Neural Computing and Applications
https://doi.org/10.1007/s00521-020-05366-9

ORIGINAL ARTICLE

Structural health monitoring of railway tracks using IoT-based multi-robot system

Srikrishna Iyer[1] · T. Velmurugan[2] · A. H. Gandomi[3] · V. Noor Mohammed[2] · K. Saravanan[2] · S. Nandakumar[2]

Received: 12 July 2019 / Accepted: 16 September 2020
© Springer-Verlag London Ltd., part of Springer Nature 2020

Abstract

A multi-robot-based fault detection system for railway tracks is proposed to eliminate manual human visual inspection. A hardware prototype is designed to implement a master–slave robot mechanism capable of detecting rail surface defects, which include cracks, squats, corrugations, and rust. The system incorporates ultrasonic sensor inputs coupled with image processing using OpenCV and deep learning algorithms to classify the surface faults detected. The proposed Convolutional Neural Network (CNN) model fared better compared to the Artificial Neural Network (ANN), random forest, and Support Vector Machine (SVM) algorithms based on accuracy, R-squared value, F1 score, and Mean-Squared Error (MSE). To eliminate manual inspection, the location and status of the fault can be conveyed to a central location enabling immediate attention by utilizing GSM, GPS, and cloud storage-based technologies. The system is extended to a multi-robot framework designed to optimize energy utilization, increase the lifetime of individual robots, and improve the overall network throughput. Thus, the Low Energy Adaptive Clustering Hierarchy (LEACH) protocol is simulated using 100 robot nodes, and the corresponding performance metrics are obtained.

Keywords Multi-robot system · Convolutional neural network (CNN) · Artificial Neural Network (ANN) · Random forest · Support Vector Machine (SVM) · LEACH protocol

1 Introduction

Indian railways provide a major mode of transportation for more than 25 million people every single day. Now, the world's third-largest rail network, structurally monitoring railway tracks, has become an integral part of avoiding railway accidents and loss of life. While the number of railway mishaps has reduced over the last ten years, the death toll due to the derailment has risen significantly. In 2016–2017, 193 people died in these accidents out of which 78 of these accidents occurred due to derailment, bringing the count to an all-time high in the past decade. According to a study conducted, in the 6 years between 2009 and 2015, there were a total of 803 accidents in Indian Railways killing 620 people and injuring 1855 people. Moreover, 47% of these accidents were due to the derailment of trains making it the primary cause for significant train accidents in India over the past decade, and this number has only worsened over the last couple of years where almost 53% of 586 train accidents that occurred

✉ T. Velmurugan
tvelmurugan@vit.ac.in

Srikrishna Iyer
srikrishna.iyer2015@vit.ac.in

A. H. Gandomi
gandomi@uts.edu.au

V. Noor Mohammed
vnoormohammed@vit.ac.in

K. Saravanan
kasisaravanan@vit.ac.in

S. Nandakumar
snandakumar@vit.ac.in

[1] Software Engineer II, ASM Pacific Technology, Yishun Avenue, Singapore 768924, Singapore

[2] School of Electronics Engineering, VIT University, Vellore, India

[3] Faculty of Engineering and IT, University of Technology Sydney, Sydney, Australia

were due to derailments. Therefore, these issues must be addressed. The main problem is caused due to old and antecedent tracks. Figure 1 shows four types of major rail surface faults that contribute to derailments. Rust caused due to humid or wet weather can cause the railway tracks to be brittle. Deep fractures or cracks can be formed as a result of continuous contraction and thermal expansion during fluctuating weather conditions in India. Corrugations generally arise due to differential wear caused by excessive friction between the wheel and the rail when accelerating, braking, or lateral motion across the rail. Squats are found to be initiated by wheel slip or rolling contact fatigue. Extreme cases of rust, squats, and corrugations ultimately lead to the formation of fractures or cracks. Hence, the proposed system ensures timely detection of rust, squats, and corrugations, besides crack detection [1]. The system relies mostly on excessive manpower which leaves Indian railways behind in the race with the ongoing technological revolution. This usually puts maintenance personnel at risk of injury as studies show that an average of 400 men died, while on duty, mostly hit by trains. An efficient way is needed to improve the safety of both maintenance personnel and passengers without using much manpower by making the system more autonomous [2–4]. Currently, the fault detection techniques are largely carried out by manual inspection, which is cumbersome. This system of fault-checking requires plenty of manual labor and time, and it is highly inefficient. The dependence of this system on human judgment and external factors such as ambient light raises quite a few red flags due to a high likelihood of misdiagnosing or inadvertently ignoring some errors, making the system unreliable and in need of a desperate fix. The current research mainly aims to tackle the issues related to deep fractures in the railway track [5].

In this paper, it is proposed hardware implementation of a two-robot system; a master and a slave to structural monitor and detect faults as shown in Fig. 1 on a railway track. RF communication protocol is used to coordinate the movement between the two bots, i.e., the slave is designed to mimic the master bot movement. These bots consist of Ultrasonic sensors for detecting deep fractures, squats and corrugations. Also, the sensors are coupled with a pi camera to capture images to validate the presence of faults which include fractures, squats, and rust. The captured images of rust and cracks are trained and validated using a CNN model. To test its performance, the proposed CNN model is compared to machine learning algorithms namely, ANN, random forests and SVM based on accuracy, R-squared value, Mean-squared error, and F1 score. Furthermore, this solution is extended by simulating a multi-robot environment using the LEACH protocol - essentially a TDMA-based MAC protocol which helps to minimize energy consumption while routing the information to the base station. The protocol was tested using MATLAB, and conclusive results based on certain performance metrics were obtained to determine suitable network parameters. The novelty of this paper lies in the fact that this is the only proposed system where deep learning, computer vision, and ultrasonic sensor technology are utilized to detect not one but four different types of faults. Also, in a practical scenario, this paper posits a communication protocol for coordinated movement and data transfer in a multi-robot system that allows minimization of energy utilization.

2 Literature review

Current research on Structural Health Monitoring (SHM) is focused on acoustic and vision-based systems to monitor and detect structural faults and deformities. A drone-based vision system is implemented which utilizes aerial images captured to perform pre-processing and gauge measurement analysis to detect abnormalities on railway tracks [6]. However, overall costs may be too high since drones are expensive and may require high-end cameras with features like optical image stabilization. An autonomous vehicle

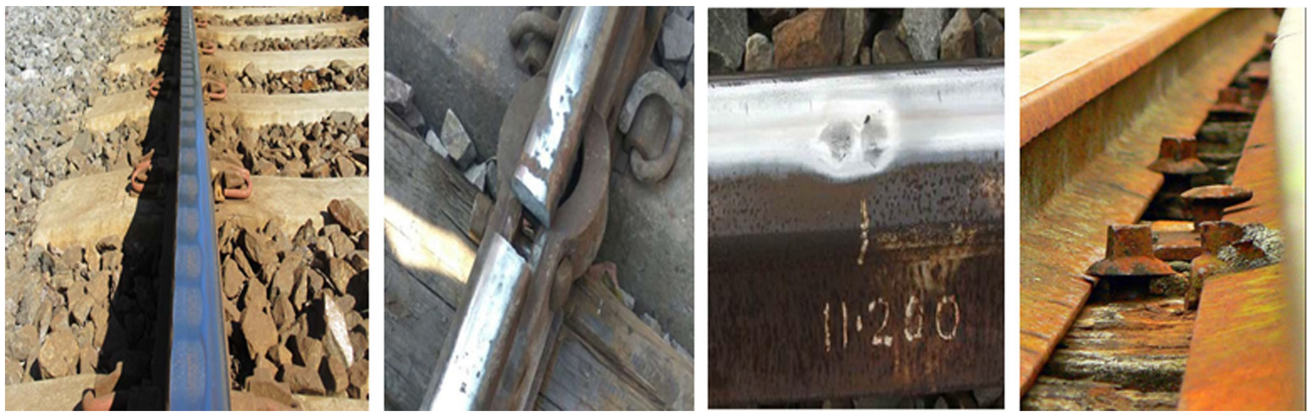


Fig. 1 Rail surface corrugations, crack, squats, and rust (from left to right)

using a PIC microcontroller with obstacle sensors setup was later proposed. This vehicle was capable of monitoring the location of the crack using a GPS module and sending text alerts through a GSM module [7]. The processing time of this system is approximately 0.245 s, and the error rate is claimed to be 5% but the system proposed has to be manually controlled by pushing the equipment which can be cumbersome. A vision-based automated system (using RPi) is also proposed which utilizes a two-stage pipeline that consists of canny edge detection and wavelet transform both of which have massive computational overheads and false detections of unwanted edges [8]. A wavelet transform-based indicator is proposed to detect abrupt vertical accelerations from an IMU (Inertial Measurement Unit) fixed on the train wheel [9]. However, timely prediction of a rail surface defect is not possible; hence, acoustic emission(AE) monitoring is utilized to monitor and study the growth of cracks in railway lines even in the presence of unwanted machine noise [10]. This system has been further extended by extracting features of an AE signal in its frequency domain and performing threshold-based classification [11]. Furthermore, an AE-based system is proposed, which extracts signal features using non-negative matrix factorization. The extracted features are then fed as input to a relevance vector machine (RVM) for classification [12]. Since acoustic vision technology cannot adequately detect small cracks or corrosion, a real-time Visual Inspection System (VIS) is introduced in [13] which involves image localization of track area, image enhancement using the Local Normalization (LN) method, and finally fault detection using Defect Localization Based on Projection profile (DLBP) algorithm. To eliminate the problem of illumination in [13], a vision-based system is proposed that uses the local Michelson-Like Contrast (MLC) to enhance captured images and performs automatic thresholding using Proportion Emphasized Maximum Entropy (PEME) [14]. A machine vision-based prototype is implemented by defining a Region of Interest (RoI) to extract the target area of a railway track image, perform morphological processing to obtain an outline of cracks, and carry out feature extract using direction chain coding [15]. To detect internal rail defects, an ultrasonic air-coupled guided wave signal detection prototype is implemented to help monitor the structural integrity of a railway line [16]. To detect rail squats, a wavelet spectrum analysis based algorithm is presented using axle box acceleration (ABA) measurements that are found using vertical accelerations of the train [17]. A fault detection framework called Track line multi-target defect detection network (TLMDDNet) is introduced based on You Only Look Once (YOLO)v3 [18]. To optimize and reduce time complexity, they have modified their fault detection framework named Dense Connection based TLMDDNet (DC-TLMDDNet). However, the framework consists of a 10 convolution layer backbone network with high time and spatial complexity of 28.4 Billion FLOPS and 56.3 MB, respectively. Another deep learning framework is proposed called FaultNet which involves the detection of faults that are restricted to rail valves only [19]. TrackNet is a neural network framework based on ResNet/DenseNet which solely detects the presence of cracks [20]. A two-stage pipeline involving localization and a deep CNN model was used to detect different types of faults in [21]. A high precision line-structured laser detection scheme is also proposed but it is unsuitable for real-time fault detection and may be costly to set up at multiple locations [22]. Though there are several computer vision, laser-based and acoustic-based systems developed, a system for fault detection is required which is simple to use, cost-effective, more robust, and reduced time complexity. To the best of one's knowledge, a multi-robot system for large-scale fault detection for railway tracks is not developed yet. This paper aims to implement an energy adaptive communication protocol called LEACH [23] to enable IoT and seamlessly transfer data between the wireless robots and receiving mobile stations. However, due to the cost limitations of this project, this implementation is restricted to MATLAB simulations to test the effectiveness of a network based on throughput, the number of clusters, and the number of robot nodes alive. It is important to note that the paper does not consider the security aspects of the energy adaptive wireless sensor network. Most security frameworks are created at the application layer. However, since LEACH is a TDMA-based MAC protocol that belongs to the data-link layer, an energy-efficient link-layer security framework needs to be established. There are several link-layer architectures which include but are not limited to SPINS [26], TinySec [27], SenSec [28], MiniSec [29], and FlexiSec [30]. FlexiSec is a flexible link-layer security architecture that claims to outperform the other algorithms by comparing it to those based on computational, storage, and energy resource overheads [31]. More research may be pursued in utilizing the FlexiSec security framework for the LEACH protocol.

