



iJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: <https://doi.org/10.22214/ijraset.2026.85171>

www.ijraset.com

Call: ☎ 08813907089

E-mail ID: ijraset@gmail.com

A Survey of AI-Based Visual Defect Detection Techniques for Automotive Manufacturing

Rutvi Landge¹, Sakshi Barhate², Rutuja Kashid³, Ritesh Kadam⁴, Manisha Darak⁵

^{1, 2, 3, 4}Students, Department of Computer Engineering), RMD Sinhgad School of Engineering, Pune, Maharashtra, India

⁵Professor, Department of Computer Engineering), RMD Sinhgad School of Engineering, Pune, Maharashtra, India

Abstract: *Quality inspection is an essential process in automotive manufacturing, as defects in vehicle components can affect product quality, reliability, and production efficiency. Traditional inspection methods often rely on manual examination, which can be time-consuming and may produce inconsistent results under varying manufacturing conditions. Recent advancements in Artificial Intelligence (AI) and Computer Vision have created opportunities for automated visual defect inspection.*

This paper presents a survey of AI-based visual defect detection techniques applicable to automotive manufacturing. It examines approaches such as You Only Look Once (YOLO), image segmentation, and anomaly detection for identifying defects including scratches, dents, cracks, corrosion, and surface irregularities. The survey aims to compare these techniques based on their reported capabilities, limitations, computational requirements, and suitability for industrial applications. It also examines challenges related to dataset availability, class imbalance, changing environmental conditions, and deployment in real-world manufacturing environments. Based on the literature review, research gaps are identified to establish a foundation for Defect Lens, a proposed AI-powered visual quality inspection platform for automotive manufacturing.

Keywords: *Artificial Intelligence, Computer Vision, Automotive Manufacturing, Visual Defect Detection, YOLO, Quality Inspection.*

I. INTRODUCTION

Product quality is a central determinant of safety, reliability, durability, and customer satisfaction in the automotive industry. Components manufactured for vehicles must meet stringent specifications, yet the production process can introduce surface and structural defects, including scratches, dents, cracks, corrosion, paint irregularities, and other forms of surface damage. Detecting such defects at an early stage is essential, as it reduces rework, limits material waste, and lowers overall manufacturing costs.

Conventional quality control relies largely on manual visual inspection, which depends on the observation skills and experience of human inspectors. Although widely practiced, this approach is time-consuming and prone to inconsistency. Inspector fatigue, variations in illumination, complex surface textures, and the small size of certain defects can all degrade inspection reliability. These limitations have driven growing interest in automated visual inspection systems that apply computer vision and artificial intelligence (AI) to the analysis of images of manufactured components.

Recent advances in computer vision have made it possible to identify and localise defects in industrial imagery with increasing accuracy. Image preprocessing techniques improve input quality by enhancing contrast, suppressing noise, and normalising illumination, while object detection algorithms such as You Only Look Once (YOLO) can classify defect categories and estimate their approximate locations in a single forward pass. Libraries such as OpenCV provide the image-processing capabilities needed to support preprocessing and visual analysis. Together, these techniques can assist quality inspectors by highlighting potential defects and presenting inspection outcomes in a structured, traceable format.

Nevertheless, reliable defect detection remains a challenging problem. Variations in lighting conditions, specular reflections from metallic surfaces, limited availability of labelled training data, diversity in defect appearance, and the difficulty of detecting small defects can all adversely affect system performance. Consequently, the selection of appropriate techniques, together with a clear understanding of their limitations, is critical when developing AI-based inspection solutions for automotive manufacturing.

This paper presents a survey of AI-based visual defect detection techniques, with particular emphasis on computer vision, image preprocessing, and YOLO-based object detection. It reviews existing approaches, compares their capabilities and limitations, examines the factors that influence detection performance, and identifies research gaps and directions for future work.

In addition, this paper introduces Defect Lens, a proposed AI-powered visual quality inspection platform for automotive manufacturing.

The proposed architecture integrates React.js and TypeScript for the user interface, Java and Spring Boot for backend processing, Python and OpenCV for image processing, YOLO for defect detection, and PostgreSQL for the storage of inspection records. The platform is intended to identify potential surface defects, organise inspection data, and present results through an intuitive dashboard. Its practical effectiveness, however, remains to be established through implementation and systematic experimental evaluation.

II. LITERATURE SURVEY

A. Traditional Visual Inspection and Classical Machine Vision

Early automated inspection systems relied on handcrafted image-processing pipelines. These typically combined thresholding, morphological operations, edge detection [9], and texture descriptors such as Local Binary Patterns (LBP) [17] with conventional classifiers such as Support Vector Machines. Such methods are computationally inexpensive and interpretable, and they perform acceptably under controlled illumination and with homogeneous surfaces. However, they depend heavily on manually engineered features, which limits generalisation across defect types, materials, and imaging conditions. Czimmermann et al. [10] reviewed visual-based defect detection approaches for industrial applications and noted that classical methods struggle with low-contrast defects, textured backgrounds, and variable lighting. These shortcomings motivated the shift towards learning-based approaches [25].

B. Image Preprocessing Techniques

Preprocessing is an important stage in inspection pipelines because image quality directly influences detection accuracy. Commonly used techniques include grayscale conversion, noise suppression (Gaussian and median filtering), contrast enhancement, and illumination normalisation. Contrast Limited Adaptive Histogram Equalisation (CLAHE) [8] is widely adopted to improve local contrast and reveal low-visibility defects such as fine scratches. Edge-based operators [9] and morphological filters help highlight crack-like and linear structures. OpenCV [7] is the most widely used open-source library for implementing these operations. On reflective metallic surfaces, preprocessing alone is often insufficient to remove specular highlights, so it is typically combined with controlled lighting configurations or learned feature extraction. Data augmentation, including flipping, rotation, brightness and contrast jitter, and noise injection, is also a standard strategy for improving robustness when labelled data is scarce [15].

C. Deep Learning-Based Defect Detection

Convolutional Neural Networks (CNNs) have largely replaced handcrafted features by learning hierarchical representations directly from images. Reviews such as Tulbure et al. [20] and Jha and Babiceanu [21] show that deep learning models outperform classical techniques for surface defect classification, localisation, and segmentation across industrial domains, provided sufficient training data is available. He et al. [13] proposed an end-to-end steel surface defect detection approach that fuses multiple hierarchical features and evaluated it on the NEU surface defect dataset [12], demonstrating the benefit of multi-level feature fusion for defects of varying scale.

Deep learning approaches to defect analysis can be broadly grouped into three categories:

- Classification: assigns an image or patch a defect label, without localisation.
- Detection: predicts a class label and bounding box for each defect instance.
- Segmentation: produces pixel-level defect masks, for example using U-Net [19]-style architectures, which is useful where defect area and shape must be measured.

