

Indigenous Smart Railway Track Monitoring Using Computer Vision and Sensor Fusion

Selvarani D¹, Kruthika Y K², Rohith Kumar G A², & Thanuja C²

¹Assistant Professor, Department of Information Science and Engineering
Sri Sairam College of Engineering, Bengaluru

²Final-year B.E(Information Science and Engineering), Sri Sairam College of Engineering, Bengaluru

DOI: doi.org/10.34293/iejcsa.v4i4.159

Abstract - Track recording on Indian Railways depends largely on imported, costly, proprietary inspection cars. These systems are expensive to acquire and service, difficult to adapt to India-specific track conditions, and run on infrequent, centrally scheduled cycles, so defects that develop between runs can remain undetected for long periods. This paper presents RAIL-GUARD, an indigenous, low-cost, contactless Integrated Track Monitoring System built on a portable wheel-mounted trolley that can be operated by local staff without a dedicated train path. A camera, Time-of-Flight (ToF) laser sensors, an inertial measurement unit, and a rotary encoder are combined through reliability-weighted sensor fusion, in which each sensor's contribution is scaled by a confidence score, to estimate track geometry (gauge, alignment, twist), while a YOLO-based detector identifies rail cracks, corrosion, damaged sleepers, and loose fasteners. Each defect is geo-tagged by chainage and GPS, stored in a chainage-indexed historical database, and analysed with deterioration-trend models to produce an explainable Track Health Score, risk level, and maintenance priority on a React/FastAPI dashboard. The system follows an embedded-edge-cloud architecture that keeps logging and anomaly flagging available when connectivity is lost, using Wi-Fi, GSM, or LoRa links with local MicroSD buffering. A working prototype and dashboard are implemented, and a four-level validation protocol covering unit, integration, field, and robustness testing is defined for field trials. Quantitative accuracy and cost figures await those trials. RAIL-GUARD offers a scalable, economical route to condition-based railway maintenance.

Keywords: railway track monitoring, sensor fusion, YOLO, predictive maintenance

INTRODUCTION

A. Background

Railway infrastructure is among the largest transportation assets in India, carrying millions of passengers and large freight volumes over a network of more than one hundred thousand track-kilometres. Safety, ride comfort, and permissible speed depend directly on track condition, yet dynamic wheel loads, thermal cycling, moisture ingress, and ballast settlement progressively cause gauge widening, twist, rail cracking, corrosion, and loosened fastenings that compromise safety if not rectified in time.

B. Motivation

The central motivation of this work is to reduce Indian Railways' dependence on costly imported inspection technology by demonstrating that an indigenous system built

from commercially available sensors, open-source computer-vision frameworks, and low-cost embedded controllers can deliver comparable defect-detection functionality at a substantially lower cost.

C. Contributions

The main contributions of this work are: (a) a low-cost, contactless, multi-sensor trolley integrating vision, ToF/laser, inertial, and encoder-based sensing; (b) an adaptive, reliability-weighted fusion scheme that is robust to sensor noise and partial occlusion; (c) a YOLO/OpenCV pipeline for rail cracks, corrosion, damaged sleepers, and loose fasteners; (d) a chainage-based historical database and deterioration-trend model that yields an explainable Track Health Score, risk level, and maintenance priority; and (e) a complete embedded-to-cloud architecture with a React/FastAPI dashboard.

LITERATURE REVIEW

Automated visual track inspection has been studied extensively. Early systems used image processing and classical machine learning to locate rail components and fasteners (Aytekin et al., 2015; Gibert et al., 2015) and to detect rail-surface defects (Zhang et al., 2018). Deep learning then improved defect recognition (Faghih-Roohi et al., 2016; Gibert et al., 2017), with comparative evidence on fastener inspection (Wei et al., 2019) and one-stage detectors for real-time component detection (Wang et al., 2020) that motivate the YOLO-based detection used here. Mittal and Rao (2017) showed that transfer learning copes with scarce labelled data and proposed a track health index, a single interpretable output of the kind RAIL-GUARD also adopts. On the prediction side, Liao et al. (2022) reviewed machine-learning models of geometry degradation, and Koohmishi et al. (2024) reviewed the fusion of complementary sensing technologies with machine learning for predictive, condition-based maintenance. Table I summarises representative studies. Many of these works stop at defect detection or geometry traces; RAIL-GUARD instead combines edge pre-processing on ESP32-class microcontrollers, cloud-based inference, and a dashboard built around one Track Health Score, risk level, and maintenance priority.

Table 1 Summary of Representative Related Work

Study	Focus	Contribution
Faghih-Roohi et al. (2016)	Rail surface defects	Deep convolutional neural networks for detecting rail-surface defects.
Gibert et al. (2017)	Track inspection	Deep multitask learning for inspecting track components and defects.
Mittal & Rao (2017)	Vision-based monitoring	Transfer learning to detect sun-kinks and loose ballast; proposed a track health index.
Wang et al. (2020)	Component detection	One-stage deep networks for real-time detection of railway track components.
Liao et al. (2022)	Geometry degradation	Review of ANN, SVM, and Grey Model approaches for predicting geometry degradation.
Koohmishi et al. (2024)	Sensor fusion for health monitoring	Review of GPR–InSAR fusion and machine learning for predictive, condition-based maintenance.