3 Proposed multi-robot system

The Indian railway is one of the largest networks in the world spanning a track length of approximately 67,000 km. A multi-robot system is proposed to establish a reliable distributed system that allows now to monitor the status and location of bots that are deployed. Hence, providing the flexibility to a random deployment of bots capable of identifying surface defects on railway tracks by communicating through a well-established routing mechanism. This mechanism is implemented using the LEACH protocol. Unlike multiple frameworks for each robot that makes use of an IoT server, a multi-robot system incorporating the LEACH protocol can utilize a remote server simultaneously. In this paper, a framework is introduced for an IoT-based cloud server for communication and coordination among robots. To reduce computation time, enhance battery life, and improve bandwidth utilization, the robots are organized in pre-determined clusters. For every round, the cluster elects its cluster head due to which the load is well distributed among the bots. The bot communicates to the closest cluster head for uploading real-time data onto a cloud server. Hence, reducing communication and computational costs. Since only the cluster head can communicate to the server, the LEACH protocol uniformly distributes energy utilization for each robot in a cluster. A wireless network implementing the LEACH protocol has two types of nodes namely, advanced and normal nodes. Due to the uneven distribution of energy, advanced nodes are considered to consume more energy than a normal node by a factor α. Consider a length of the railway track required to be monitored, and then their clusters in a wireless robot network can be depicted as follows (Fig. 2).

If robot nodes are homogenous, i.e., all nodes deployed have the same initial energy, the LEACH protocol ensures that each robot node becomes a cluster head exactly once $1/P_o$ iterations, where P_o refers to the optimal probability that a robot node becomes a cluster head. The remaining nodes not elected as a cluster head belong to set G, and the probability of a node in G to be elected as a cluster head increases steadily after every iteration. The process of selecting a cluster head begins by randomly selecting a number in the range [0, 1]. A node $g \varepsilon G$ becomes a cluster head based on the following condition [22]:

$$CH = \begin{cases} Cluster\ head\ K, & t < T(g) \\ Robot\ node\ i \in G, & t \geq T(g) \end{cases} \tag{1}$$

Here, $T(g)$ is defined as

$$T(g) = \frac{P_o}{1 - P_o \cdot (i \cdot \mathrm{mod}(1/P_o))} \tag{2}$$

where i is the current iteration number.

3.1 LEACH protocol: optimal clustering

Optimal clustering allows uniform distribution of energy degradation for all robots, thereby minimizing total energy consumption by the entire multi-robot system. The energy dissipation model is described in Fig. 3 [23]. The energy released by each robot system in a cluster of area YxY, transmitting data packets of size L over a distance D, is given by

$$E_{\text{transmitter}} = \begin{cases} L.E_d + L \cdot \in_{fs} \cdot D^2, & D \leq D_0 \\ L.E_d + L \cdot \in_{mp} \cdot D^4, & D > D_0 \end{cases} \tag{3}$$

where E_d is the dissipated energy per bit of a transmitter circuit on a node. ε_{fs} and ε_{mp} are constants whose values

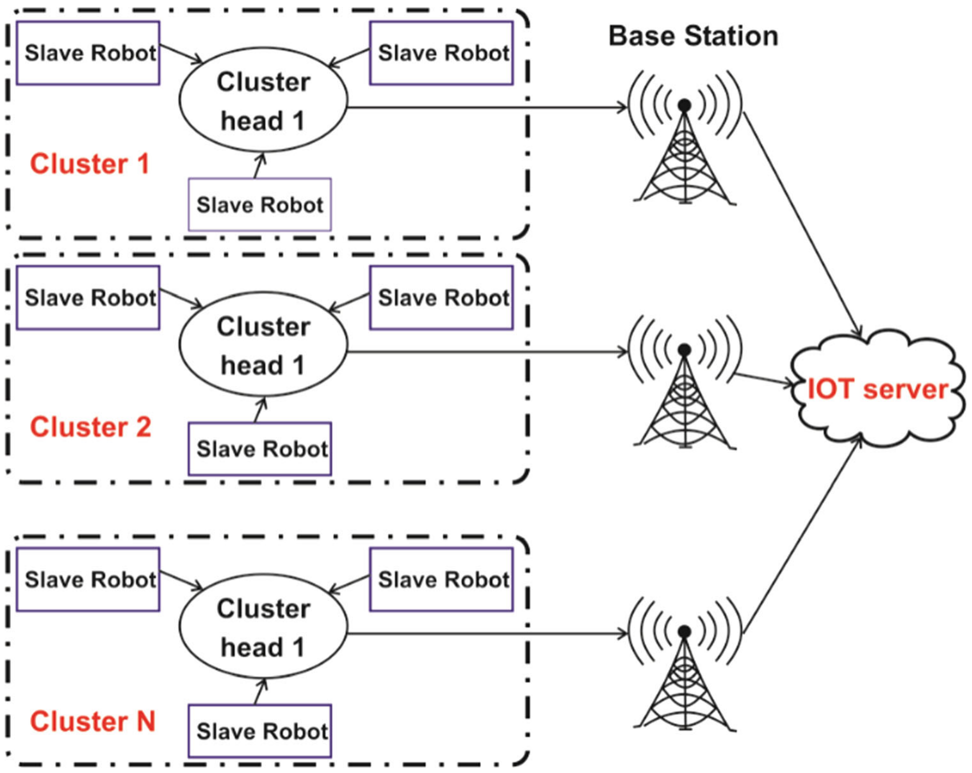


Fig. 2 Framework for a multi-robot wireless network system

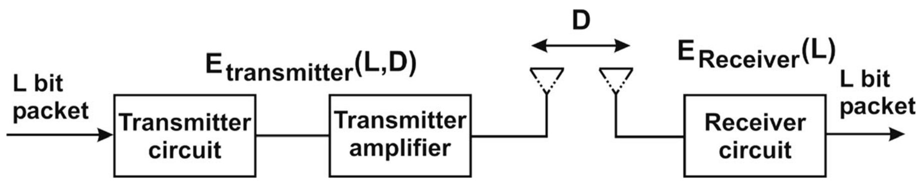


Fig. 3 First-order radio energy model [23]

depend on the type transmitter amplifier used in the circuit. D is the distance between a node and a cluster head and $D_0 = \sqrt{\varepsilon_{fs}/\varepsilon_{mp}}$. Assuming that distance $D \le D_0$, then energy utilized by a cluster head for one iteration is given as follows:

$$E_{\text{clusterhead}} = (n/C - 1)L.E_d + \frac{n}{C}L.E_{\text{data}} + L.E_d + L \cdot \varepsilon_{fs} \cdot D^2 \tag{4}$$

where n is the number of nodes in a cluster and C is the number of clusters. The energy dissipated by a robot non-cluster head is given as follows:

$$E_{\text{non-clusterhead}} = L.E_d + L \cdot \varepsilon_{fs} \cdot d_{NCH}^2 \tag{5}$$

Finally, energy utilized by a cluster can be approximated as follows:

$$E_{\text{total}} = E_{\text{clusterhead}} + \frac{n}{C} \cdot E_{\text{non-clusterhead}} \tag{6}$$

Hence, E_{total} is

$$E_{\text{total}} = L\left[2nE_d + nE_{\text{data}} + \varepsilon_{fs}\left(C \cdot d_{NCH}^2 + nd_{CHS}^2\right)\right] \tag{7}$$

where d_{NCH} is the average distance between cluster member and cluster head robot, and d_{CHS} is the average distance between the cluster head and sink. By differentiating (7) with respect to C and equating to zero, one can get the optimal number of clusters CH_{opt}.

$$CH_{\text{opt}} = \sqrt{\frac{n}{2\pi}}\frac{Y}{d_{CHS}} = \sqrt{\frac{n}{2\pi}}\frac{2}{0.765} \tag{8}$$

$$d_{CHS} = 0.765\frac{Y}{2} \tag{9}$$

where $n = 100$ nodes, $M = 1000$ m, and the optimal number of clusters CH_{opt} is six. The corresponding results of a MATLAB implementation of the proposed multi-robot system are presented in Sect. 6.

4 Proposed system of detection for rail surface defects

The main objective of this paper is to design a multi-robot system that uses multiple sensor data and computer vision for monitoring and detection of rail surface defects. Each robot node uploads sensor data continuously to a remote server, which allows now to monitor the condition of railway tracks remotely. Furthermore, the RPi-cam uses OpenCV software and deep learning to help classify the type of defect, which is notified to concerned authorities via SMS and a web server. The server also pinpoints the location of the defect on a map that may be detected so that maintenance authorities can initiate necessary action to rectify the same. Most importantly, for the entire system to run, stable internet connectivity is required for seamless data transfer between the Raspberry Pi and the cloud storage server. A general framework of the system is illustrated in Fig. 4.

4.1 Robot design

Each robot consists of a Raspberry-pi 3 microcontroller capable of sending real-time data to an internet server. The Raspberry Pi is intended to automatically sync the condition of rail surfaces through sensor outputs, GPS data, and images captured. Four ultrasonic sensors are fixed to detect faults above and on either side of a railway track surface as shown in Fig. 5. Each robot node has two DC motors for movement, which is coordinated to mimic each other through an RF communication module. Thereby, a master–slave mechanism is set up between the two robots designed in this paper. Furthermore, a Pi camera is interfaced to the Raspberry Pi, which performs video and image processing to help detect faults. The rail surface detection and

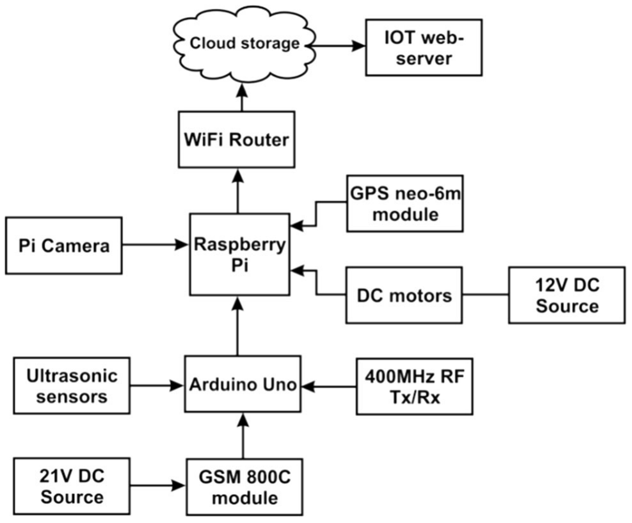


Fig. 4 The general framework of the proposed system

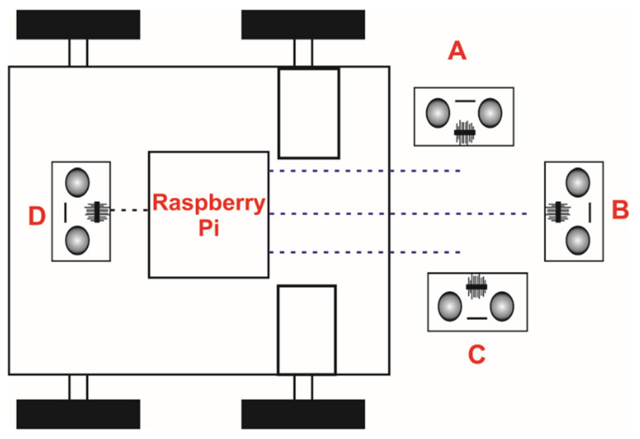


Fig. 5 Ultrasonic sensor placement

classification system using image processing are outlined in Sect. 5. Lastly, an IoT web server is created to relay the position, status, and extent of damage present in railway tracks.