D. Two-Stage and Single-Stage Object Detectors

Region-based detectors such as Faster R-CNN [2] generate region proposals before classification and bounding-box refinement. They generally achieve strong localisation accuracy but at higher computational cost, which can limit their suitability for high-throughput production lines. Single-stage detectors, including SSD [3] and the YOLO family [1], predict classes and bounding boxes directly in one pass, favouring inference speed [31]. Feature Pyramid Networks (FPN) [18] introduced multi-scale feature fusion, which has proven valuable for detecting small objects and is widely incorporated into modern detectors.

E. YOLO-Based Defect Detection

Since the introduction of YOLO by Redmon et al. [1], successive versions have improved accuracy, speed, and training strategies. YOLOv4 [4] introduced a set of training and architectural refinements (“bag of freebies” and “bag of specials”), and YOLOv7 [5] further improved the trade-off between accuracy and speed through optimised training strategies.

Terven et al. [6] provide a comprehensive review of the YOLO lineage from YOLOv1 to YOLOv8 and YOLO-NAS, discussing architectural evolution, anchor-free designs, and deployment considerations. The Ultralytics YOLO implementations [22] are popular in practice because of their ease of training, export, and deployment.

In industrial inspection, YOLO-based models are attractive because they offer real-time inference, a unified pipeline for classification and localisation, and compatibility with edge deployment. Many studies adapt YOLO for surface defect detection, including automotive parts [26], by adding attention modules, enhancing multi-scale feature fusion, or modifying loss functions to improve small-defect sensitivity. Common weaknesses reported across such work include reduced accuracy on very small or low-contrast defects, sensitivity to class imbalance, and performance degradation when test conditions differ from the training distribution.

F. Anomaly Detection and Limited-Data Approaches

Supervised detectors require annotated defect images, which are costly and often scarce, since defects occur rarely in production. Unsupervised and anomaly-detection approaches address this by learning the appearance of defect-free samples and flagging deviations. The MVTec Anomaly Detection dataset [11] became a standard benchmark for this setting, AutoVI [28] is a more recent industrial dataset for evaluating such methods, and methods such as PatchCore [14] achieve strong results using memory banks of nominal patch features. These methods reduce the annotation burden but typically provide less precise defect-type classification than supervised detectors. Transfer learning [16], where models pre-trained on large datasets are fine-tuned on small industrial datasets, is another widely used strategy for coping with limited labelled data.

G. Comparative Summary

Approach	Strengths	Limitations
Classical machine vision (thresholding, edges, LBP) [9], [17]	Low computational cost; interpretable	Weak generalisation; sensitive to lighting and texture
Two-stage detectors (Faster R-CNN) [2]	High localisation accuracy	Slower inference; heavier resource needs
Single-stage detectors (SSD, YOLO) [1], [3], [5]	Real-time speed; unified detection pipeline	Difficulty with very small or low-contrast defects
Segmentation networks (U-Net) [19]	Pixel-level defect extent	Requires costly pixel-wise annotation
Anomaly detection (PatchCore) [14]	Needs only defect-free samples	Limited defect-type classification

H. Factors Affecting Detection Performance

The literature identifies several recurring factors that affect the reliability of AI-based inspection:

- 1) *Illumination and reflections*: specular highlights on metallic or painted surfaces can mask or mimic defects.
- 2) *Defect scale and contrast*: small, thin, or low-contrast defects such as hairline cracks and fine scratches are frequently missed.
- 3) *Dataset limitations*: small, imbalanced, or inconsistently annotated datasets reduce generalisation [15], [16].
- 4) *Intra-class variation and inter-class similarity*: defects of the same class may look different, while different classes (for example, scratches and surface marks) may look alike.
- 5) *Domain shift*: models trained under one camera or lighting setup may degrade when deployed in another.
- 6) *Real-time and deployment constraints*: accuracy must be balanced against latency and hardware cost.

I. Research Gaps

Based on the reviewed literature, the following gaps are identified:

- 1) Most public benchmarks (for example, NEU [12] and MVTec AD [11]) focus on steel strips or generic objects, and datasets specific to automotive components are limited.
- 2) Many studies concentrate on detection accuracy while giving limited attention to system-level integration, including data management, traceability, and inspector-facing interfaces.

- 3) Few works combine classical preprocessing with modern YOLO-based detection in an end-to-end, deployable platform evaluated under realistic factory conditions.
- 4) Robustness to reflective surfaces and varying illumination remains insufficiently addressed.
- 5) Reliable detection of small defects while maintaining real-time performance remains an open challenge.

III. PROPOSED SYSTEM

A. System Overview

Defect Lens is a proposed AI-powered visual quality inspection platform for automotive component manufacturing. It accepts images of manufactured components, preprocesses them, detects and localises surface defects using a YOLO-based object detector [1], [6], and stores the results as structured inspection records. These records are presented to quality inspectors through a web dashboard. The platform is designed as an inspector-assist tool: it highlights potential defects and records the outcome, while the final accept/reject decision remains with the human inspector.

The design addresses the gaps identified in Section II, particularly the limited attention to system-level integration, traceability, and inspector-facing interfaces, and the lack of end-to-end pipelines that combine classical preprocessing with modern YOLO-based detection.

B. System Architecture

The architecture follows a layered, service-oriented design with four main tiers:

- 1) Presentation layer: React.js with TypeScript.
- 2) Application layer: Java with Spring Boot.
- 3) AI processing layer: Python with OpenCV and YOLO.
- 4) Data layer: PostgreSQL.

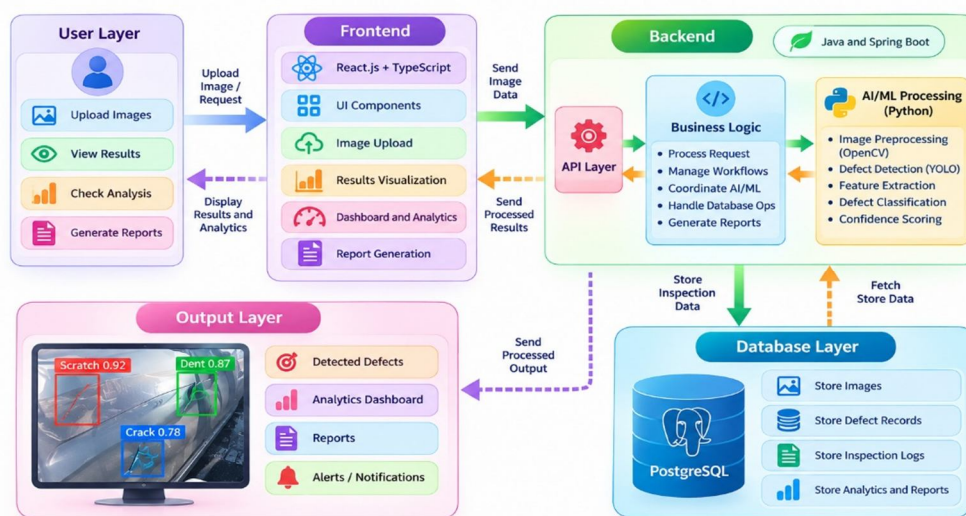


Fig. 1. Overall architecture of Defect Lens.