Beyond these studies, a wider literature addresses IoT-based structural health monitoring, edge-AI deployment on microcontrollers, and cloud-based asset-management dashboards. Common themes include the trade-off between on-device (edge) and cloud inference, the importance of geo-referencing defect data so that maintenance planning is actionable, and the value of giving maintenance staff a small number of prioritised, explainable alerts rather than raw sensor streams. A further recurring finding is that no single sensing modality is sufficient for reliable condition assessment: complementary sensors (imaging, geophysical, or vibration data) improve detection robustness, which motivates the reliability-weighted fusion adopted in this work.

RESEARCH GAP

Existing work and current practice leave four gaps. First, imported Track Recording Cars are costly to acquire and service, cannot easily be adapted to India-specific track conditions, and run on infrequent, centrally scheduled cycles. Second, many published studies stop at defect detection or geometry traces and do not deliver a single, explainable output that maintenance staff can act on. Third, no single sensing modality is reliable under real-world noise, so complementary sensors must be fused with explicit weighting. Fourth, few systems combine edge pre-processing, geo-referenced chainage storage, deterioration-trend prediction, and a dashboard in one low-cost, contactless platform. RAIL-GUARD is designed to address these gaps.

PROBLEM STATEMENT

Track recording on Indian Railways is presently carried out largely with imported Track Recording Cars (TRCs). These systems have three structural drawbacks. First, the hardware and the specialised calibration and service support must be imported, making capital and maintenance costs very high. Second, their proprietary architectures cannot easily be customised or extended for India-specific track conditions and defect types. Third, infrequent, centrally scheduled inspection runs let defects that develop between cycles go

undetected for long periods. The problem addressed here is therefore the development of an indigenous, contactless Integrated Track Monitoring System (ITMS) for track recording. Its design objectives are to (a) detect multiple defect types without contact using low-cost commodity sensors, (b) fuse heterogeneous sensor data reliably under real-world noise and vibration, (c) geo-locate every defect through chainage mapping, (d) deliver an explainable, predictive track-health output instead of raw data, and (e) keep hardware cost an order of magnitude below comparable imported systems.

PROPOSED SYSTEM

This paper proposes RAIL-GUARD, an AI-based predictive track-health monitoring system built around a lightweight wheel-mounted trolley that can be pushed manually or motor-driven along the track. Unlike a TRC, it needs no dedicated train path and can be operated by local staff on yard lines, branch lines, and sections between scheduled TRC runs.

A. Adopted Solution

The adopted solution is a portable inspection trolley that integrates the following capabilities: a camera combined with AI models detects cracks, corrosion, damaged sleepers, and loose fasteners; ToF/laser sensors measure geometry and alignment; a vibration sensor identifies abnormal irregularities during motion; GPS tagging records the exact location of each defect; an AI-based fusion layer combines camera, vibration, and geometry data for robust detection; ESP32 and Raspberry Pi units process data on board and transmit it over LoRa, GSM, or Wi-Fi; a web dashboard displays defect type, severity, location, and overall track-health status; and automatic alerts prioritise critical defects for maintenance action. Overall, the system is positioned as a low-cost, indigenous, and scalable alternative to imported track-recording systems, capable of being deployed at several locations simultaneously across a railway division.

B. System Requirements

Functional requirements. The system must (a) acquire synchronised image, distance, inertial, and positional data while the trolley is in motion; (b) detect and classify rail cracks, corrosion, damaged sleepers, and loose fasteners from the acquired imagery; (c) estimate track geometry parameters, namely gauge, alignment, and twist, from ToF/laser measurements; (d) tag every detected defect with its chainage location using encoder and GPS data; (e) compute and store a Track Health Score, risk level, and maintenance priority for each inspected section; (f) transmit inspection data to a central server over LoRa, GSM, or Wi-Fi depending on connectivity; and (g) present live monitoring data, defect maps, and alerts on a web-based dashboard.

Non-functional requirements. Portability requires the trolley to be light enough for two-person placement on and removal from the track. Cost requires total hardware cost to

remain a small fraction of that of an imported track-recording system. Reliability requires safe operation under partial sensor failure through adaptive fusion. Real-time performance requires defect detection and geometry estimation to keep pace with the operating speed without perceptible data loss.

Extensibility requires that new defect classes or sensors can be added without redesigning the full pipeline.

C. Hardware Design

The mechanical platform is a rover chassis with four railway-profile track wheels driven by four 12 V DC geared motors; an adjustable camera pole sets the viewing angle over the rail head and fasteners. The electronics comprise two ESP32 DevKit V1 boards (motor control, sensor polling, communication), two ESP32-CAM modules (image capture), a BTS7960 driver for the traction motors, VL53L1X ToF sensors (non-contact gauge and profile measurement), two NEO-6M GPS modules (absolute positioning to complement encoder chainage), a MicroSD module (local backup logging), and an active buzzer that sounds on critical defects or faults. A 12 V Li-ion/LiPo pack with a main switch feeds the motors directly and, through a 5 V buck converter, the logic and sensors. This two-rail arrangement isolates the electronics from motor-driver switching noise. Table II lists the complete hardware bill of materials.