4.2 Overall working mechanism

The general working of the proposed system for structurally monitoring, detecting, and classifying surface defects is described using a flowchart as shown in Fig. 6. The workflow applies to each robot manually deployed on a railway track. This system can classify four types of rail surface defects. It includes multi-directional fractures, corrosive rust on steel squats, and rail corrugations. The entire system functionality can be subdivided into three parts A, B, and C to intimate authorized persons to take appropriate action based on the type of defect found. Part A consists of three ultrasonic sensors namely, A, B, and C as positioned in Fig. 5. These sensors iteratively calculate the distance from the running and two vertical surfaces of a railway track. If any of the sensors detect distances greater than a threshold value (preset value based on a distance measurement between the sensor and the railway track surface), the 2 DC motors are programmed to halt for a delay of 10 s during which among several functionalities, an image is captured using the Raspberry Pi camera module.

Consequently, image-processing techniques are employed as will be described in the next section for further validation. During this stoppage time of the robot, authorities concerned are notified through a text alert that a structural defect is found. The text message includes an HTTP link of a webserver where additional information regarding the type of defect and its precise timestamp can be found. In addition, as depicted in Sect. 5, railway authorities can view the crack remotely by downloading the image from the website. Furthermore, a web browser pinpointing the exact GPS location of a structural defect is displayed onto a computer screen. Hence, allowing maintenance personnel to arrive at the precise location with necessary tools for repair, maintenance, or restoration work that may be required. Part B involves two ultrasonic sensors B and D whose distance measurement inputs are compared continuously. If the distances are unequal, the previously described alert mechanism is followed and authorities are intimated that rail corrugations are detected. Finally, the third part is solely based on real-time video processing of railway tracks to detect running surface fractures and rusts. The video processing is performed by extracting a 100 frames per second, where every frame is processed using the same detection system as described for an image in the following section.

5 Rail surface detection system using image processing

5.1 Fracture detection

The first part of the system as described in the previous section involves further validation of running surface fracture and crack detection through image processing. Key steps outlined in Fig. 7 are image acquisition, pre-processing, feature extraction, classification, and information storage.

During image acquisition, a noisy image is obtained due to several external factors that include unsuitable lighting conditions, vibrating robot body due to uneven terrain, or interference from foreign particle occlusion. Thus, the image pre-processing removes noise that may affect the efficiency of fault detection. After the image is converted from BGR to grayscale, A 7×7 median filter is applied which sorts the pixel values in the ascending order for each window and the median value replaces the center pixel. From Fig. 8, one can observe that the morphological gradient gives an outline of the fracture in the image by taking the difference between dilation and erosion of the resultant image. Next, to retain only the horizontal edges of cracks, a horizontal Sobel filter mask shown in Fig. 9 is applied.

Furthermore, to eliminate resembling noise pixels over the entire portion of an image, a fast non-local means denoising algorithm is incorporated [24]. Figure 9 shows the implementation of this algorithm. The image then undergoes morphological processing which utilizes a 2×2 rectangular structuring element to perform closing followed by opening operations. The opening operation allows removing background noise, if any, by performing erosion followed by dilation. While closing operation removed foreground noise by dilation followed by erosion of the image. For the next stage, key feature points of the

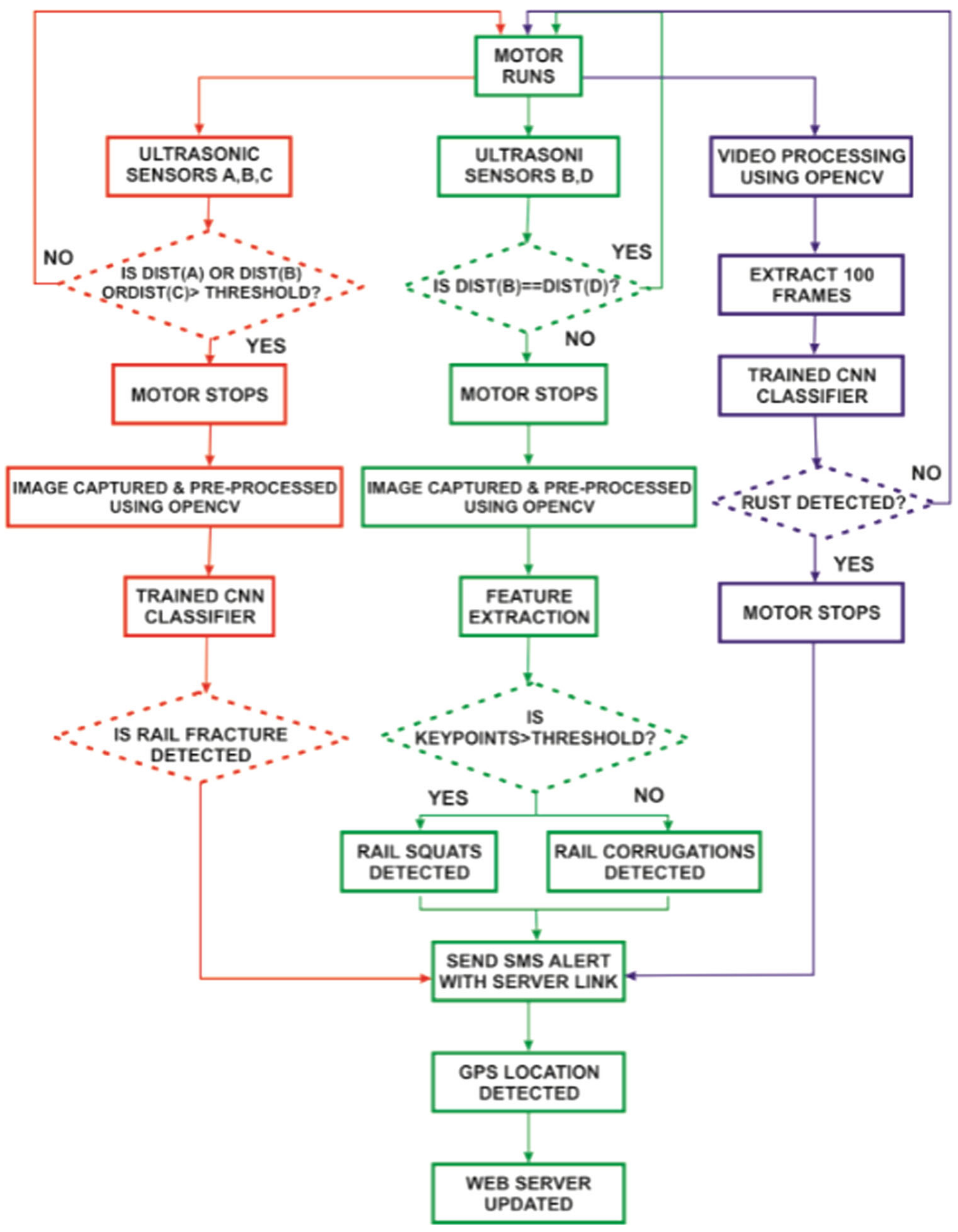


Fig. 6 The overall working of the proposed system

crack from the image are extracted using the Speeded-up robust features (SURF) algorithm [25]. This algorithm allows the operation to automatically detect feature points of distinctive objects in an image that may vary in terms of transformations, scale, rotation, or abrupt changes in pixel intensities. Finally, a count for the number of key points is established based on which a decision is made whether a multi-directional fracture is detected. This classification is performed by training a CNN classifier whose results are described in Sect. 5. Finally, the processed image is updated to the server for remote visualization of the railway line fracture. The same process outlined in Fig. 7 is followed for the detection of squats.

5.2 Detection of rust

From Fig. 10, this system utilizes an image pre-processing stage, which is different from the previous system of detection of railway line fractures. After the captured image is median filtered, it is converted into the HSV (Hue, saturation, and value) color space.

Hue value is the most accurate representation of different colors within range [0, 2π] degrees. For the detection of rust, which is predominantly red in color, the H measure is tuned accordingly, while S and V values are kept constant for all intervals. The color space conversion is performed since Hue is independent at a value, i.e., light intensity. Hence, it allows performing feature detection

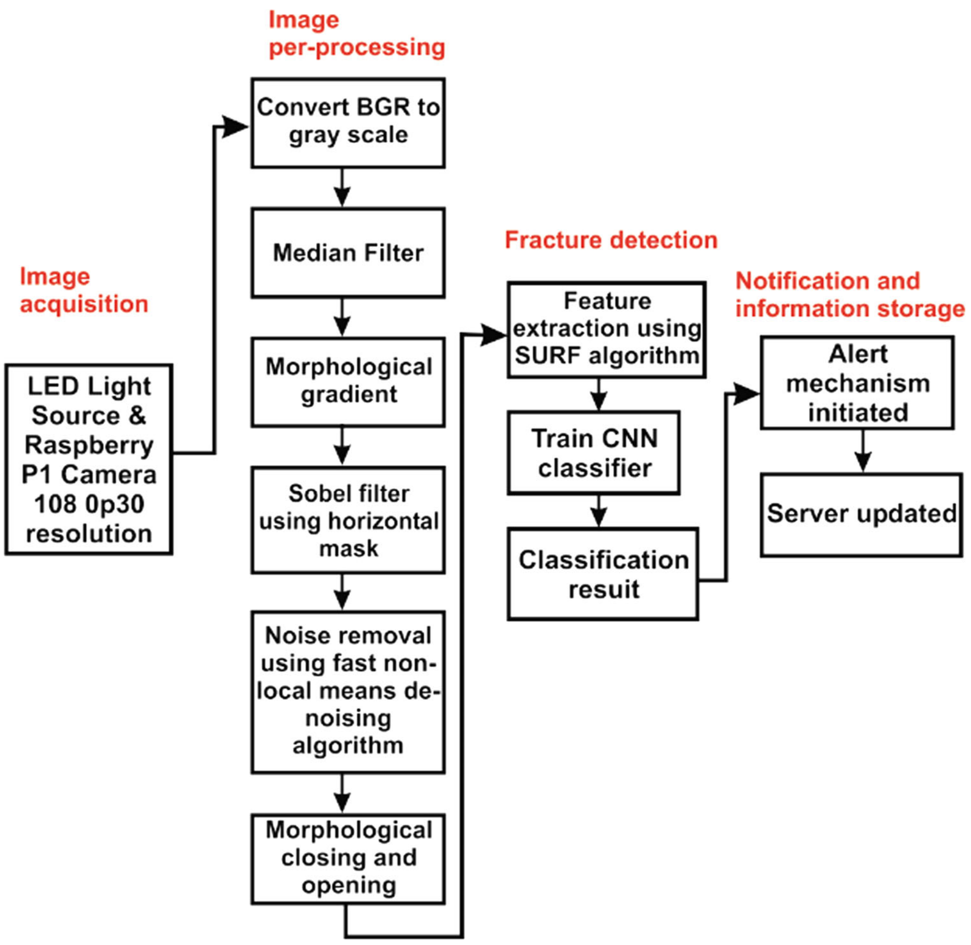


Image per-processing
Convert BGR to gray scale
Median Filter
Morphological gradient
Sobel filter using horizontal mask
Noise removal using fast non-local means de-noising algorithm
Morphological closing and opening
Image acquisition
LED Light Source & Raspberry P1 Camera 108 0p30 resolution
Fracture detection
Feature extraction using SURF algorithm
Train CNN classifier
Classification resuit
Notification and information storage
Alert mechanism initiated
Server updated