1) User Interface Module (React.js and TypeScript)

This module provides the inspector-facing dashboard. Its intended functions are:

- uploading component images, individually or in batches;
- displaying the original image with bounding boxes, defect class labels, and confidence scores overlaid;
- showing inspection history with filters by date, component type, defect category, and status;
- allowing an inspector to confirm or override each detection, so that the corrected labels can later support model retraining;
- presenting summary views of defect frequency and trends.

TypeScript's static typing helps keep the data contracts between the interface and the backend consistent.

2) Backend Service Module (Java and Spring Boot)

The backend is the central coordinator. It exposes REST APIs to the frontend, validates incoming requests and image files, forwards images to the inference service, receives structured detection results, applies business rules (for example, flagging a component for review when any defect exceeds a configurable confidence threshold), and persists records to the database. It also manages user authentication, role-based access control, and audit logging.

3) Image Preprocessing Module (Python and OpenCV)

Image quality strongly affects detection accuracy, so each image passes through a configurable preprocessing pipeline implemented with OpenCV [7]:

- *Resizing and normalisation*: conversion to the input resolution expected by the detector.
- *Noise suppression*: Gaussian or median filtering to reduce sensor noise.
- *Contrast enhancement*: CLAHE [8] to improve local contrast and reveal low-visibility defects such as fine scratches.
- *Illumination normalisation*: to reduce the effect of uneven lighting across the surface.

Each step is optional and parameterised, because the benefit of preprocessing varies by surface type and imaging setup. The contribution of each step is to be measured through ablation experiments rather than assumed. For highly reflective surfaces, preprocessing is expected to be insufficient on its own and is intended to be combined with controlled lighting.

4) Defect Detection Module (YOLO)

The preprocessed image is passed to a YOLO-based detector, which predicts a defect class, bounding box, and confidence score for each detected region in a single forward pass [1]. The implementation is planned using the Ultralytics framework [22], which supports training, validation, and export. The target defect classes are scratches, dents, cracks, corrosion, and paint irregularities. Training is planned to use transfer learning from pre-trained weights [16] and data augmentation [15] to mitigate limited labelled data. The specific YOLO version and model size will be selected experimentally, balancing accuracy, small-defect sensitivity, and inference latency, with reference to the YOLO family comparisons in [5], [6].

5) Post-Processing and Result Generation

Raw detections are filtered using a confidence threshold and non-maximum suppression. The module then produces:

- an annotated image with bounding boxes and labels;
- a structured JSON record containing defect class, confidence, bounding-box coordinates, and relative defect size.

6) Data Storage Module (PostgreSQL)

PostgreSQL stores inspection records in a relational schema, enabling traceability and querying. A proposed logical schema is shown in Table I.

TABLE I. Proposed database entities

Entity	Key Attributes
User	user_id, name, role, credentials (hashed)
Component	component_id, part_type, batch_number, production_date
Inspection	inspection_id, component_id, user_id, timestamp, overall_status
Defect	defect_id, inspection_id, class_label, confidence, bbox_coordinates, inspector_verdict
Image	image_id, inspection_id, file_path, type (original/annotated)

Image files are stored on the file system or object storage, and only their paths and metadata are held in the database.

C. Workflow

The end-to-end inspection workflow is as follows:

- 1) The inspector logs in and uploads an image of a component through the dashboard.
- 2) The Spring Boot backend validates the request and sends the image to the Python inference service.
- 3) The inference service applies the OpenCV preprocessing pipeline.

- 4) The YOLO model detects defects and returns classes, bounding boxes, and confidence scores.
- 5) Post-processing filters the detections and generates an annotated image.
- 6) The backend stores the inspection record, defect entries, and images in PostgreSQL.
- 7) The dashboard displays the annotated result, and the inspector confirms or corrects each detection.
- 8) Confirmed and corrected labels are retained as potential data for future model retraining.



Fig. 2. Flowchart of the inspection workflow.

IV. COMPARATIVE ANALYSIS

Deep learning and computer vision techniques differ in what they can detect, how much computing power they need, how much annotation they require, and how well they suit real-time inspection. This section compares four representative approaches and assesses how well each fits the requirements of the proposed *Defect Lens* platform.

A. Comparison of Techniques

TABLE II. Comparison of defect detection techniques

Technique	Main Function	Advantages	Limitations	Suitability for Defect Lens
CNN-based classification [20]	Assigns an image or patch to a defect category	Learns discriminative visual features automatically; relatively simple to train	Does not directly provide defect location	Useful for classifying components as defective or non-defective
YOLO-based object detection [1], [5], [6]	Predicts defect class and bounding box in a single pass	Fast inference; combines detection and localisation in one pipeline	Small or low-contrast defects may be missed	Suitable for locating visible defects on automotive components
Image segmentation (e.g., U-Net) [19]	Labels defect regions at pixel level	Provides detailed defect boundaries and extent	Requires pixel-level annotation; may need more computation	Useful for analysing irregular scratches, cracks, and corrosion
Unsupervised anomaly detection [11], [14]	Detects deviations from the normal appearance of a product	Reduces dependence on labelled defect samples	May struggle with natural variation in normal products; limited defect-type classification	Useful when defective samples are scarce

B. Discussion

- 1) **CNN-based classification.** Classification networks can reliably support the identification of defect categories such as scratches, dents, and corrosion, and reviews of industrial defect detection report that deep models generally outperform handcrafted approaches when sufficient training data is available [20]. However, classification alone does not indicate where a defect lies on a component. This limits its value in inspection workflows where inspectors need to see and verify the exact defect region.
- 2) **YOLO-based object detection.** YOLO detectors identify and localise multiple defects within a single image through one forward pass [1], and later versions have improved the balance between accuracy and speed [5], [6]. These properties suit industrial settings where throughput matters. Their performance nevertheless depends on the model architecture, the quality and diversity of the dataset, the size of the defects, and the available computing resources. Small and low-contrast defects remain a known weakness.
- 3) **Image segmentation.** Segmentation produces pixel-level masks, which allow defect area and shape to be measured [19]. This is particularly relevant for irregular defects such as long scratches, cracks, and corrosion patches. The main drawback is the effort required to prepare pixel-level annotations, which can be considerably greater than drawing bounding boxes.
- 4) **Unsupervised anomaly detection.** By learning the appearance of defect-free products, anomaly detection methods can flag unusual patterns without requiring labelled defect images [11], [14]. This is an attractive option because defects are rare in production. Nevertheless, variations in lighting, surface texture, and manufacturing conditions can lead to false alarms, and these methods typically provide less precise defect-type information than supervised detectors.

C. Rationale for the Proposed Approach

The comparison shows that no single technique satisfies every requirement of automotive visual inspection. Classification supports defect categorisation, object detection provides localisation, segmentation offers detailed defect boundaries, and anomaly detection helps identify previously unseen abnormalities.

For the first implementation of *Defect Lens*, YOLO-based object detection is selected as the core detector, for three reasons. First, the platform must show inspectors where each defect is, together with its class and confidence. Second, bounding-box annotation is less costly than pixel-level annotation. Third, single-stage detection is compatible with near real-time inspection. OpenCV-based preprocessing is added to improve input quality before detection. Segmentation and anomaly detection are not included in the initial design. They are considered possible extensions, with segmentation for measuring defect extent and anomaly detection for settings where defective samples are scarce. The suitability of these choices remains to be confirmed through implementation and systematic experimental evaluation.