Table 2 Hardware Components and Their Functions

Component	Quantity	Function
Rover chassis with track wheels	1 set (4 wheels)	Mechanical platform and rail guidance
12 V DC geared motor	4	Traction drive
BTS7960 motor driver	1–2	High-current motor control
ESP32 DevKit V1	2	Control and communication
ESP32-CAM (AI-Thinker)	2	Onboard image acquisition
VL53L1X ToF sensor	2+	Contactless geometry measurement
NEO-6M GPS module	2	Absolute position reference
MicroSD module and card	1	Local data logging and backup
Active buzzer	1	On-trolley audible alert
12 V Li-ion/LiPo pack with 5 V buck converter	1	Power supply

D. Calibration and Operating Considerations

Calibration. Before each inspection run, the ToF sensors are zeroed against a reference gauge block of known width, the encoder is calibrated by driving the trolley over a measured length of track to derive millimetres per pulse, and the camera exposure and focus are verified on a printed reference target. Wheel wear and slip change the effective wheel circumference, so periodic re-calibration against surveyed chainage markers is part of the operating procedure. Where the GPS fix is available, it is used to bound the accumulated

encoder drift, and the chainage value is re-anchored whenever the trolley passes a known marker.

Operating environment. The trolley is intended for yard lines, branch lines, and sections between scheduled recording runs, and must therefore tolerate dust, vibration, direct sunlight, and intermittent rain. The camera is shaded to limit glare on the rail head, the electronics are housed in a sealed enclosure, and the ToF measurements are filtered because reflective or wet rail surfaces change the returned signal strength. Operation is restricted to blocks protected by the competent railway authority so that the trolley never shares the track with train movements, and the audible buzzer and dashboard alerts give the operator immediate notice of a critical defect or sensor fault.

SYSTEM ARCHITECTURE

A. Embedded–Edge–Cloud Architecture

The end-to-end flow has twelve stages (Fig. 1): data acquisition, pre-processing, sensor-quality assessment, adaptive sensor fusion, track-feature extraction, AI-based defect detection with severity estimation, chainage mapping through encoder and GPS data, database management, deterioration analysis, predictive modelling, decision support with real-time alerts, and a web dashboard. It operates as a continuous inspection-and-learning loop in which dashboard outputs and maintenance feedback refine detection thresholds and prediction models.

Physically, data flows from an ESP32-based embedded layer to a Raspberry Pi edge layer, which runs lightweight inference and buffers data during connectivity loss, and then to a cloud layer for historical storage, model retraining, and dashboard hosting. This embedded–edge–cloud structure lets the trolley keep logging and flagging anomalies when the network is unavailable, as in remote or tunnel sections.

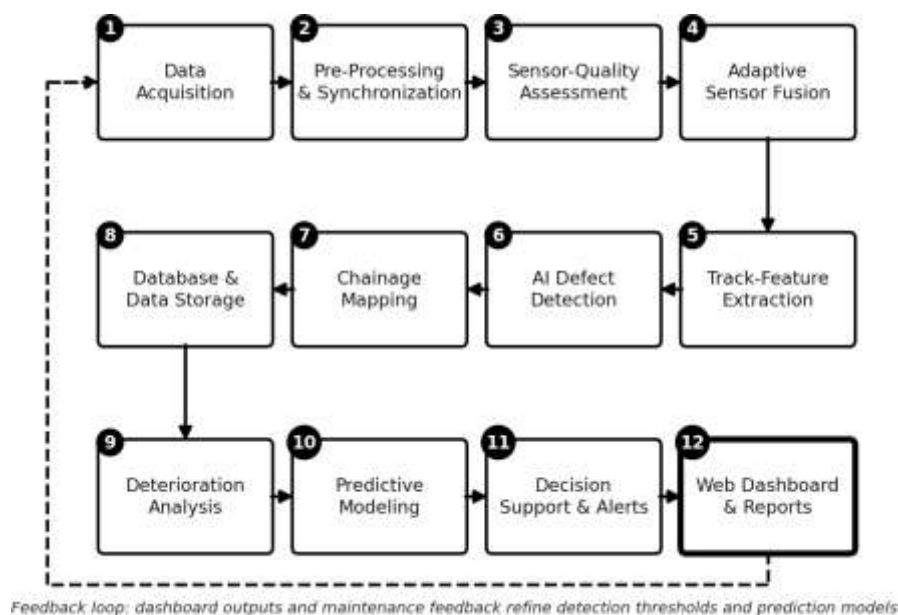


Figure 1 Twelve-Stage Process Flow of RAIL-GUARD with Feedback Loop

B. Software and Communication

Firmware for the ESP32 boards (Arduino IDE 2.x) handles motor control, fixed-interval sensor polling, timestamping, MicroSD buffering, and packetised transmission, with a watchdog that restarts acquisition if a sensor read fails. The detection pipeline uses Python and OpenCV for contrast normalisation and region-of-interest cropping, and a YOLO model trained with TensorFlow/PyTorch at a frame rate matched to trolley speed. The backend exposes Flask/FastAPI REST endpoints for data ingestion, defect queries, and health-score retrieval; Firebase provides real-time synchronisation with the dashboard, and MongoDB is the chainage-indexed historical store.