Fig. 7 Rail surface fracture detection system

based on color and in the presence of poor or excess lighting conditions. The conversion of RGB to HSV color space is as follows:

$$V = \max(R, G, B) \tag{10}$$

$$S = \frac{\max(R, G, B) - \min(R, G, B)}{V} \tag{11}$$

$$H = \begin{cases} \cos^{-1}\left[\dfrac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right] & B \le G \\ 2\pi - \cos^{-1}\left[\dfrac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right] & B > G \end{cases} \tag{12}$$

From Fig. 11, one can clearly observe that the rusted parts of the railway track are detected (in white) after a mask is convolved with the image. The mask is a 1 × 1 matrix created within a specific HSV boundary to eliminate different shades of red. Next, the image undergoes a morphological closing operation that helps smooth the transition between non-rust and rusted areas of the image. The resultant image is a binary image whose white pixels are counted. Any image with white pixels greater than 0.5% is classified as rust. Finally, the classification of the processed image is updated to the IoT server.

6 Experimental discussion and results

6.1 System implementation

The hardware system of implementation consists of a two-robot system that consists of a master and a slave robot. The slave robot is programmed and designed to mimic the master robot while it runs over a track to monitor and detect structural defects. RF communication protocol is established between them for the coordinated robot movement. The same has been extended to simulate ZigBee communication for a 100 node multi-robot system using the LEACH protocol on MATLAB. Hence, allowing efficient path coordination improved throughput and optimized energy utilization by transmitter circuits. The proposed system described in Sect. 3 consists of a 433 MHz RF communication module, GSM900 module for SMS

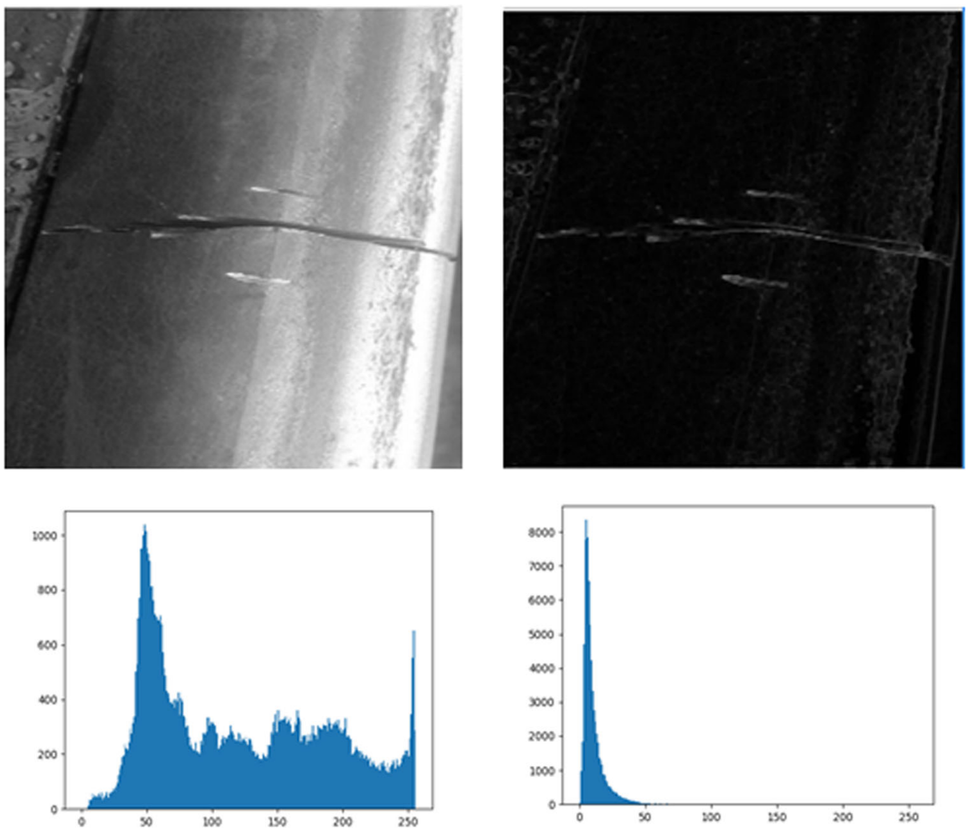


Fig. 8 Morphological gradient (right) of a rail surface crack (left)

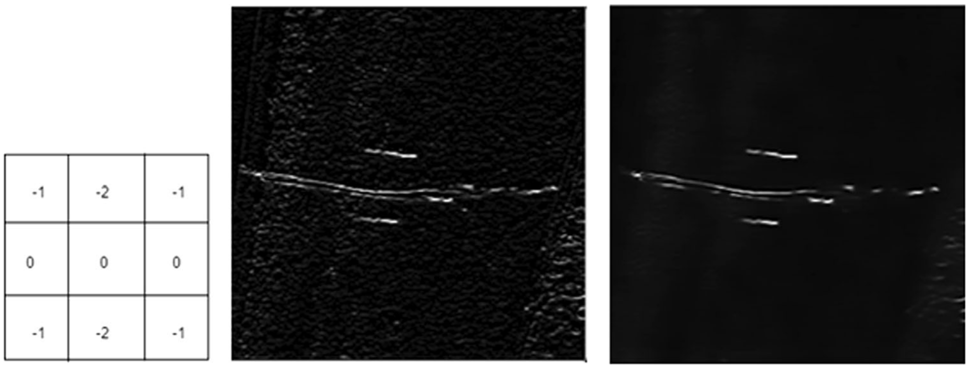


Fig. 9 Sobel filtered (center) image using a horizontal kernel (left). Image de-noised (right) using fast non-local means de-noising algorithm

alerts, a Raspberry Pi camera mounted on a servo motor, HC-SR04 ultrasonic sensors to detect cracks, corrugations, and squats, neo-6 M, GPS module, L293N motor driver IC, 12 V DC motors powered by a 12 V rechargeable battery and an Arduino UNO, all of which interfaced to a Raspberry Pi 3 Model B microcontroller. The fault detection system prototype based on computer vision and ultrasonic sensor technology is shown in Fig. 12.

6.2 Simulation of the proposed wireless multi-robot system

A wireless sensor network containing 100 nodes as shown in Fig. 13 is simulated in a 1 km × 1 km space. The colored dots represent alive nodes, circles denote dead nodes, + are advanced nodes, and the x marked at *(500,500)* is the sink. The node positions are randomly selected within the vector space.

The first-order radio characteristics used for these simulations are summarized in Table 1 [22]. The simulation was performed on MATLAB R2016a.

The network was simulated, and the outputs were compared for all nodes (normal nodes and advanced nodes) based on the fraction of advanced nodes present (m), and a factor of the energy difference between normal and advanced nodes (α).

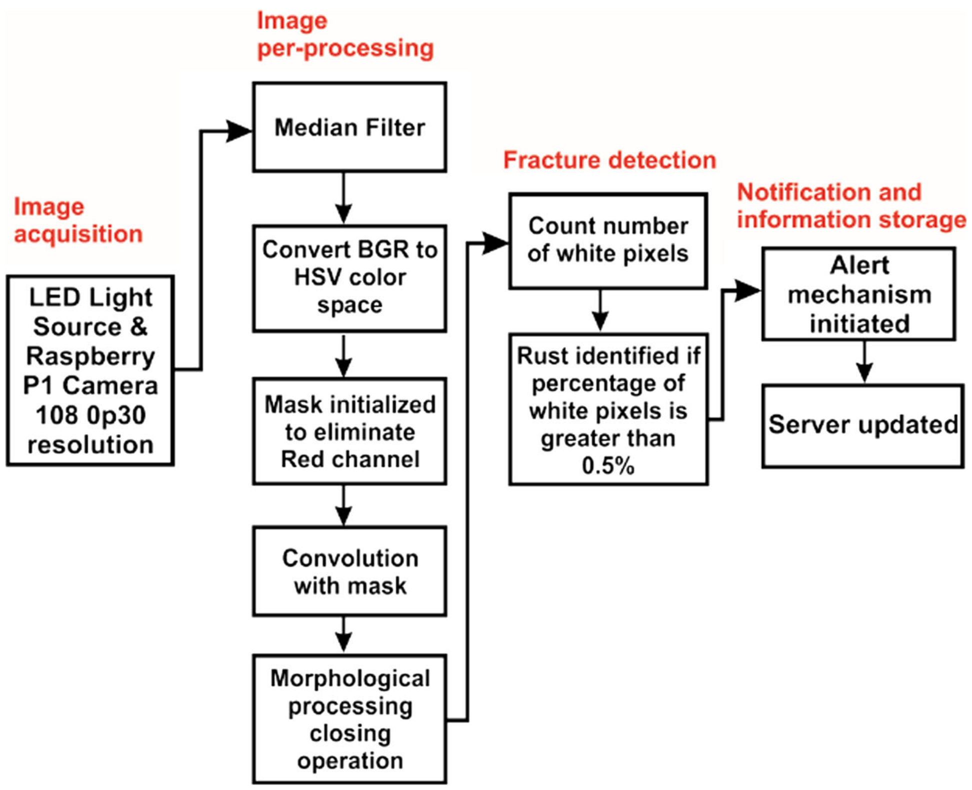


Image per-processing
Median Filter
Convert BGR to HSV color space
Mask initialized to eliminate Red channel
Convolution with mask
Morphological processing closing operation
Image acquisition
LED Light Source & Raspberry P1 Camera 108 0p30 resolution
Fracture detection
Count number of white pixels
Rust identified if percentage of white pixels is greater than 0.5%
Notification and information storage
Alert mechanism initiated
Server updated