V. TAXONOMY OF AI-BASED DEFECT DETECTION TECHNIQUES

To organise the techniques reviewed in this paper, a taxonomy is proposed in Fig. 3. It groups methods by their role in an inspection pipeline: classical machine vision, deep learning-based detection, and supporting methods that improve data quality and training.

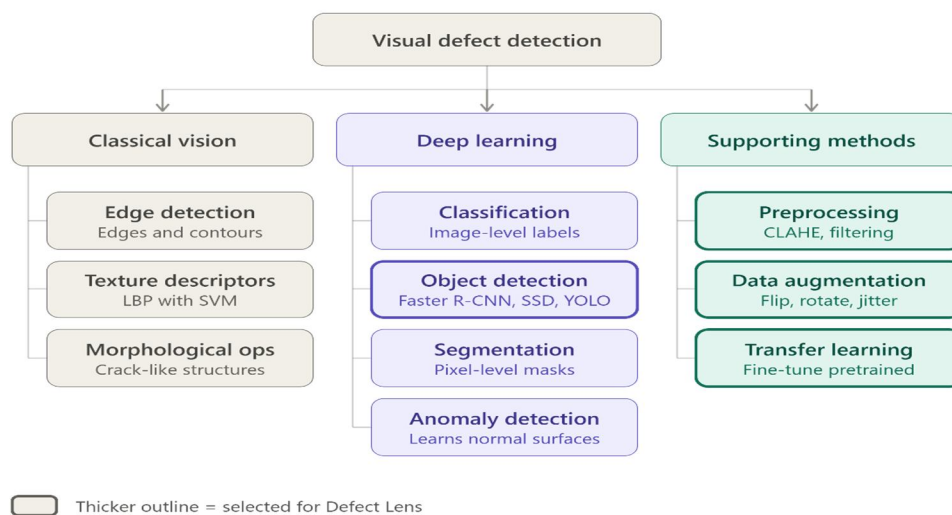


Fig. 3. Taxonomy of visual defect detection techniques, with the approaches selected for Defect Lens highlighted.

A. Classification Criteria

The taxonomy is organised along three dimensions that the literature commonly uses to distinguish inspection methods:

- 1) Feature design: whether features are manually engineered (classical vision) or learned from data (deep learning).
- 2) Output type: whether the method produces an image-level label (classification), a class with a bounding box (detection), or a pixel-level mask (segmentation).
- 3) Supervision level: whether the method needs labelled defect images (supervised) or only defect-free samples (unsupervised anomaly detection).

B. Classical Machine Vision

Classical methods rely on handcrafted features. They include edge detection [9], texture descriptors such as Local Binary Patterns paired with classifiers such as SVMs [17], and morphological operations for highlighting linear or crack-like structures. These methods are inexpensive and interpretable, but they generalise poorly across materials, textures, and lighting conditions [10].

C. Deep Learning-Based Methods

- 1) Classification: CNN-based models assign a defect label to an image or patch but do not locate the defect [20].
- 2) Object detection: Two-stage detectors such as Faster R-CNN [2] favour localisation accuracy, whereas single-stage detectors such as SSD [3] and YOLO [1], [5], [6] favour inference speed. Both predict a class and bounding box for each defect.
- 3) Segmentation: Architectures such as U-Net [19], [27] produce pixel-level masks, which allow defect extent and shape to be measured, at the cost of more demanding annotation.
- 4) Anomaly detection: Methods such as PatchCore [14], commonly benchmarked on MVTec AD [11], learn the appearance of defect-free samples and flag deviations. They reduce the annotation burden but offer limited defect-type classification.

D. Supporting Methods

These methods are not detectors themselves but strongly influence detector performance:

- 1) Image preprocessing: noise suppression, contrast enhancement (CLAHE [8]), and illumination normalisation, typically implemented with OpenCV [7].
- 2) Data augmentation: geometric and photometric transformations that improve robustness when labelled data is limited [15].
- 3) Transfer learning: fine-tuning pre-trained models on small industrial datasets [16].

E. Mapping to Defect Lens

TABLE III. Taxonomy categories and their use in Defect Lens

Category	Technique	Role in Defect Lens
Deep learning	Object detection (YOLO)	Core detector for defect class, location, and confidence
Supporting	Preprocessing (OpenCV)	Improves input quality before detection
Supporting	Data augmentation	Mitigates limited labelled data
Supporting	Transfer learning	Initialises the detector from pre-trained weights
Deep learning	Segmentation	Possible future extension for measuring defect extent
Deep learning	Anomaly detection	Possible future extension where defect samples are scarce
Classical	Edge and morphological operations	Not used as the primary detector; may assist preprocessing

This mapping shows how the proposed platform draws on one detection technique and three supporting methods, while leaving the other branches as extensions.

VI. PERFORMANCE FACTORS AFFECTING AI-BASED DEFECT DETECTION

The performance of an AI-based inspection system depends on the metrics used to evaluate it and on the conditions under which it operates. This section summarises the common evaluation metrics and the factors reported to influence detection performance, and relates them to the proposed *Defect Lens* platform.

A. Performance Evaluation Metrics

TABLE IV. Common evaluation metrics for defect detection

Metric	What It Measures	Relevance to Defect Lens
Precision	Proportion of reported defects that are correct	Indicates false-alarm burden on inspectors
Recall	Proportion of real defects that are detected	Indicates risk of missed defects
F1-score	Balance between precision and recall	Summarises per-class performance
IoU	Overlap between predicted and ground-truth boxes	Determines whether a detection is counted as correct
mAP@0.5	Mean average precision at an IoU of 0.5	Overall detection capability
mAP@0.5:0.95	Mean average precision averaged over IoU thresholds	Stricter measure of localisation quality
Inference latency / throughput	Processing time per image and images per second	Suitability for production-line use
Inspection time and consistency	Comparison with manual inspection	Practical benefit for inspectors

Segmentation methods are commonly evaluated with pixel-level IoU or the Dice coefficient, and anomaly detection methods with image-level and pixel-level AUROC [11], [14]. Aggregate scores can hide weaknesses, since a high overall mAP may still conceal poor performance on rare or small defect classes. Results should therefore be reported per class and separately for small defects. The definitions of IoU, average precision, and mAP follow the evaluation protocols of the PASCAL VOC [29] and COCO [30] benchmarks.