Communication strategy. Data are sent opportunistically over Wi-Fi where available, falling back to GSM in areas with cellular coverage, and to long-range, low-bandwidth LoRa in remote sections. If no link exists, data are buffered on the MicroSD card and uploaded once connectivity returns, so no inspection data are lost. This fallback logic is shown in Fig. 2.

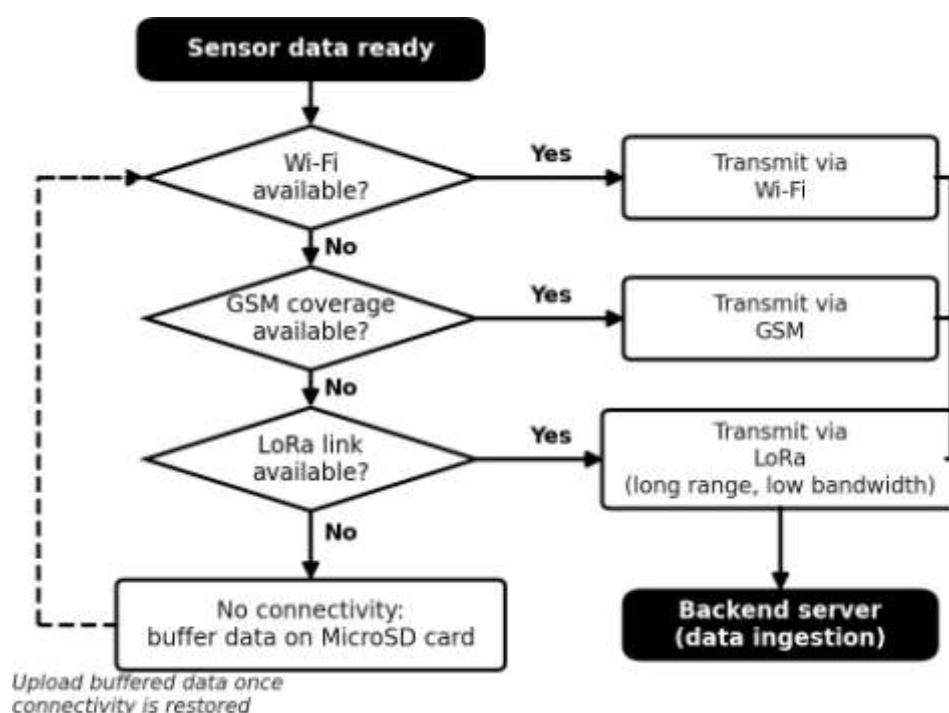


Figure 2 Communication Strategy of the Trolley: Wi-Fi, GSM, and LoRa Fallback with Local MicroSD Buffering METHODOLOGY / WORKFLOW

The methodology is a seven-stage pipeline from multi-sensor acquisition to maintenance decision (Fig. 3).

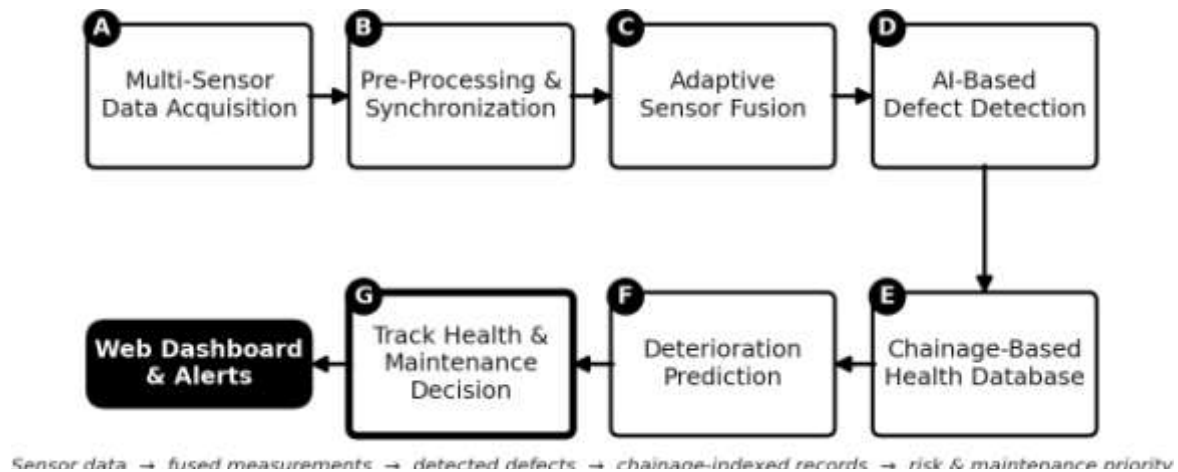


Figure 3 Seven-Stage Methodology Pipeline of RAIL-GUARD

Data acquisition. A forward-facing camera captures the rail surface, sleepers, and fasteners; a laser/ToF sensor pair measures lateral rail-head separation (gauge) and vertical profile (alignment and twist); an IMU records vibration and tilt signatures; and a wheel-mounted rotary encoder supplies the longitudinal distance used for chainage referencing.