Fig. 10 Rail surface rust detection system

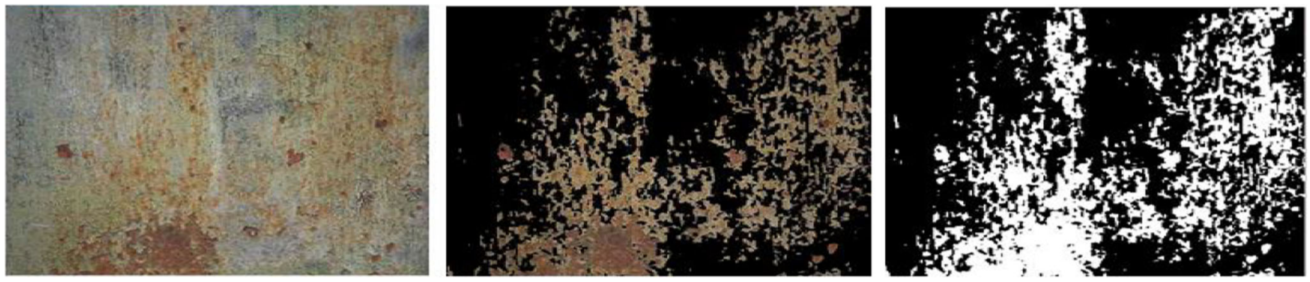


Fig. 11 A rusted portion (center) detected from an image of a corroded metal (left) with white pixels representing corrosion (right)

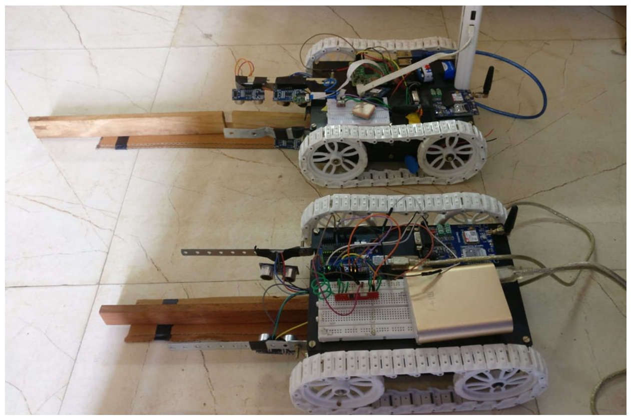


Fig. 12 Hardware prototype of the two-robot fault detection system

$$m = \frac{Number\ of\ advanced\ nodes}{Total\ number\ of\ nodes} \tag{13}$$

$$\alpha = \frac{\text{energy consumption of advanced nodes}}{\text{energy consumption of normal nodes}} \tag{14}$$

The performance measures used to evaluate these simulations are as follows:

- **The number of nodes alive**: It signifies the number of nodes that have not completely utilized their energy. Figure 14 illustrates a comparison between different values of m and α. From Fig. 14, one can clearly observe that more nodes were alive at the end of
- 5000 rounds when the energy factor is increased to 3 (α = 3). Approximately, 5 nodes were alive uniformly after the 3000th round for $\alpha = 3$ while almost no nodes were active when $\alpha = 2$ and $\alpha = 0$.

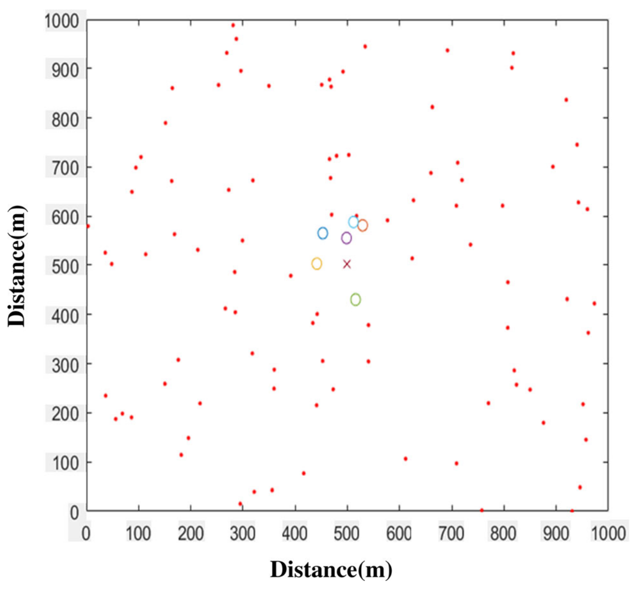


Fig. 13 The wireless sensor network containing alive and dead nodes

Table 1 First-order radio characteristics implemented

Parameter	Energy dissipated
Transmitter/Receiver circuit	$E_d = 50$ nJ/bit
Data aggregation	$E_{\text{data}} = 5$ nJ/bit/round
Transmitter amplifier If $d_{CHS} \leq D_0$	$\varepsilon_{fs} = 10$ pJ/bit/m^2
Transmitter amplifier If $d_{CHS} > D_0$	$\varepsilon_{mp} = 0.0013$ pJ/bit/m^4
Node initial energy	$E_0 = 0.5J$

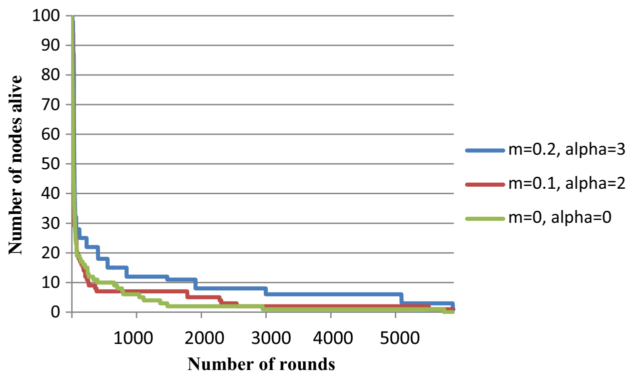


Fig. 14 Number of nodes alive for every round

- **The number of cluster heads**: It affects the number of packets transmitted by the cluster to the sink per round. Hence, the higher the number of cluster heads, the higher is the overall network throughput. The sink in the multi-robot network refers to a base station of an internet service provider.

On substituting M = 1000 m, d_{CHS} = 382.5 m from (10) and n = 100 nodes in (9), one gets the optimal number of cluster heads as $CH_{opt} = 6$. From Fig. 15, it is found that with α = 3, the network consists of six optimal cluster heads from rounds 1000 to 3000. While α = 2 and α = 0 had no cluster heads by the 4000th round, α = 3 network fared well with three cluster heads, thereby managing to extend the network life.

- **Throughput**: It refers to the rate of transmission of data packets from node to cluster head and cluster head to a base station. Figure 16 shows the throughput from node to cluster head. It is observed that, while all the three parameters set for the network had throughput initially at 180 kbps, it decreased drastically for $m = 0.1, \alpha = 2$ and $m = 0, \alpha = 0$ to zero by the 3000th round. However, the higher-order energy factor, α = 3 decreases more gradually till the 2000th round. After that, it varies from 15 to 20 kbps till the 4600th round. Maximum throughput was obtained after round 2000 for $m = 0.2, \alpha = 3$.

Figure 17 shows the throughput from the cluster head to a base station. One can observe that, while all the three parameters set for the network had initial throughputs at 17 kbps, $m = 0, \alpha = 0$ decreased to zero at round 3500 while $m = 0.1, \alpha = 2$ and $m = 0.2, \alpha = 3$ were found to reduce the throughput to zero at rounds 4000 and 4500, respectively. However, the cluster head reached maximum throughput after the 2500th round for $m = 0.2, \alpha = 3$.

The overall network throughput in Fig. 18 shows that the network should be simulated with m = 0.2 and α = 3 to maximize throughput. The corresponding throughput obtained for a node to cluster head and cluster head to the base station proves that the above condition maximizes the number of packets transmitted for every round.

To summarize, the simulation concluded that for a node incorporating radio characteristics as mentioned in Table 1, the wireless network must use m = 0.2 and α = 3 to

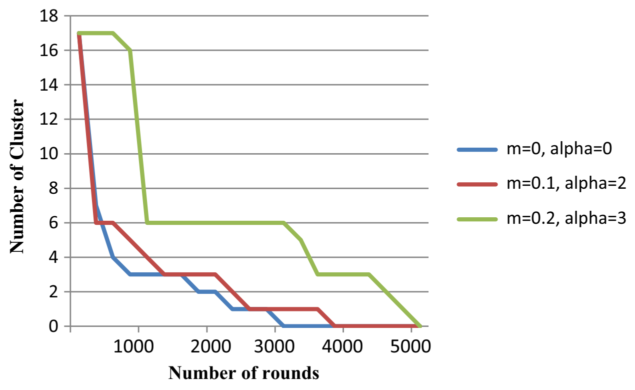


Fig. 15 Number of cluster heads per round

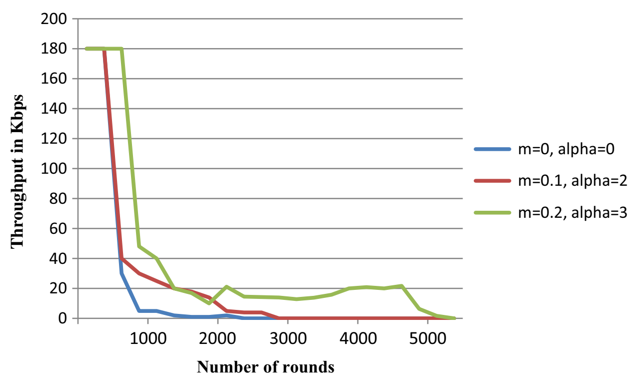


Fig. 16 Throughput for a node to cluster head

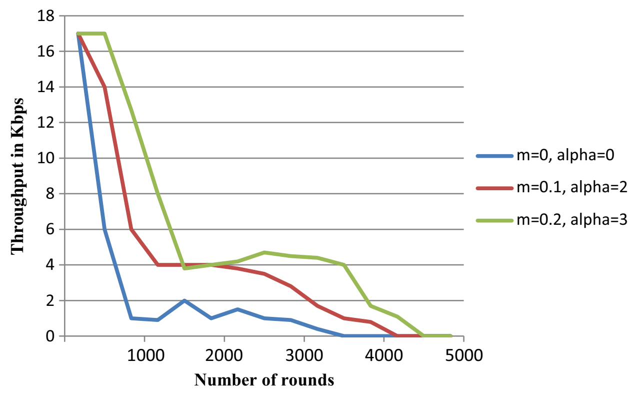


Fig. 17 Throughput for a cluster head to the base station

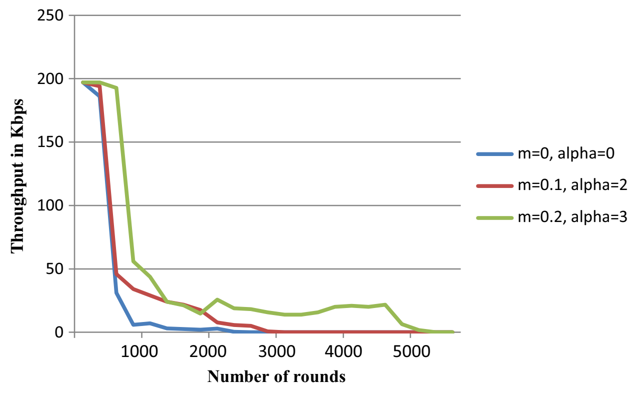


Fig. 18 Overall throughput of the entire network

maximize the number of alive nodes, network throughput, and optimize the number of cluster heads created.