B. Factors Affecting Detection Performance

TABLE V. Factors affecting detection performance and proposed mitigation

Factor	Effect on Performance	Mitigation in Defect Lens
Uneven illumination	Changes apparent defect contrast between images	Illumination normalisation and CLAHE [8]; standardised lighting
Specular reflections on metallic or painted surfaces	Mask real defects or create false ones	Controlled lighting; robustness testing on reflective parts
Complex surface texture	Defects are hard to separate from normal appearance	Learned features [20]; evaluation on varied surfaces
Small or low-contrast defects	Frequently missed, especially by single-stage detectors	Adequate input resolution; multi-scale features [18]; small-defect evaluation
Limited labelled data	Poor generalisation to new samples	Transfer learning [16]; data augmentation [15]
Class imbalance	Bias toward frequent defect classes	Per-class evaluation; augmentation of rare classes
Annotation inconsistency	Noisy training and evaluation	Labelling guidelines; inspector review of detections
Intra-class variation and inter-class similarity	Confusion between similar defect types	Diverse training data; inspector confirmation
Domain shift (camera, lighting, product)	Degradation when deployment differs from training	Robustness testing; retraining on confirmed labels
Computational constraints	Limits model size and throughput	Single-stage detector; separate scalable inference service

C. Discussion

Imaging conditions are often the first limit on performance. On reflective automotive surfaces, specular highlights can hide a scratch or resemble one, and preprocessing alone is usually insufficient to remove them, so lighting design must be engineered alongside the software [10]. Data characteristics form a second group of factors. Because defects are rare, labelled datasets are small and imbalanced, and detectors tend to favour frequent classes. Transfer learning [16] and data augmentation [15] reduce this problem but do not remove it.

Defect size is a third factor. Small and low-contrast defects such as hairline cracks and fine scratches remain difficult, and multi-scale feature fusion [18] improves, but does not solve, small-defect detection. Finally, deployment constraints create a trade-off between accuracy and speed. Two-stage detectors offer strong localisation but slower inference [2], while YOLO-based detectors favour speed [1], [5], [6], sometimes at the cost of small-object sensitivity.

D. Summary

Detection performance is shaped by interacting factors across imaging conditions, data quality, and model and deployment constraints. The mitigation measures in *Defect Lens* are expected to reduce, but not eliminate, their effect. Their actual contribution will be measured through the ablation, robustness, and system-level experiments planned for the evaluation of the proposed platform.

VII. RESEARCH GAPS AND CHALLENGES

Although AI-based visual inspection has progressed considerably, several gaps remain between published research and the needs of automotive manufacturing. This section identifies the main research gaps and open challenges arising from the reviewed literature, and relates them to the design of the proposed *Defect Lens* platform.

A. Research Gaps

TABLE VI. Identified research gaps and how Defect Lens addresses them

Research Gap	Description	Relevance to Defect Lens
Limited automotive-specific datasets	Widely used benchmarks focus on steel strips or generic industrial objects, such as NEU [12] and MVTec AD [11]	Requires collection and annotation of automotive component images, supplemented by public datasets for benchmarking
Limited system-level integration	Many studies focus on detection accuracy, with less attention to data management, traceability, and inspector-facing interfaces	Provides a dashboard, backend, and database for structured inspection records
Few end-to-end deployable solutions	Few works combine classical preprocessing with modern YOLO-based detection in a complete platform tested under factory conditions	Integrates OpenCV preprocessing, YOLO detection, and storage in a single architecture
Insufficient robustness to reflective surfaces and lighting variation	Specular highlights and uneven illumination remain only partly addressed [10]	Uses illumination normalisation, controlled lighting, and robustness testing
Small-defect detection at real-time speed	Reliable detection of small, low-contrast defects while keeping low latency remains open [6], [18]	Plans small-defect evaluation and model selection balancing accuracy and latency
Limited use of inspector feedback	Human verification is rarely used as a source of training data	Stores inspector confirmations and corrections for future retraining
Lack of standard evaluation practice	Studies differ in datasets, metrics, and conditions, which makes comparison difficult	Plans per-class, small-defect, robustness, and system-level evaluation

B. Open Challenges

- 1) *Data Availability and Quality*: Defects are rare in production, so collecting enough labelled examples of each defect type is difficult. Annotation is also costly and subjective, particularly for thin or irregular defects where the boundary is ambiguous. Transfer learning [16] and data augmentation [15] help but cannot replace representative data. Synthetic data generation and few-shot learning are promising directions that remain insufficiently validated for automotive surfaces.
- 2) *Imaging Conditions and Surface Variability*: Automotive components vary in material, geometry, colour, and finish. Glossy paint and metallic surfaces produce reflections that change with viewing angle. Systems that perform well in one lighting setup may fail in another, so imaging design and software must be treated together rather than separately.
- 3) *Small and Low-Contrast Defects*: Hairline cracks, fine scratches, and early-stage corrosion occupy few pixels and have little contrast against the background. Higher input resolution and multi-scale feature fusion [18] improve sensitivity but increase computational cost, creating a direct conflict with real-time requirements.
- 4) *Generalisation and Domain Shift*: Models trained on one camera, product line, or lighting condition can degrade when any of these change. Few studies evaluate generalisation across production batches, sites, or sensor types, and continuous adaptation without costly retraining is still an open problem.
- 5) *Balancing Accuracy, Speed, and Cost*: Higher-accuracy models usually need more computation, whereas production lines require low latency and affordable hardware [2], [5], [6]. Finding a suitable operating point is application specific and must be established experimentally.
- 6) *Trust, Explainability, and Human Integration*: Inspectors must be able to trust and verify automated results. Confidence scores and bounding boxes offer limited explanation, and thresholds that suit one product may not suit another. Designing workflows in which the system assists rather than replaces the inspector, and handling disagreements between inspector and model, remain practical challenges.
- 7) *Standardisation and Reproducibility*: Differences in datasets, annotation protocols, metrics, and evaluation conditions make published results hard to compare. Inconsistent reporting of per-class and small-defect performance further limits reproducibility.

C. Future Research Directions

Based on these gaps and challenges, the following directions are identified:

- 1) *Automotive defect datasets*: building publicly available, carefully annotated datasets for automotive components under varied lighting and surface conditions.
- 2) *Reflection-robust inspection*: combining lighting design, polarisation or multi-angle imaging, and learned features to reduce the effect of specular highlights.
- 3) *Improved small-defect detection*: attention mechanisms, multi-scale fusion, and tiled or high-resolution inference with bounded latency.
- 4) *Hybrid detection and segmentation pipelines*: using detection for localisation and segmentation [19] for measuring defect extent.
- 5) *Reduced annotation dependence*: semi-supervised, self-supervised, and anomaly detection approaches [14] to reduce labelling effort.
- 6) *Domain adaptation and continual learning*: keeping performance stable across new products and conditions, using inspector-confirmed labels.
- 7) *Edge deployment and optimisation*: model compression and efficient inference for production-line hardware.
- 8) *Standard benchmarks and protocols*: consistent reporting of per-class, small-defect, and robustness results.

D. Summary

The reviewed literature shows strong progress in detection accuracy but limited progress in automotive-specific datasets, robustness to real factory conditions, reliable small-defect detection, and integration of detection into complete, traceable inspection workflows. *Defect Lens* addresses these gaps at the architectural level through preprocessing, a YOLO-based detector, structured storage, and inspector feedback.