Pre-processing and synchronisation. Moving-average and median filters remove noise and outliers, and all readings are time-synchronised to a common chainage reference derived from the encoder pulse count. A camera detection, a ToF geometry anomaly, and an IMU vibration spike can thus be associated with the same physical location.

Adaptive sensor fusion. Each modality can degrade independently, for example the camera through poor lighting or dust and the ToF sensors through surface reflectivity. For sensor i , a confidence score c_i is computed from signal-quality indicators such as image sharpness and return-signal strength. The fused estimate of a track parameter is the confidence-weighted average in Equation 1, where x_i is the reading of sensor i and the weight w_i is its normalised confidence.

$$x_{fused} = \sum_{i=1 \dots n} w_i x_i, \quad w_i = c_i / \sum_{j=1 \dots n} c_j \quad (1)$$

A temporarily unreliable sensor, such as a camera frame affected by glare, therefore contributes less to the fused estimate without being discarded.

AI-based defect detection. Camera frames are processed by a YOLO-based detector (implemented with OpenCV, TensorFlow, and PyTorch) trained to recognise rail cracks, corrosion patches, damaged sleepers, and loose or missing fasteners. Each detection carries a class label, bounding box, and confidence score, which is later combined with geometry and vibration anomalies during fusion to reduce false positives.

Chainage database and deterioration prediction. Fused measurements, defects, and confidence scores are stored in a chainage-indexed database that can be queried by exact location and inspection timestamp, because degradation trends must be computed per location. For each segment k , trend-based machine-learning models estimate the rate of

change of key parameters, and a deterioration-risk indicator is obtained as in Equation 2, where S_k is the current severity, m_k the observed trend slope, and α and β tunable weights.

$$R_k = \alpha S_k + \beta m_k \quad (2)$$

The weights balance current condition against speed of degradation, so that rapidly deteriorating segments are flagged even when their present severity is moderate.

Track Health Score formulation. The Track Health Score is computed per segment from three normalised sub-indices, each scaled to the range 0 to 1, where 0 denotes a fully healthy condition: a defect index, a geometry index, and the deterioration-risk indicator R_k from Equation 2. The defect index aggregates the severity of all detections in the segment, weighting each detection by its class-specific criticality and its fused confidence, so that a high-confidence rail crack contributes more than a low-confidence fastener anomaly. The geometry index expresses how far gauge, alignment, and twist deviate from their nominal values relative to the permissible tolerance band. The three indices are combined by a weighted sum, and the result is subtracted from 100 so that a higher score always means a healthier segment. Because the weights are explicit parameters, a maintenance engineer can inspect which component drove a low score, which is what makes the output explainable rather than a black-box number.

Risk levels and priority ranking. The 0 to 100 score is mapped to the categorical risk levels Low, Medium, and High through two configurable thresholds, which are exposed on the dashboard settings page shown in Fig. 6. Segments are then ranked for maintenance by risk level first and by the deterioration-risk indicator second, so that two segments with the same risk level are ordered by how fast they are degrading. Any single detection of a critical class, such as a suspected rail crack above the configured confidence threshold, raises an immediate alert regardless of the segment score, because averaging must never hide a safety-critical defect.

Defect severity estimation. Each detected defect is assigned a severity grade from its class, its apparent size in the image relative to the calibrated field of view, and whether a co-located geometry or vibration anomaly corroborates it. Corroboration across modalities increases the grade, while an isolated, low-confidence detection with no supporting evidence is held at a lower grade and flagged for visual confirmation. This cross-checking reduces false positives, which is important because unnecessary alerts erode the trust of maintenance staff in an automated system.

Track health and maintenance decision. Defect severity, geometry deviation, and deterioration risk are combined into an explainable Track Health Score on a 0–100 scale, a categorical risk level (Low, Medium, or High), and a maintenance-priority ranking, all presented on the web dashboard.

IMPLEMENTATION AND EVALUATION

A. Prototype and Dashboard

The assembled prototype (Fig. 4) has a four-wheel-drive chassis, motor housing, and battery compartment. The React dashboard, served by the FastAPI backend, displays defect type, severity, chainage location, and Track Health Score for each segment, with a defect map and summary widgets for risk-level distribution and maintenance-priority counts. Figs. 5 and 6 show the implemented dashboard: rover control, live sensor monitoring, and the alert centre (Fig. 5), and inspection-threshold settings, the live ESP32-CAM feed, and inspection-location status (Fig. 6).