6.3 Fault detection using image processing and machine learning

In this paper, OpenCV 3.3 is used for real-time detection and feature extraction of rail surface faults. For the implementation of ANN, SVM, and random forests, the scikitLearn library was used and Keras on Tensorflow was used to implement the CNN model in Python 3. The entire python implementation was run on Raspberry Pi 3. Figure 19 shows the processed image of a railway line fracture (center) and its key features (right) detected using the SURF algorithm. The extracted features were used as training features for the ANN, SVM, and random forest machine learning models. The processed image (without SURF based feature detection) was used to train the stated CNN model. The dataset used to train the machine learning models consisted of 195 images of rail surface cracks and 200 images of concrete surfaces with no visible cracks. Concrete surfaces were used since to the best of the knowledge gained, there are no readily available datasets for healthy rail surfaces.

The CNN model trained on 395 images is graphically represented in Fig. 20. The dataset can be downloaded from https://github.com/kriiyer/Dataset.git. It consists of two convolutions and two max-pooling layers for feature extraction, followed by a three-layered FCN (Fully connected layer) for classification. The input gray level image dimensions of the rail defect database are 1000×160 while the dimensions of each image in the concrete database are 227×227. For uniformity, both the images were resized to 512×256. The images are then normalized by dividing each pixel value by 255. The model was trained for 25 epochs, and the learning rate was initially set to 10^{-3} with a decay factor of 10^{-5} to avoid overfitting of data. The first CNN layer filters the input normalized images with a kernel size of 16×8, a stride of 1×1, with zero padding and relu activation function. After the first convolution, 32 2D feature maps are obtained each of size 497×249. Next, a max-pooling layer of size 3×3 is performed which downsamples the feature maps to 124×62. The second CNN layer filters the downsampled feature map with a kernel size of 4×2, a stride of 1×1, with zero passing and relu activation function. Thus, 31 2D feature maps of dimension 121×61 are obtained. The second max-pooling layer of size 4×4 downsamples the feature map to 30×15. Finally, the feature maps are fed to a three-layered FCN which has 128 nodes (input layer) and 32 nodes (hidden layer). The output layer activation function is the sigmoidal function which gives now the final classification results.

The CNN model is evaluated based on its spatial complexity and time complexity as shown in Table 2. Here, memory utilization refers to RAM space occupied by all the feature maps for every image. FLOPS refers to the number of floating-point operations per second. Since there are no open-source rail defect datasets for benchmarking, It is felt that comparisons with existing CNN-based models lie outside the scope of this paper. From Table 3, with almost negligible mean-squared error values and high accuracy, R-squared and F1 scores, the CNN model was

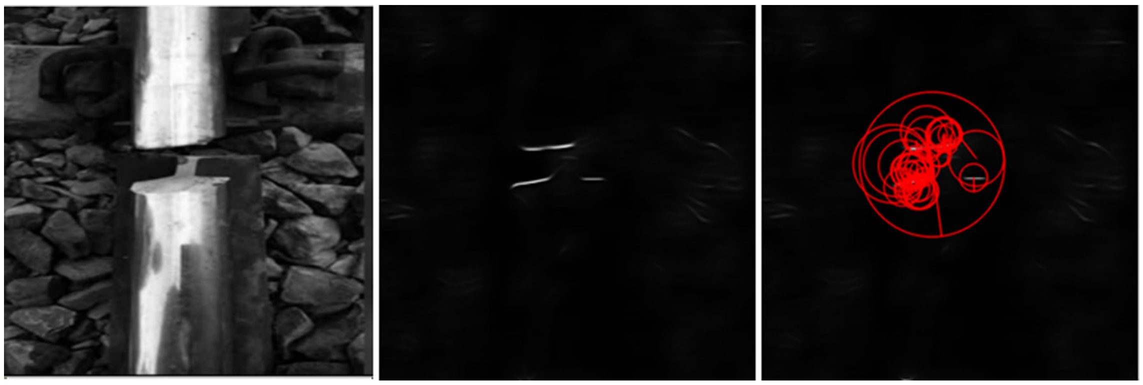


Fig. 19 Processed output depicting railway fracture (center) found in the input image (left) with key feature point extraction using SURF (right)

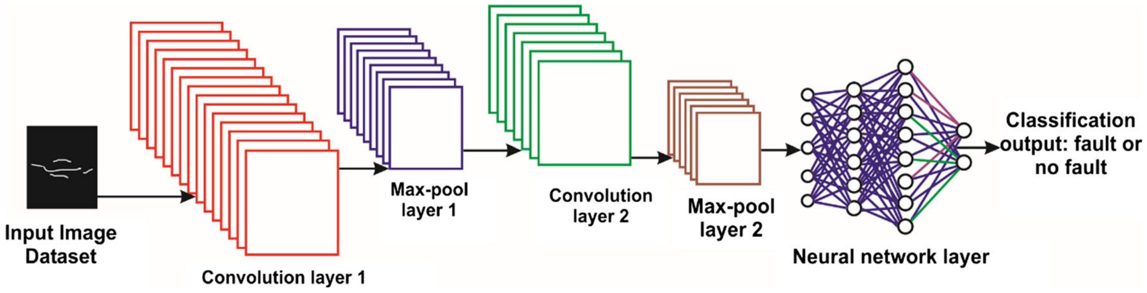


Fig. 20 Proposed model of CNN

Table 2 Performance metrics based on the crack detection dataset

Metrics	Value
Accuracy	0.9857
Number of parameters	1.8 Million
Memory utilization	37.2 MB
FLOPS	508 Million

able to predict a railway fracture from a pre-processed image more accurately in comparison to the other algorithms. The CNN model was compared to an Adam optimized ANN, SVM, and random forest algorithms. Both training and validation metrics were compared as shown in Figs. 21 and 22.

The highest R-squared values during training and validation indicate that the CNN model was able to accurately describe the deviation of predicted outputs from the regression line (Threshold). Moreover, the highest F1 scores show that the model was able to predict 98.7% of classification outputs correctly during training while for validation, 97.7% of the outputs were obtained correctly. Minimized mean-squared error (MSE) obtained by the proposed model signifies that the predicted outputs were closer to the trained target outputs.

From Fig. 21, one can observe that the proposed CNN model outperformed other algorithms during the training phase. SVM was the worst-performing algorithm with the lowest figures for accuracy, R-squared value, and F1 score and maximized mean-squared error at 0.188. The random forest algorithm fared better than the ANN model, yet it was unable to perform better than CNN.

Table 3 Comparison of proposed CNN model with machine learning algorithms for classification of rail surface crack

Performance metrics	ANN		CNN		SVM		Random forests	
	Training	Validation	Training	Validation	Training	Validation	Training	Validation
Accuracy (%)	0.76063	0.76856	0.9857	0.9504	0.69321	0.57443	0.9101	0.89723
MSE	0.164	0.178	0.012	0.014	0.188	0.264	0.02	0.14
R-squared	– 0.0175	– 0.0925	0.95	0.935	– 0.018	0.099	0.874	0.733
F1 score	0.83	0.7988	0.987	0.977	0.822	0.759	0.9	0.898

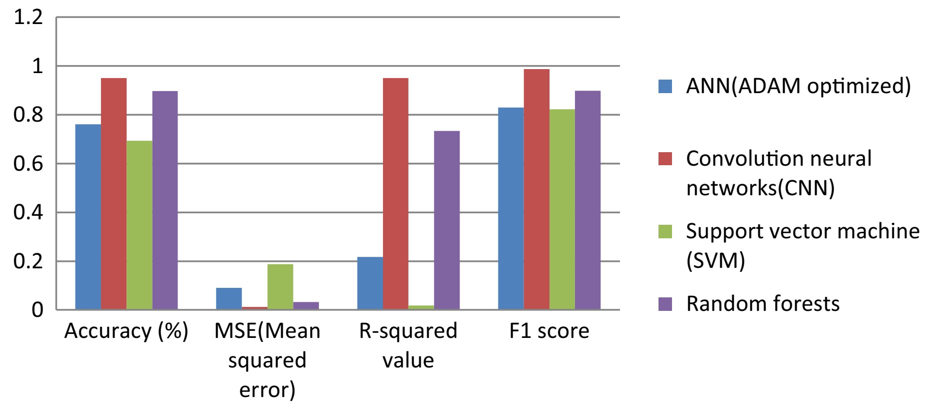


Fig. 21 Comparison of training performance metrics for crack classification

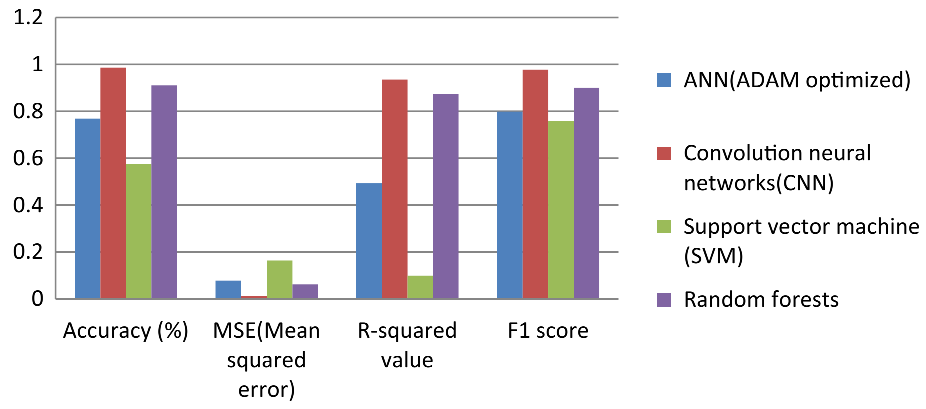


Fig. 22 Comparison of validation performance metrics for crack classification

Again, from Fig. 22, the proposed CNN model outperformed other algorithms during the validation phase followed very closely by the random forest algorithm. Once again, the SVM was the least performing algorithm. Hence, similar trends were observed during the training and validation stages.

Since ultrasonic sensors are used to detect both rail corrugations and squats, image processing was further used to validate if the captured image contained squats as shown in Fig. 23. Since there are no readily available datasets for the classification of squats using deep learning, count for the number of key points detected using the SURF algorithm is monitored. The system detected squats if the count was greater than a preset threshold value.

The rust detection system was applied to a single image and can be implemented for each frame during video processing. Since corrosion due to rust cannot be characterized by specific shapes or edges, the proposed rust detection system extracts all shades of reddish coloration in the HSV color space.

