7. The detailed computing time of each module is listed in Table 3.

Conclusions

This paper introduced a novel method for recognizing bolt defects under complex conditions via computer vision techniques.

1. The proposed recognition method consists of two main procedures: bolt localization and defect identification. To localize bolt regions effectively, we apply a coarse-to-fine strategy and make full use of the distribution of the three bolts on the wheel bearing. The defect identification procedure is based on further analysing the bolt regions. We combined the similarity of bolt regions in normal ROI and a hexagonal pattern to find defective bolts among a number of normal bolts.
2. Our self-updating strategy in classifier training has the advantage of avoiding the need to manually find the wrongly classified patterns, which improves the training efficiency.
3. In this paper, we focused on improving the defect detection rate. Some false alarms occurred in the defect identification module because of blurred edges. Further work will be devoted to improving the quality of raw images and reducing the false alarm rate.

Funding

This work is supported by the National Key Scientific Instrument and Equipment Project of China (grant 2012YQ140032).

References

1. Hsieh HY, Chen NM and Liao CL. Visual recognition system of elastic rail clips for mass rapid transit systems. In: *The joint rail conference and Internal Combustion Engine Division of the ASME spring technical conference*, Pueblo, CO, 13–16 March 2007, paper no. JRC/ICE2007-40080, pp.319–325. New York: ASME.
2. Yang J, Tao W, Liu M, et al. An efficient direction field-based method for the detection of fasteners on high-speed railways. *Sensors* 2011; 11(7): 7364–7381.
3. Xia Y, Xie F and Jiang Z. Broken railway fastener detection based on adaboost algorithm. In: *The international conference on optoelectronics and image processing*, Haiko, China, 11–12 November 2010, pp.313–316. New York: IEEE.
4. Feng H, Jiang Z, Xie F, et al. Automatic fastener classification and defect detection in vision-based railway inspection systems. *IEEE Trans Instrum Meas* 2014; 63(4): 877–888.
5. Marino F, Distante A, Mazzeo PL, et al. A real time visual inspection system for railway maintenance: automatic hexagonal-headed bolts detection. *IEEE Trans Syst Man Cybern, Part C: Appl Rev* 2007; 37(3): 418–428.
6. De Ruvo P, Distante A, Stella E, et al. A GPU-based vision system for real time detection of fastening elements in railway inspection. In: *The 16th IEEE international conference on image processing*, Cairo, Egypt, 7–10 November 2009, pp.2309–2312. New York: IEEE.
7. Anderson GB. Acoustic detection of distressed freight car roller bearings. In: *The joint rail conference and Internal Combustion Engine Division of the ASME spring technical conference*, Pueblo, CO, 13–16 March 2007, pp.91–103. New York: ASME.
8. Hart J, Resendiz E, Freid B, et al. Machine vision using multi-spectral imaging for under carriage inspection of railroad equipment. In: *The eighth world congress on railway research*, Seoul, Korea, 12– 12 May 2008, pp.1–8. Seoul: WCRR.
9. Kim HC and Kim WY. Automated inspection system for rolling stock brake shoes. *IEEE Trans Instrum Meas* 2011; 60(8): 2835–2847.
10. Zhou F, Zou R, Qiu Y, et al. Automated visual inspection of angle cocks during train operation. *Proc IMechE, Part F: J Rail Rapid Transit* 2014; 228(7): 794–806.
11. Nan L, Wei ZZ, Gao ZP and Wei XG. Automatic fault recognition for losing of train bogie center plate bolt. In: *The IEEE 14th international conference on communication technology,* Chengdu, China. 9–11 November 2012, pp.1001–1005. Piscataway, NJ: IEEE Press.
12. Liu ZH, Xiao DY and Chen YM. Displacement fault detection of bearing weight saddle in TFDS based on Hough transform and symmetry validation. In: *The ninth international conference on FSKD*, Chongqing, China, 29–31 May 2012, pp.1404–1408. New York: IEEE.
13. Zhu ZX, Wang GY, Liu JG, et al. Fast and robust 2D-shape extraction using discrete-point sampling and centerline grouping in complex images. *IEEE Trans Image Process* 2013; 22(12): 4762–4774.
14. Ahonen T, Hadid A and Pietikainen M. Face recognition with local binary patterns. In: *The eighth European conference on computer vision*, Prague, Czech Republic, 11–14 May 2004, paper no. 3021, pp 469–481. Berlin, Germany: Springer.

15. Ojala T, Pietikainen M and Maenpaa T. Multiresolution gray scale and rotation invariant texture classification with local binary patterns. *IEEE Trans Pattern Anal Mach Intell* 2002; 24(7): 971–987.
16. Chang CC and Lin CJ. LIBSVM: a library for support vector machines. *ACM Trans Intell Syst Technol* 2011; 2(3): 1–27.
17. Akinlar C and Topal C. EDCircles: a real-time circle detector with a false detection control. *Pattern Recog* 2013; 46(3): 725–740.
18. Sung K and Poggio T. Example-based learning for view-based human face detection. *IEEE Trans Pattern Anal Mach Intell* 1998; 20(1): 39–51.
19. Navneet D and Bill T. Histograms of oriented gradients for human detection. In: *The IEEE Computer Society conference on computer vision and pattern recognition*, San Diego, CA, 20–25 June 2005, paper no. 886–893. New York: IEEE.
20. Yuen HK, Princen J, Illingworth J, et al. Comparative study of Hough transform methods for circle finding. *Image Vision Comput* 1990; 8(1): 71–77.
21. Von Gioi RG, Jakubowicz J, Morel JM, et al. LSD: a fast line segment detector with a false detection control. *IEEE Trans Pattern Anal Mach Intell* 2010; 32(4): 722–732.
22. Fawcett T. An introduction to ROC analysis. *Pattern Recog Lett* 2006; 27(8): 861–874.