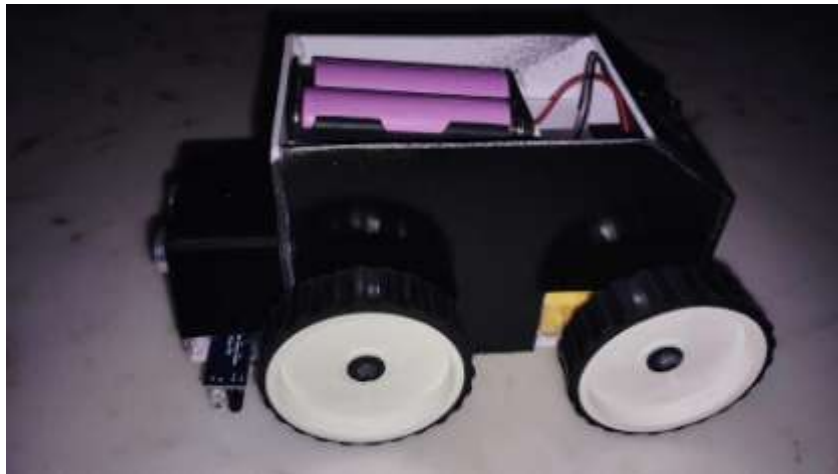


Figure 4 Assembled Prototype Trolley

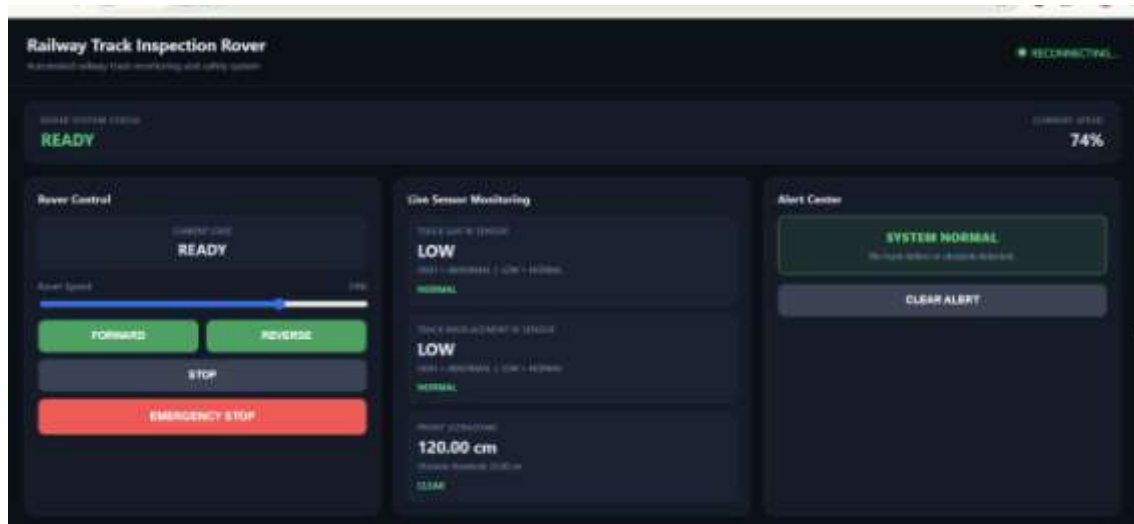


Figure 5 Web Dashboard: Rover Control, Live Sensor Monitoring, and Alert Center

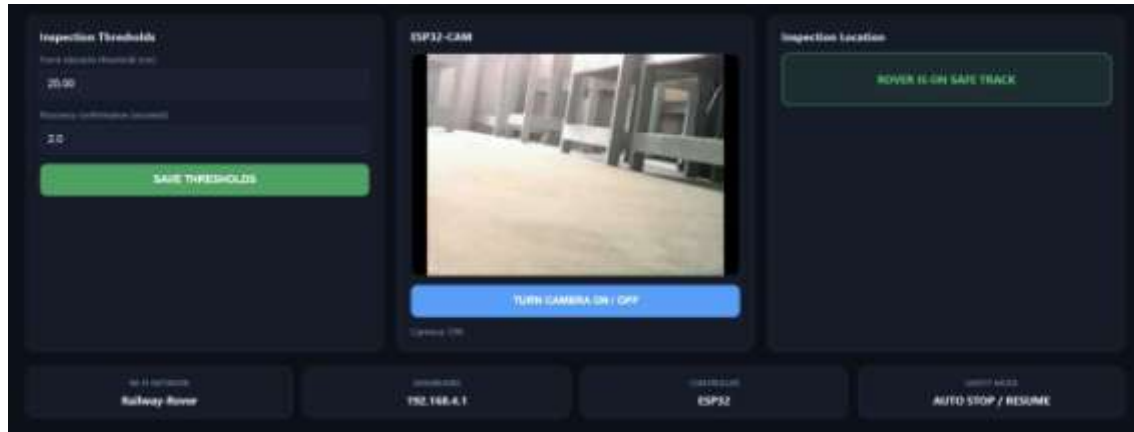


Figure 6 Web Dashboard: Inspection Threshold Settings, Live ESP32-CAM Feed, and Location Status

B. Module Development and Integration

Table III lists the modules developed. Development was bottom-up: each module was first validated in isolation on recorded sample data, and integration then proceeded in data-flow order, with each newly connected stage tested against the output of the previous one. This exposed data-format, timestamp-alignment, and unit-conversion problems early.