VIII. TECHNOLOGIES USED IN AI-BASED VISUAL DEFECT DETECTION

AI-based visual defect detection systems combine frontend technologies, backend frameworks, computer-vision libraries, machine-learning models, and database systems. Together, these components process images, identify defects, manage inspection records, and present results through a user interface. The proposed *Defect Lens* architecture integrates them to support automated visual quality inspection in automotive manufacturing.

A. React.js and TypeScript

React.js [32] is a JavaScript library for building interactive user interfaces, and TypeScript adds static typing to improve maintainability and reduce interface errors. In *Defect Lens*, they provide the frontend, including image upload, visualisation of detection results, dashboard analytics, and report generation. The frontend communicates with the backend to submit inspection requests and display processed results.

B. Java and Spring Boot

Java is a general-purpose programming language, and Spring Boot [33] is a framework for building Java-based applications and REST services. In the proposed architecture, they form the backend layer. This layer handles incoming requests, coordinates application workflows, manages authentication and database operations, and communicates with the AI processing component.

C. Python and OpenCV

Python provides a flexible environment for implementing computer-vision and machine-learning workflows. OpenCV [7] is a computer-vision library that supports operations such as resizing, colour conversion, noise reduction, and contrast adjustment, including CLAHE [8]. In *Defect Lens*, these technologies preprocess automotive component images before defect analysis.

D. YOLO-Based Defect Detection

You Only Look Once (YOLO) is a family of object-detection models that identify objects and estimate their locations in a single forward pass [1], [6]. In *Defect Lens*, YOLO-based detection is intended to identify and localise visible defects such as scratches, dents, and cracks, using an implementation such as Ultralytics [22]. The model's performance depends on its configuration, training data, image quality, and evaluation conditions.

E. PostgreSQL Database

PostgreSQL [34] is an open-source relational database management system that supports structured storage and querying. In *Defect Lens*, it is proposed for maintaining inspection records, defect information, inspection logs, and report-related metadata. The backend retrieves these records to support inspection history, analytics, and reporting.

F. Integration of Technologies

TABLE VII. Technologies and their roles in Defect Lens

Layer	Technology	Role in Defect Lens
Presentation	React.js, TypeScript	Image upload, result visualisation, dashboard, reports
Application	Java, Spring Boot	Request handling, workflow coordination, authentication, database access
Image processing	Python, OpenCV	Resizing, noise reduction, contrast enhancement
AI detection	YOLO	Defect classification, localisation, and confidence scoring
Data	PostgreSQL	Inspection records, defect data, logs, report metadata

The technologies operate as interconnected components. React.js and TypeScript provide the user interface, Java and Spring Boot manage application logic and request processing, Python, OpenCV, and YOLO perform preprocessing and defect analysis, and PostgreSQL stores the inspection information. In the intended workflow, users submit images, the system analyses them, inspection information is stored, and processed results are presented through the dashboard and reports.

G. Summary

The proposed Defect Lens architecture combines frontend development, backend application processing, computer vision, AI-based defect detection, and relational database management. Each technology contributes a distinct function: presenting results to inspectors, coordinating workflows, preparing images, detecting defects, and storing records for traceability. Separating these responsibilities allows each component to be developed, scaled, and updated independently. The effectiveness of the integrated system must still be established through implementation and the systematic evaluation described earlier.

IX. FUTURE SCOPE

AI-based visual defect detection offers considerable opportunities to improve automated quality inspection in automotive manufacturing.

Future work on *Defect Lens* and similar systems can focus on improving detection accuracy, supporting a wider range of defect categories, and integrating intelligent inspection into manufacturing workflows. The directions below follow from the research gaps and challenges identified in the previous section.

A. Improved Defect Detection

Future research can investigate advanced deep-learning architectures to improve the identification of small scratches, fine cracks, dents, and other surface defects. Attention mechanisms, multi-scale feature fusion [18], and high-resolution or tiled inference are candidate approaches for small-defect sensitivity. Model optimisation and suitable preprocessing may further improve performance under varying inspection conditions, with their effect confirmed through ablation studies.

B. Integration of Object Detection and Segmentation

Combining object detection with image segmentation [19] could provide both defect localisation and detailed boundary information. In such a hybrid pipeline, a YOLO-based detector [1], [6] would first localise candidate defects, and a segmentation model would then delineate their extent. This could support more comprehensive analysis of irregular defects such as long scratches and corrosion patches.

C. Expansion of Training Datasets

Larger and more diverse automotive defect datasets are needed, covering different component types, materials, colours, lighting conditions, and defect categories. Data augmentation [15], transfer learning [16], and suitable sampling strategies can help address limited and imbalanced training data. Semi-supervised and anomaly detection approaches [14] may further reduce dependence on annotated defect images.

D. Defect Severity Assessment

Future versions of *Defect Lens* could estimate defect severity from characteristics such as defect area, dimensions, and location on the component. Such estimates would require clearly defined severity criteria, validation against expert judgement, and agreement with applicable quality standards before being used in actual quality-control decisions.

E. Predictive Quality Analytics

Inspection records accumulated in the database could be analysed to identify recurring defect patterns and trends in product quality. Machine-learning techniques may help explore relationships between defect categories, component types, production batches, and inspection history, provided sufficient reliable data are available. Such analysis could support earlier identification of process problems.

F. Explainable Artificial Intelligence

Explainable AI techniques such as Grad-CAM [23] and SHAP [24] could help users understand aspects of model predictions. Visual explanations can show which image regions influenced a detection, and feature-contribution analysis can support the investigation of incorrect predictions. This may improve transparency and inspector trust, although explanation methods have limitations and should themselves be validated.

G. Real-Time Inspection and System Integration

Future work could investigate optimising model inference through compression and efficient deployment on production-line hardware, and integrating AI-based inspection with manufacturing workflows. Improvements in processing speed, communication with production systems, and inspection-data management may support more efficient quality-control operations.

TABLE VIII. Future research directions and expected benefits

Direction	Focus	Expected Benefit
Improved defect detection	Small-defect sensitivity, model optimisation	Fewer missed small or low-contrast defects
Detection and segmentation	Hybrid pipeline [19]	Localisation plus detailed defect boundaries
Dataset expansion	Diverse automotive data, augmentation, semi-supervised methods	Better generalisation across components and conditions
Severity assessment	Area, dimensions, location	Support for prioritising defects, subject to validation
Predictive analytics	Trend analysis of inspection records	Earlier detection of recurring quality problems
Explainable AI	Grad-CAM [23], SHAP [24]	Greater transparency and easier error analysis
Real-time integration	Inference optimisation, workflow integration	Efficient, production-ready inspection

H. Summary

Future development of Defect Lens could focus on improved defect localisation, combined detection and segmentation, diverse training datasets, severity assessment, predictive analytics, explainability, and efficient system integration. These directions offer opportunities for further research into reliable AI-powered visual inspection for automotive manufacturing, and each must be validated through implementation and systematic evaluation.