Through extensive testing by hit and trial method, it is found that the red component has two ranges in the HSV color space. The HSV range to detect reddish coloration of rust is defined for two ranges as:

$$Rust = \begin{cases} 1,(0,50,255) \le (H,S,V) \le (21,50,255) \\ \quad \text{or } (175,50,255) \le (H,S,V) \le (179,50,255) \\ 0,\text{otherwise} \end{cases} . \tag{15}$$

S and V components are fixed between [50,255] for both ranges while [0, 21] and [175,179] are Hue intervals for first and second ranges, respectively. After the image was thresholded using both HSV ranges to eliminate non-red color components, the resultant outputs were added, and binary thresholding was performed. Figure 24 (right) depicts the rusted area after binary thresholding. The binary images were then fed to a CNN model as shown in Fig. 20 and compared to three machine learning algorithms as summarized in Table 4. ANN, SVM, and random forests were trained on SURF-based features. It is important to note that the CNN model is separately trained for rust and crack detection as images of the rust database have different image dimensions and kernel sizes used to extract feature maps for rust detection. A CNN model as shown in Fig. 20 was trained using 200 images of rusted iron, out of which a 100 were positive cases and the remaining were rust-free. All the input images were resized to 256 $\times$ 256. The CNN model parameters are kept the same except for

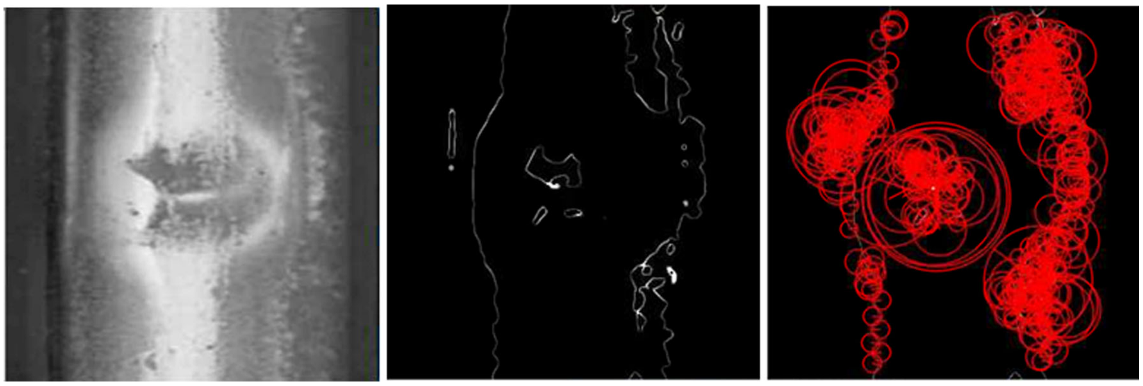


Fig. 23 Processed output depicting squats (center) found from the input image (left) with key feature point extraction using SURF (right)

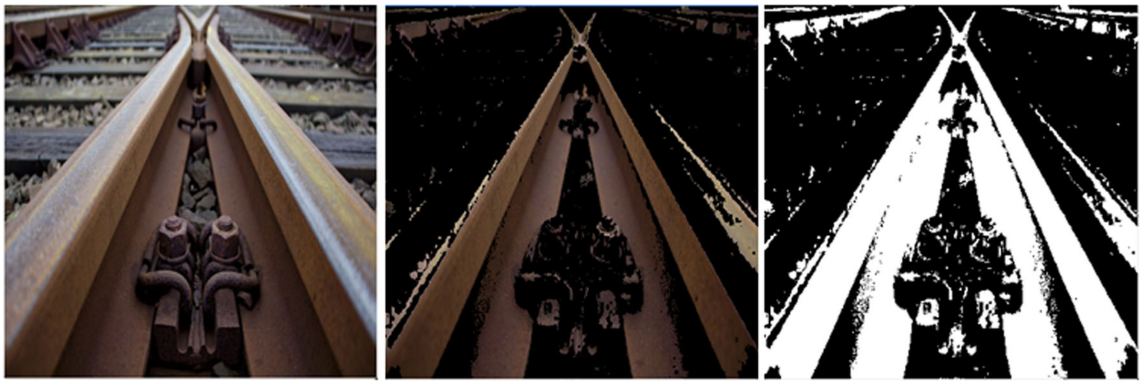


Fig. 24 Rust detection program output (center) of rusted tracks (left) with white pixels representing corrosion (right)

Table 4 Performance metrics based on the crack detection dataset

Metrics	Value
Accuracy	0.9032
Number of parameters	927 K
Memory utilization	36 MB
FLOPS	33 Million

the kernel size which is set as 4 × 4 for the 1st CNN layer and 2 × 2 for the 2nd CNN layer (Table 5).

From Table 4, the computational and spatial complexity of training the rust dataset is much lower than that of crack detection. The number of FLOPS is 15 times lower than crack detection as the number of parameters trained in the convolution layers is halved. However, the trade-off is that the accuracy is lowered.

The highest R-squared values during training and validation indicate that the CNN model was able to more accurately describe the deviation of predicted outputs from the regression line. Moreover, the highest F1 scores show that the model was able to predict 95.5% of classification outputs correctly during training while for validation, 84.1% of the outputs were obtained correctly. The above CNN model can be implemented for the classification of railway squats; however, there are no readily available datasets for the classification of squats; hence, this paper

Table 5 Comparison of proposed CNN model with machine learning algorithms for classification of rust

Performance metrics	CNN		ANN		SVM		Random forests	
	Training	Validation	Training	Validation	Training	Validation	Training	Validation
Accuracy (%)	0.9032	0.8276	0.894	0.8276	0.56	0.612	0.875	0.911
MSE	0.09	0.12	0.097	0.187	0.197	0.203	0.109	0.182
R-squared	0.699	0.309	0.599	0.171	0.144	0.05	0.673	0.298
F1 score	0.955	0.841	0.941	0.823	0.45	0.509	0.946	0.771

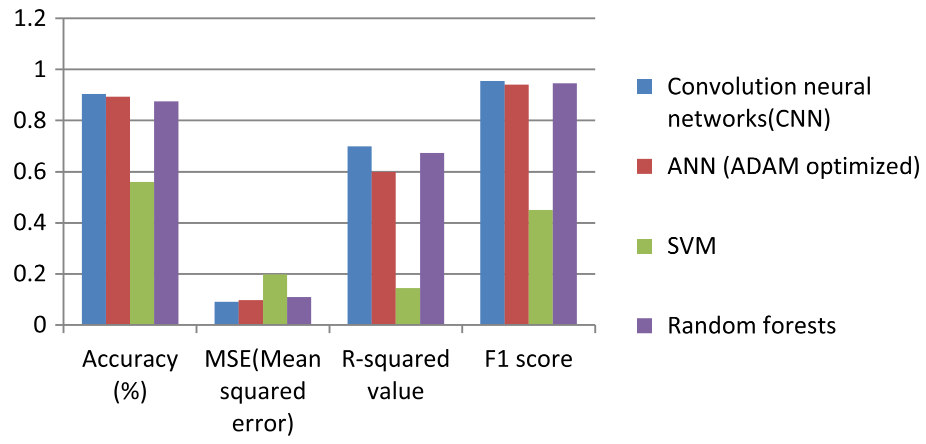


Fig. 25 Comparison of training performance metrics for rust classification

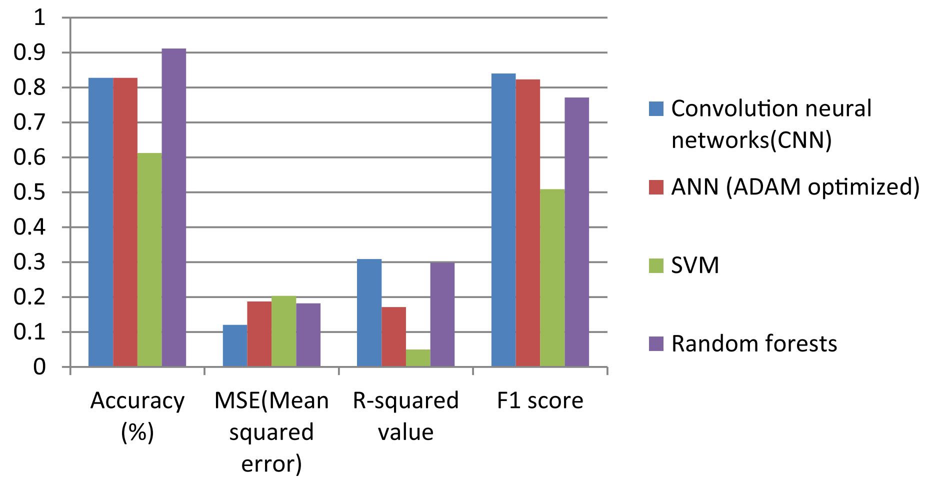
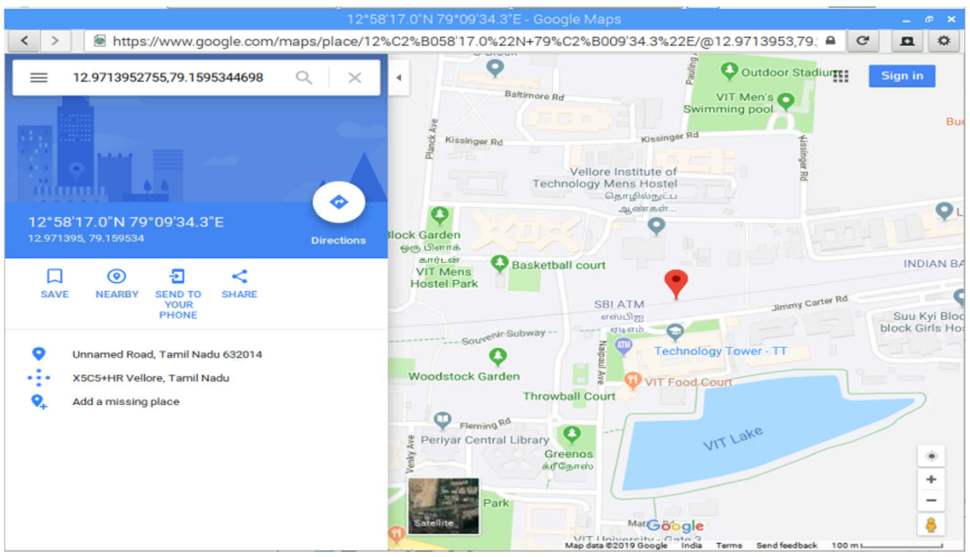


Fig. 26 Comparison of validation performance metrics for rust classification


12°58'17.0"N 79°09'34.3"E - Google Maps

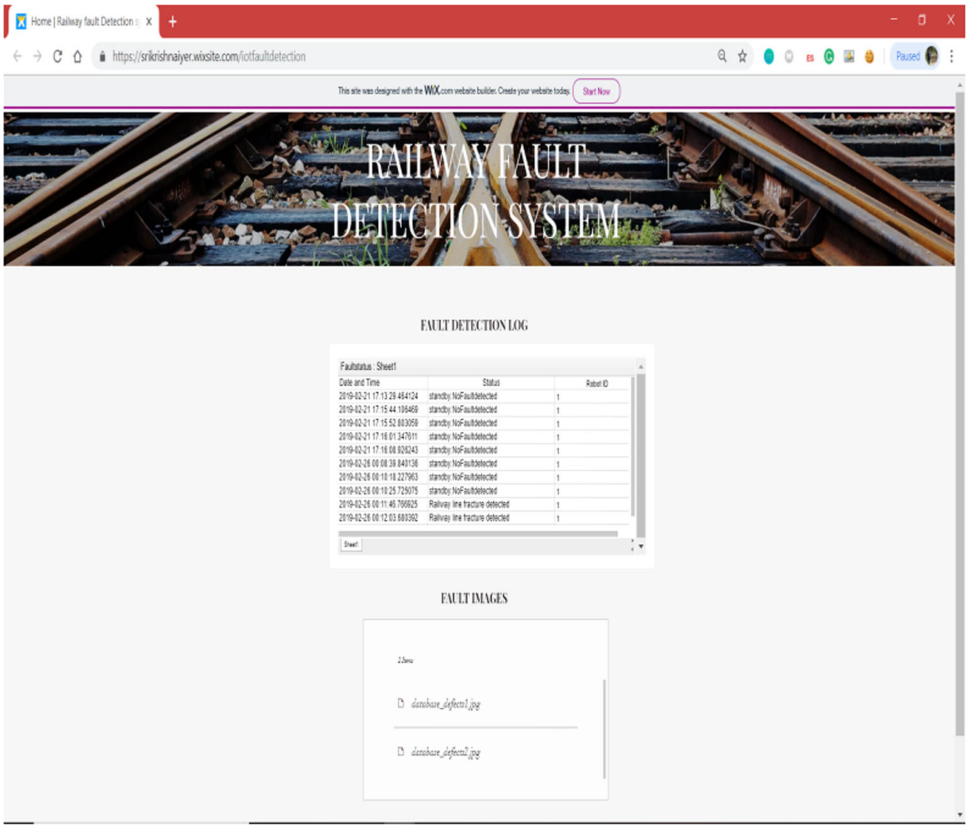
https://www.google.com/maps/place/12%C2%B058'17.0%22N+79%C2%B009'34.3%22E/@12.9713953,79.