Table 3 Module Development and Initial Integration Outcome

Module	Development	Initial integration
Data acquisition	Camera + IMU + laser + encoder	Raw sensor data
Pre-processing	Filtering and noise removal	Clean sensor data
Chainage mapping	Encoder and timestamp synchronisation	Location-tagged data
AI defect detection	YOLO + OpenCV	Defect + confidence
Adaptive sensor fusion	Camera + laser + IMU reliability weighting	Fused condition
Track health	Health scoring model	Health score
Historical database	Chainage-wise data storage	Track history
Prediction	ML + deterioration trend analysis	Deterioration risk
Maintenance priority	Risk-based ranking	Priority alerts
Dashboard	React + FastAPI	Live monitoring and reports

C. Validation Plan

Because the system is at the prototype stage, a structured four-level plan guides subsequent field trials.

Unit-level testing. Each hardware and software module (sensor drivers, the detection model, the fusion algorithm, the database, and the dashboard API) is tested independently against known reference inputs, such as calibrated distance targets for the ToF sensors and a labelled image set for the YOLO detector, to verify correctness before integration.

Integration testing. Once modules pass unit tests, they are connected in sequence and exercised with recorded multi-sensor datasets to confirm that data flows correctly from acquisition through fusion to the dashboard, with correct chainage alignment across all modalities.

Field trial protocol. The trolley will be run over track sections with known, documented defects verified through manual inspection records. Detected locations, classifications, and geometry estimates will be compared with this ground truth using per-class detection precision and recall, mean absolute error in gauge and alignment relative to reference instruments, and chainage localisation error relative to known survey markers.

Reliability and robustness testing. Individual sensor inputs will be deliberately degraded, for example by partially obstructing the camera or operating in low ambient light, to verify that the adaptive fusion scheme down-weights the affected sensor while the overall Track Health Score remains stable and does not produce spurious high-severity alerts.

Evaluation metrics. Detection performance will be reported per defect class using precision, recall, and the F1 score, together with mean average precision at a fixed intersection-over-union threshold, since a single overall accuracy figure would hide poor performance on rare classes such as cracks. Geometry estimation will be reported as the mean absolute error and the maximum error of gauge, alignment, and twist against reference instruments, and chainage localisation as the distance error against surveyed markers. For the Track Health Score, agreement with the ranking produced by manual inspection records will be assessed, since the practical value of the score lies in ordering sections correctly for maintenance. Processing latency from frame capture to dashboard update will also be measured to confirm real-time operation at the intended trolley speed.

Dataset considerations. Defect images from Indian track conditions are scarce, and defective samples are far fewer than healthy ones. The training set will therefore be built from images collected during trial runs, augmented with changes in brightness, contrast, blur, and orientation, and the train, validation, and test partitions will be separated by track section rather than by individual frame, so that near-duplicate consecutive frames do not leak between partitions and inflate the measured accuracy. Transfer learning from a pre-trained detector, as supported by Mittal and Rao (2017), is expected to reduce the amount of labelled data needed.

Cost evaluation. Because the cost advantage is a design objective, it will be tested by preparing an itemised bill of costs for the trolley, covering hardware, assembly, calibration, and consumables, and comparing it with published or quoted figures for imported track-recording systems on both capital and annual maintenance cost. The comparison will also consider the difference in capability, since a trolley complements rather than replaces a full track-recording car, and it will be reported only after the field trials establish the accuracy that the lower-cost system actually achieves.

D. Discussion

The prototype shows that the full chain from multi-sensor acquisition to dashboard alerting can be built from commercially available sensors, open-source vision frameworks, and low-cost microcontrollers. Its advantages are low cost, local serviceability, extensibility to new defect classes, and deployment of several units across a division instead of one centrally scheduled run. Three limitations remain. First, quantitative detection accuracy, geometry-measurement error, and localisation error have not yet been established and await the field trials above. Second, the deterioration model will initially work with a small historical dataset, which favours simple trend-based and grey-type models of the kind reviewed by Liao et al. (2022). Third, the cost advantage is a design objective that must be confirmed against a complete bill of costs.

Several design choices are intended to make the system practical for maintenance staff. The embedded–edge–cloud split keeps basic anomaly flagging available when connectivity is lost, which matters in tunnels and remote sections. Producing a single Track Health Score with a risk level and priority ranking, rather than raw logs, follows the interpretable-output philosophy of earlier track health index work (Mittal & Rao, 2017) and lets planners act on a short list of sections instead of interpreting sensor traces. Reliability-weighted fusion addresses the finding that no single modality is sufficient: a degraded camera frame or a reflective rail surface lowers one sensor's weight instead of corrupting the estimate. Finally, the modular pipeline allows new defect classes, sensors, or prediction models to be added with limited rework.

Threats to validity. Several factors could affect the conclusions drawn from the planned trials. Defects documented in manual records may themselves be incomplete, so detections that appear to be false positives may sometimes be real defects missed by earlier inspection; confirmed disagreements will therefore be re-inspected on site. Results obtained on yard and branch lines may not transfer to high-speed main lines, where vibration levels and operating constraints differ. Finally, a model trained on images from a limited number of sections may generalise poorly to unseen lighting or ballast conditions, which is why separate test sections and the degradation tests above are included.

Path to deployment. A staged route to deployment is envisaged: validation on a short test section under controlled conditions, supervised trials on a yard or branch line with railway engineers verifying alerts, and finally operation of several trolleys across a division with a shared dashboard. At each stage the thresholds, the fusion weights, and the health-score weights can be tuned using feedback from maintenance staff, so that the system matures through use rather than requiring a complete redesign.