X. DISCUSSION AND IMPLICATIONS FOR AUTOMOTIVE MANUFACTURING

This section synthesises the findings of the survey and considers what they imply for automotive manufacturing, and for the design and adoption of *Defect Lens*. Because *Defect Lens* is a proposed platform that has not yet been implemented or evaluated, the implications below are expected benefits and considerations, not demonstrated results.

A. Discussion of Key Findings

- 1) *No single technique meets every inspection requirement.* The comparison of classification, object detection, segmentation, and anomaly detection (Section IV) shows that each method addresses a different need. Classification gives defect categories, detection gives locations, segmentation gives boundaries, and anomaly detection reduces dependence on labelled defects. YOLO-based detection [1], [5], [6] appears to be the most practical starting point for an inspector-assist platform, because it provides class, location, and confidence at speeds compatible with production use. It leaves gaps in boundary measurement and rare-defect handling, which the future directions in Section IX address.
- 2) *Performance depends on more than the model.* Illumination, surface reflectance, image resolution, dataset quality, and domain shift all affect results (Section VI). In many industrial settings, imaging design and data quality may therefore matter as much as the choice of detector. A strong model combined with poor lighting or unrepresentative training data is unlikely to perform reliably.
- 3) *System integration is as important as detection accuracy.* The literature gives limited attention to traceability, data management, and inspector-facing interfaces. In *Defect Lens*, the backend, database, and dashboard are part of the design, because a detection is only useful in production if it can be reviewed, recorded, and acted upon.
- 4) *Human oversight remains essential.* Small and low-contrast defects, reflective surfaces, and unfamiliar conditions remain difficult for current models. This supports the decision-support role of *Defect Lens*, in which inspectors confirm or correct detections. Their corrections can also serve as a source of training data.

B. Implications for Automotive Manufacturing

TABLE IX. Implications of AI-based visual inspection for automotive manufacturing

Area	Potential Implication	Condition or Consideration
Quality assurance	More consistent inspection, with less variation from fatigue and subjective judgement	Requires validation against inspector performance
Early defect detection	Earlier identification of defects can reduce rework and material waste	Depends on inspection being placed at suitable process stages
Traceability	Structured, searchable records of each inspection and defect	Requires reliable linkage to component and batch identifiers
Process improvement	Defect trends may reveal recurring process or supplier problems	Needs sufficient, reliable data over time
Inspector productivity	Highlighted defects may reduce time spent on routine checks	Must be measured; inspectors need to trust the output
Cost	Possible reductions in rework, scrap, and escaped defects	Offset by costs of cameras, lighting, computing, and maintenance
Scalability	Modular architecture allows the AI service to be updated independently	Requires model monitoring and retraining processes

- 1) *Quality and Consistency:* Automated inspection can apply the same criteria to every image, which may reduce the variability associated with manual inspection. This benefit is only realised if the model is validated against agreed defect definitions and reviewed by experienced inspectors.
- 2) *Cost and Waste Reduction:* Detecting defects earlier may limit the value added to a defective part before it is discovered, reducing rework and waste. However, these savings must be weighed against the costs of imaging hardware, lighting, computing infrastructure, labelling effort, and ongoing maintenance. Economic benefit should be assessed case by case.
- 3) *Data-Driven Quality Management:* Structured inspection records make it possible to analyse defect frequency by component, batch, shift, or supplier. Over time, this could support predictive quality analytics (Section IX-E) and feed into broader quality-management processes.
- 4) *Workforce and Workflow:* The platform is intended to assist inspectors, not replace them. Its practical value will depend on how well it fits existing workflows, how clearly results are presented, and whether inspectors trust it. Training and gradual introduction alongside manual inspection are likely to be necessary.

C. Practical Considerations for Adoption

- 1) *Imaging infrastructure.* Camera placement, lighting, and fixturing should be engineered with the software. Controlled and consistent imaging conditions are among the most effective ways to improve reliability, especially for reflective surfaces [10].
- 2) *Data collection and labelling.* Organisations need representative images of their own components, with consistent annotation guidelines. Defect definitions should be agreed with quality engineers before labelling begins.
- 3) *Validation before deployment.* Models should be tested on data from the target production line, with attention to per-class and small-defect performance and to false-accept and false-reject rates, before being used in decisions.
- 4) *Monitoring and maintenance.* Performance can drift as products, suppliers, or imaging conditions change. Periodic evaluation and retraining, using inspector-confirmed labels, are needed to maintain reliability.
- 5) *Integration and security.* Connecting inspection to production systems requires attention to data security, access control, and audit logging, which the backend design includes through authentication and role-based access.
- 6) *Standards and responsibility.* Automotive quality processes are governed by established quality-management standards. Any AI-based inspection should be introduced within these frameworks, with clear accountability for final accept/reject decisions.

XI. CONCLUSION

AI-based visual defect detection has emerged as a promising approach for improving quality inspection in automotive manufacturing. This survey reviewed AI-based techniques, particularly YOLO-based object detection, image preprocessing, and computer vision methods, for identifying defects such as scratches, dents, cracks, and corrosion on automotive components.

The discussion highlights that detection performance depends on several factors, including image quality, lighting conditions, training data diversity, defect size, and processing requirements. Challenges such as limited labelled datasets, subtle defects, variations in surface appearance, and prediction reliability must be addressed before these systems can be adopted in manufacturing environments. The proposed Defect Lens platform aims to integrate React.js and TypeScript for the user interface, Java and Spring Boot for backend processing, Python and OpenCV for image processing, YOLO for defect detection, and PostgreSQL for storing inspection records. This integrated approach aims to identify potential defects, organise inspection information, and present results through a user-friendly dashboard.

Future improvements could include defect severity assessment, enhanced detection of small defects, explainable AI, and integration with manufacturing workflows. The actual effectiveness of Defect Lens must be established through implementation, testing, and evaluation using suitable automotive defect images.