12.9713952755,79.1595344698

12°58'17.0"N 79°09'34.3"E
12.971395, 79.159534

Unnamed Road, Tamil Nadu 632014
X5C5+HR Vellore, Tamil Nadu
Add a missing place

Fig. 27 Location of fault marked on Google maps


Home | Railway fault Detection
https://srikrishnaiyer.wixsite.com/iotfaultdetection
This site was designed with the WiX.com website builder. Create your website today.
Start Now
RAILWAY FAULT DETECTION SYSTEM
FAULT IMAGES
database_defects1.jpg
database_defects2.jpg

FAULT DETECTION LOG

Faultstatus : Sheet1

Date and Time	Status	Robot ID
2019-02-21 17:13:29.464124	standby:NoFaultdetected	1
2019-02-21 17:15:44.106468	standby:NoFaultdetected	1
2019-02-21 17:15:52.883058	standby:NoFaultdetected	1
2019-02-21 17:16:01.347611	standby:NoFaultdetected	1
2019-02-21 17:16:08.926243	standby:NoFaultdetected	1
2019-02-26 00:08:39.840136	standby:NoFaultdetected	1
2019-02-26 00:10:18.227963	standby:NoFaultdetected	1
2019-02-26 00:10:25.725075	standby:NoFaultdetected	1
2019-02-26 00:11:46.766825	Railway line fracture detected	1
2019-02-26 00:12:03.680392	Railway line fracture detected	1

Fig. 28 Railway fault detection website

devised a simplified method to count the number of key points detected on a pre-processed image using the SURF algorithm.

From Fig. 25, one can observe that the proposed CNN model fared slightly better than ANN and random forests during the training phase. SVM was the worst-performing algorithm with the lowest values of accuracy, R-squared value, and F1 score and maximized mean-squared error at 0.197. Both ANN and random forest performed equally well in terms of accuracy, MSE, and F1 score while the random forests was a clear winner based on a higher R-squared value which signifies that the random forests could more accurately fit new data with the least deviation from a regression line.

As shown in Fig. 26, the random forest algorithm had a higher validation accuracy than the stated CNN model; however, the CNN model performed slightly better in terms of minimizing MSE, maximizing R-squared value, and F1 score. Since accuracy cannot be the only determining factor for a model's performance, it can be concluded that the CNN model would be more capable of fitting any new data with minimized variance and classify outputs with the least mean-squared error. Again, the SVM was the least performing algorithm based on the given dataset.

6.4 Alert mechanism

To relay information about the fault detected, an SMS alert is sent to a mobile phone. Also, when a fault is detected, the system is designed to trigger open a web browser with the GPS location pin marked onto Google maps as shown in Fig. 27, thus allowing immediate attention and intervention of maintenance personals. Images captured and processed by the Raspberry Pi are sent to a Dropbox^{TM} account (a cloud storage service). The images can be viewed and downloaded from a website linked to the cloud storage hence allowing railway personals to easily determine the extent of the damage. A live Google spreadsheet logs in the status of real-time fault detection by both robots which can also be viewed on the website as shown in Fig. 28.

7 Conclusion

The paper aimed to propose a multi-robot system capable of monitoring and detecting surface defects on railway tracks which include fractures, squats, corrugations, and rust. A 100-node multi-sensor environment was simulated using the LEACH protocol on MATLAB. Conclusive results showed that a network must incorporate a higher energy factor, α = 3, i.e., advanced nodes with greater energy utilization than normal nodes to effectively optimize throughput and increase the lifetime of every node. Thus, a multi-robot system can be realized to monitor one of the world's largest rail networks. A two-robot hardware prototype was implemented which utilized both ultrasonic sensor inputs and image processing to accurately classify faults detected and relay the information to a remote web server. The classification of rail surface cracks and rust was implemented using CNN. The results were compared to four machine learning algorithms namely, CNN, ANN, random forests, and SVM. It was found that the CNN model outperformed all the algorithms under review, all of which were trained and validated using the same dataset. Hence, the proposed system helped to eliminate the need for visual inspection as the automated alert mechanism allowed both visual and location-based, real-time tracking of fault detection of railway lines.

Compliance with ethical standard

Conflict of interest The authors declare that they have no conflict of interests.

References

1. ARTC (Australian Rail Track Corporation) (2006) Rail defects handbook some rail defects, their characteristics, causes and control. Eng Pract Manual, (A):1–68
2. Singh M, Singh S, Jaiswal J, Hempshall J (2006) Autonomous rail track inspection using vision based system. In: 2006 IEEE International conference on computational intelligence for homeland security and personal safety, Alexandria, VA, pp 56–59
3. Zhan Dong, Yu L, Xiao Jian, Chen T-L (2014) Study on track wear cross-section measurement utilizing laser-photogrammetric technique. J China Railw Soc 36:32–37
4. Gao Y, Wang P, Wang H, Shi Y (2016) Review of railway rail defect non-destructive testing and monitoring. Yi Qi Yi Biao Xue Bao/Chin J Sci Instrument 37:1763–1780
5. Xing L (2008) Research on defect characteristics and classification of higher speed rails. China Academy of Railway Sciences, Beijing
6. Singh AK et al (2019) Vision based rail track extraction and monitoring through drone imagery. ICT Express. 5(4):250–255
7. Min Y, Xiao B, Dang J et al (2018) Real time detection system for rail surface defects based on machine vision. J Image Video Proc. 2018:3
8. Shah AA et al (2020) Real time identification of railway track surface faults using canny edge detector and 2D discrete wavelet transform. Ann Emerging Technol Comput (AETiC) 4(2):53–60
9. Malekjafarian A et al (2019) Railway track monitoring using train measurements: an experimental case study. Appl Sci 9(22):4859
10. Kostryzhev AG, Davis CL, Roberts C (2013) Detection of crack growth in rail steel using acoustic emission. Ironmak Steelmak 40(2):98–102
11. Liu X-Z, Ni Y-Q, Wu W-L, Pei Y-F, Hou YH, Qin D-Y (2015) AET-based Pattern Recognition Technique for Rail Defect Detection. https://doi.org/10.12783/shm2015/252
12. Feng N, Zhang X, Zou Z, Wang Y, Yi S (2015) Rail health monitoring using acoustic emission technique based on NMF and RVM. In: 2015 IEEE international instrumentation and measurement technology conference (I2MTC) proceedings, Pisa, pp 699–704
13. Li Qingyong, Ren Shengwei (2012) A real-time visual inspection system for discrete surface defects of rail heads. IEEE Trans Instrument Meas 61(8):2189–2199
14. Li Q, Ren S (2012) A visual detection system for rail surface defects. IEEE Trans Syst Man Cybern Part C (Appl Rev) 42(6):1531–1542
15. Liu Scarlett, Wang Quandong, Luo Yiping (2019) A review of applications of visual inspection technology based on image processing in the railway industry. Transp Safety Environ 1(3):185–204
16. Mariani S, Nguyen T, Phillips RR, Kijanka P, LanzadiScalea F, Staszewski WJ, Fateh M, Carr G, Mariani S (2013) Noncontact ultrasonic guided wave inspection of rails. Struct Health Monitoring 12(5–6):539–548
17. Molodova Maria, Li Zili, Núñez Alfredo, Dollevoet Rolf (2014) Automatic detection of squats in railway infrastructure. IEEE Trans Intell Transp Syst 15(5):1980–1990
18. Wei X et al (2020) Multi-target defect identification for railway track line based on image processing and improved YOLOv3 Model. IEEE Access 8:61973–61988
19. Pahwa RS et al (2019) FaultNet: faulty rail-valves detection using deep learning and computer vision. In: 2019 IEEE intelligent transportation systems conference (ITSC). IEEE
20. James A et al (2018) Tracknet-a deep learning based fault detection for railway track inspection. In: 2018 International conference on intelligent rail transportation (ICIRT). IEEE
21. Yanan S, Hui Z, Li L, Hang Z (2018) Rail surface defect detection method based on YOLOv3 deep learning networks. In: 2018 Chinese Automation Congress (CAC), Xi'an, China, pp 1563–1568, https://doi.org/10.1109/CAC.2018.8623082
22. Cao Xiaohui, Xie Wen, Ahmed Siddiqui, Li Cunrong (2020) Defect detection method for rail surface based on line-structured light. Measurement 159:107771. https://doi.org/10.1016/j.measurement.2020.107771
23. Heinzelman WR, Chandrakasan A, Balakrishnan H (2000) Energy-efficient communication protocol for wireless sensor networks. In: Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, Maui, HI, USA, pp. 10 pp. vol 2, pp 1–10
24. Budes Antoni, Coll Bartomeu, Morel Jean-Michel (2011) Non-local means denoising. Image Process Line 1:208–212
25. Bay H, Tuytelaars T, Van Gool L (2006) SURF: speeded up robust features. In: Leonardis A, Bischof H, Pinz A (eds) Computer Vision—ECCV 2006. ECCV 2006. Lecture Notes in Computer Science, vol 3951. Springer, Berlin. https://doi.org/10.1007/11744023_32
26. Perrig A, Szewczyk R, Tygar JD, Wen V, Culler DE (2002) SPINS: security protocols for sensor networks. Wireless Netw 8(5):521–534, Kluwer Academic Publishers, MA, USA
27. Karlof C, Sastry N, Wagner D (2004) TinySec: a link layer security architecture for wireless sensor networks. In: SenSys '04: proceedings of the 2nd international conference on Embedded Networked Sensor Systems, pp 162–175, ACM, NY, USA
28. Li T, Wu H, Wang X, Bao F (2005) SenSec: sensor security framework for TinyOS. In: Proceedings of the second

international workshop on networked sensing systems, pp 145–150. San Diego, USA

29. Luk M, Mezzour G, Perrig A, Gligor V (2007) MiniSec: a secure sensor network communication architecture. In: Proceedigns of the ACM international conference on information processing in sensor networks, pp 479–488, ACM, NY
30. Jinwala D, Patel D, Dasgupta K (2012). FlexiSec: a configurable link layer security architecture for wireless sensor networks. J Inf Assurance Secur Special Issue Inf Assurance Data Security 4

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.