E. Scalability, Safety, and Future Enhancements

Scalability. The embedded–edge–cloud structure scales by adding trolleys rather than by enlarging a single machine. Each unit logs to the same chainage-indexed database, so a division can build a shared record of track condition from many low-cost units, and the

dashboard can aggregate risk levels and maintenance priorities across sections. Because the cloud layer holds the detection model, an improved model trained on pooled data can be distributed to all units without changing their hardware.

Safety and data integrity. Safe operation depends on both the procedure and the design. Trolley movement is limited to track blocks protected by the competent authority, and the motor controller is designed to stop on loss of the control link or on a failed sensor check. For data integrity, every record carries a timestamp, chainage, and sensor-confidence value, and locally buffered data are uploaded without alteration, so that any alert can be traced back to the readings and image frames that produced it. Maintenance staff remain responsible for the final decision, and the system is designed to support their judgement, not replace it.

Future enhancements. Several extensions follow naturally from the modular design. Additional sensing, such as ultrasonic or eddy-current probes for sub-surface rail flaws, could be added as a new input to the fusion stage. More defect classes, for example ballast condition or vegetation encroachment, could be introduced by retraining the detector. The deterioration model could move from simple trend analysis to richer sequence models once enough inspection history has accumulated, and a motor-driven version with route logging could allow semi-autonomous inspection of long sections under supervision.

CONCLUSION

This paper presented RAIL-GUARD, an indigenous, AI-based, contactless Integrated Track Monitoring System aimed at reducing dependence on costly imported track-recording equipment. By fusing camera, ToF/laser, IMU, and encoder data through reliability-weighted fusion, and applying deep-learning defect detection with chainage-based localisation, the system produces an explainable Track Health Score, risk level, and maintenance priority on a live dashboard. The modular architecture, spanning acquisition, pre-processing, fusion, AI-based detection, historical data management, deterioration prediction, and decision support, enables incremental development and testing of each stage, as shown by the module-level integration outcomes. Future work will (a) run field trials to validate detection accuracy and geometry precision against reference instruments, (b) retrain the deterioration model as historical data accumulate, (c) optimise communication and power subsystems for extended unattended operation, and (d) extend the system to multi-trolley, network-wide deployment so that Indian Railways can move progressively from schedule-based to condition-based and predictive maintenance.

REFERENCES

- [1] Ç. Aytekin, Y. Rezaeitabar, S. Dogru, and I. Ulusoy, "Railway fastener inspection by real-time machine vision," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 7, pp. 1101–1107, 2015, doi: 10.1109/TSMC.2014.2388435.

- [2] X. Gibert, V. M. Patel, and R. Chellappa, "Robust fastener detection for autonomous visual railway track inspection," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2015, pp. 694–701, doi: 10.1109/WACV.2015.98.
- [3] S. Faghih-Roohi, S. Hajizadeh, A. Núñez, R. Babuska, and B. De Schutter, "Deep convolutional neural networks for detection of rail surface defects," in *Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN)*, 2016, pp. 2584–2589, doi: 10.1109/IJCNN.2016.7727522.
- [4] X. Gibert, V. M. Patel, and R. Chellappa, "Deep multitask learning for railway track inspection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 1, pp. 153–164, 2017, doi: 10.1109/TITS.2016.2568758.
- [5] S. Mittal and D. Rao, "Vision based railway track monitoring using deep learning," *arXiv preprint*, 2017. [Online]. Available: <https://arxiv.org/abs/1711.06423>
- [6] H. Zhang, X. Jin, Q. M. J. Wu, Y. Wang, and Z. He, "Automatic visual detection system of railway surface defects with curvature filter and improved Gaussian mixture model," *IEEE Transactions on Instrumentation and Measurement*, vol. 67, no. 7, pp. 1593–1608, 2018, doi: 10.1109/TIM.2018.2803830.
- [7] X. Wei, Z. Yang, Y. Liu, D. Wei, L. Jia, and Y. Li, "Railway track fastener defect detection based on image processing and deep learning techniques: A comparative study," *Engineering Applications of Artificial Intelligence*, vol. 80, pp. 66–81, 2019, doi: 10.1016/j.engappai.2019.01.008.
- [8] T. Wang, F. Yang, and K.-L. Tsui, "Real-time detection of railway track component via one-stage deep learning networks," *Sensors*, vol. 20, no. 15, Art. no. 4325, 2020, doi: 10.3390/s20154325.
- [9] Y. Liao, L. Han, H. Wang, and H. Zhang, "Prediction models for railway track geometry degradation using machine learning methods: A review," *Sensors*, vol. 22, no. 19, Art. no. 7275, 2022, doi: 10.3390/s22197275.
- [10] M. Koohmishi, S. Kaewunruen, L. Chang, and Y. Guo, "Advancing railway track health monitoring: Integrating GPR, InSAR and machine learning for enhanced asset management," *Automation in Construction*, Art. no. 105378, 2024, doi: 10.1016/j.autcon.2024.105378.