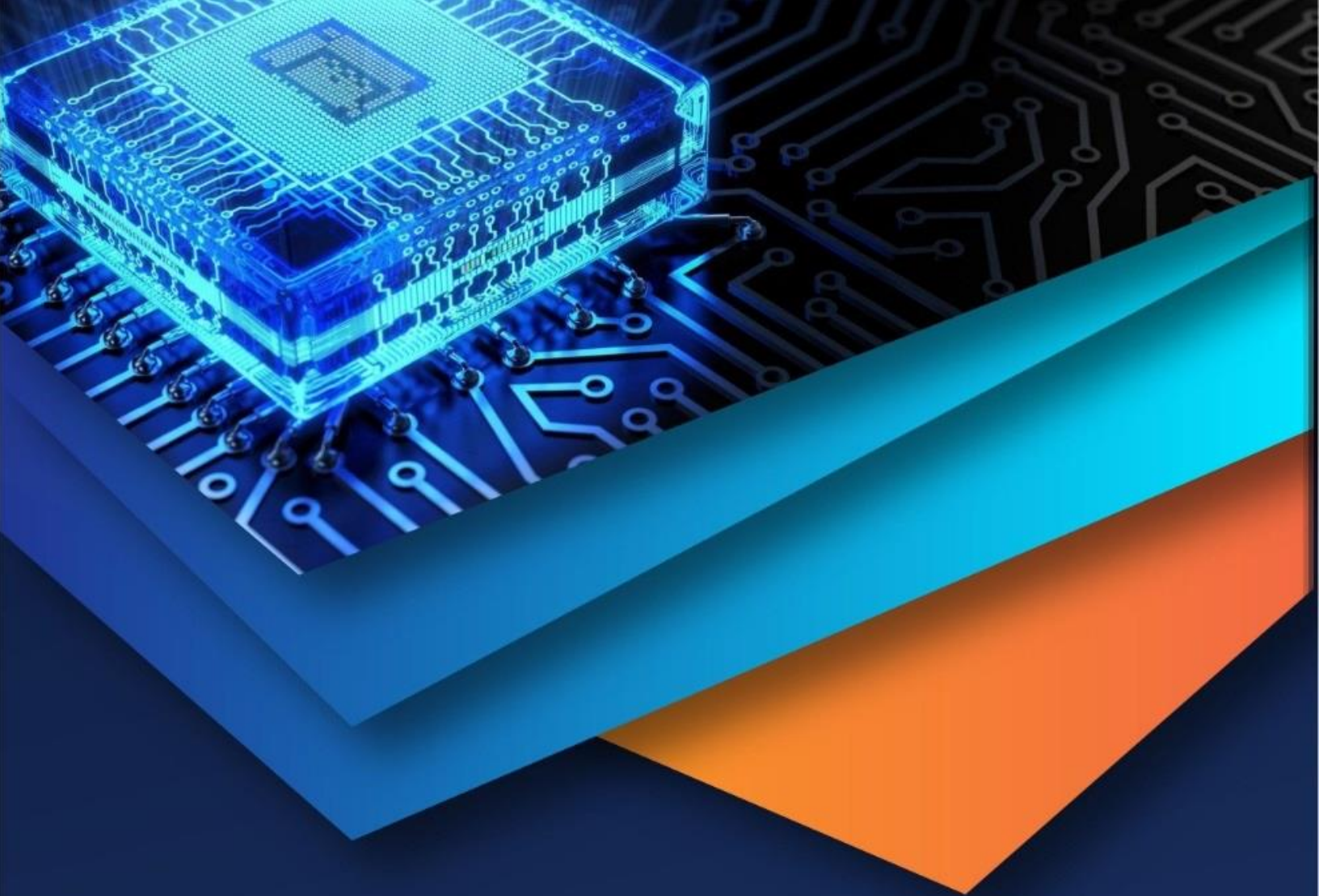
Overall, AI-based visual inspection can support more consistent and efficient automotive quality-control processes when combined with reliable data, appropriate validation, and human supervision.

REFERENCES

- [1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.
- [2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
- [3] W. Liu et al., "SSD: Single shot multibox detector," in Proc. Eur. Conf. Comput. Vis. (ECCV), 2016, pp. 21–37.
- [4] [4] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," arXiv:2004.10934, 2020.
- [5] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2023.
- [6] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS," Mach. Learn. Knowl. Extr., vol. 5, no. 4, pp. 1680–1716, 2023.
- [7] G. Bradski, "The OpenCV Library," Dr. Dobb's J. Softw. Tools, 2000.
- [8] K. Zuiderveld, "Contrast limited adaptive histogram equalization," in Graphics Gems IV. San Diego, CA, USA: Academic Press, 1994, pp. 474–485.
- [9] J. Canny, "A computational approach to edge detection," IEEE Trans. Pattern Anal. Mach. Intell., vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.
- [10] D. Zimmermann et al., "Visual-based defect detection and classification approaches for industrial applications—A survey," Sensors, vol. 20, no. 5, Art. no. 1459, 2020.
- [11] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "MVTec AD—A comprehensive real-world dataset for unsupervised anomaly detection," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2019, pp. 9592–9600.
- [12] K. Song and Y. Yan, "A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects," Appl. Surf. Sci., vol. 285, pp. 858–864, 2013.
- [13] Y. He, K. Song, Q. Meng, and Y. Yan, "An end-to-end steel surface defect detection approach via fusing multiple hierarchical features," IEEE Trans. Instrum. Meas., vol. 69, no. 4, pp. 1493–1504, Apr. 2020.
- [14] K. Roth et al., "Towards total recall in industrial anomaly detection," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2022, pp. 14318–14328.
- [15] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," J. Big Data, vol. 6, Art. no. 60, 2019.
- [16] S. J. Pan and Q. Yang, "A survey on transfer learning," IEEE Trans. Knowl. Data Eng., vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
- [17] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 7, pp. 971–987, Jul. 2002.
- [18] T.-Y. Lin et al., "Feature pyramid networks for object detection," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2017, pp. 2117–2125.
- [19] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent. (MICCAI), 2015, pp. 234–241.
- [20] R.-G. Tulbure et al., "A review on modern defect detection models using DCNNs—Deep convolutional neural networks," Alexandria Eng. J., vol. 61, no. 10, 2022.
- [21] S. B. Jha and R. F. Babiceanu, "Deep CNN-based visual defect detection: Survey of current literature," Comput. Ind., vol. 148, Art. no. 103911, Jun. 2023, doi: 10.1016/j.compind.2023.103911.
- [22] G. Jocher et al., "Ultralytics YOLO," software, 2023. [Online]. Available: <https://github.com/ultralytics/ultralytics>
- [23] R. R. Selvaraju et al., "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), 2017, pp. 618–626.
- [24] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2017, pp. 4765–4774.
- [25] Y. Chen, Y. Ding, F. Zhao, E. Zhang, Z. Wu, and L. Shao, "Surface defect detection methods for industrial products: A review," Appl. Sci., vol. 11, no. 16, Art. no. 7657, 2021, doi: 10.3390/app11167657.



- [26] H. Huang and K. Zhu, "Automotive parts defect detection based on YOLOv7," *Electronics*, vol. 13, no. 10, Art. no. 1817, 2024, doi: 10.3390/electronics13101817.
- [27] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "A review on deep learning techniques applied to semantic segmentation," *arXiv:1704.06857*, 2017.
- [28] P. Carvalho, M. Lafou, A. Durupt, A. Leblanc, and Y. Grandvalet, "Detecting visual anomalies in an industrial environment: Unsupervised methods put to the test on the AutoVI dataset," *Comput. Ind.*, vol. 163, Art. no. 104151, 2024, doi: 10.1016/j.compind.2024.104151.
- [29] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.
- [30] T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 740–755.
- [31] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proc. IEEE*, vol. 111, no. 3, pp. 257–276, Mar. 2023.
- [32] Meta Platforms, Inc., "React documentation." [Online]. Available: <https://react.dev> (accessed: Oct. 13, 2026).
- [33] VMware/Broadcom, "Spring Boot reference documentation." [Online]. Available: <https://docs.spring.io/spring-boot/> (accessed: Oct. 13, 2026).
- [34] PostgreSQL Global Development Group, "PostgreSQL documentation." [Online]. Available: <https://www.postgresql.org/docs/> (accessed: Oct. 13, 2026).



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)