Efficiency And Reliability Improvement in A Raspberry Pi Based Pipeline Inspection Robot

Gagan Murthy[1], Abhinav Dinkar[2], Hitha R Shetty[3], Manoj B[4], Soni M[5]
[1,2,3,4,5]*Department of Electrical and Electronics Engineering, Dayananda Sagar College of Engineering, Bengaluru, India*

***Abstract*—Pipeline inspection robots are commonly used to detect cracks and structural defects inside industrial pipelines. In low-cost embedded systems, deep-learning models can create high computational load and unstable detection outputs, reducing both efficiency and reliability. This work evaluates a Raspberry Pi based vision inspection robot and applies two deployment-level optimizations: INT8 quantization of a baseline FP32 YOLO model and multi-frame confidence confirmation. Quantization reduces inference latency and CPU usage, while temporal confirmation reduces false positives caused by single-frame instability. Results from controlled experiments show substantial runtime improvement with minor confidence reduction and improved detection stability during continuous inspection. The combined method provides a practical balance between efficiency and reliability for low-cost embedded pipeline inspection.**

***Index Terms*—Detection reliability, Edge AI, INT8 quantization, Pipeline inspection, Raspberry Pi, YOLO.**

I. INTRODUCTION

Pipeline infrastructure is widely used in oil and gas transportation, water distribution, chemical processing, and power generation systems. Over long operating durations, pipelines experience corrosion, pressure fluctuations, vibration, and material fatigue. These factors can produce internal cracks and surface defects that are difficult to detect externally. If not identified early, such defects may progress into leakage, environmental hazards, and costly service disruption. Traditional inspection approaches often require manual intervention or partial disassembly, increasing maintenance cost and downtime. Mobile inspection robots reduce these limitations by travelling through pipes and acquiring visual data from internal surfaces. Vision based systems are particularly useful because they provide direct evidence of crack morphology and location. Deep-learning detection models, especially YOLO family architectures, have improved automated crack identification compared with classical thresholding and edge-based approaches. However, in practical low-cost deployments, embedded hardware constraints become critical. Platforms such as Raspberry Pi provide portability and low power consumption, but have limited CPU resources compared with GPU workstations. Running full-precision detectors on such devices increases inference latency and CPU utilization. A second practical issue is temporal instability in framewise predictions. During movement inside pipes, slight illumination changes, motion blur, or sensor noise can cause confidence fluctuations. When crack decisions are made from single frames, false positives and inconsistent alerts may occur. Reliable inspection therefore requires both computational efficiency and temporal stability.

This work investigates two techniques that do not require mechanical redesign or new neural architectures: INT8 quantization to improve computational efficiency and multi-frame confidence confirmation to improve reliability. The central objective is to analyze their trade-off and combined impact for embedded real-time inspection.

II. RELATED WORK

Pipeline inspection research broadly spans robot mobility, sensing systems, and defect detection algorithms. On the mobility side, articulated and wheeled robots have been developed for curved and vertical pipe traversal [1–4]. Biomimetic and spiral leg-wheel mechanisms have also been proposed to improve manoeuvrability and traction in constrained

pipelines [5,6]. Modular multi-unit systems further support low-pressure gas pipeline operations [7].

For crack detection, early systems relied on classical image-processing methods such as Canny edge detection and thresholding [9]. These methods are lightweight but sensitive to noise and illumination changes. Recent work has shifted to deep-learning models such as improved YOLOv5 pipelines and pretrained model integration for stronger defect localization and classification [10–12]. AI-assisted inspection workflows are also reported in related non-destructive testing domains [13].

Additional work addresses sensing and visibility constraints in dark pipeline interiors. Bionic visual perception for low-illumination scenarios [14], embedded in-line robots for industrial monitoring [15], and magnetic-inductive tracking approaches [16] have been investigated. These contributions improve inspection feasibility in real field conditions.

Despite significant advances, many studies are validated on high-performance computing setups. Fewer investigations quantify both runtime efficiency and temporal detection stability on constrained embedded processors. The present work targets this gap by combining quantization and temporal confirmation in a practical Raspberry Pi based inspection system.

III. SYSTEM OVERVIEW

A. Robot Platform

The developed platform is a wheeled robot intended for operation inside an 8-inch diameter pipeline. Its structure uses a rigid chassis and wheel-driven motion for forward and backward traversal. The mechanical design remains intentionally simple to maintain low cost and ease of deployment.

A Raspberry Pi serves as the central processing unit. A front-mounted Pi camera captures forward-looking frames of the internal pipe wall. Because the interior environment is typically dark, onboard lighting is used to maintain visible surface features. For fair comparison between model settings, illumination conditions were kept as consistent as possible during test runs.

B. Processing and Logging Pipeline The inspection pipeline follows:

Camera → Frame Capture → Model Inference → Detection Output → Data Logging.

Each processed frame produces crack bounding boxes and confidence scores. The system also logs inference timestamp, CPU utilization, and per-frame processing time. The platform supports both FP32 and INT8 model execution modes for direct comparative testing under identical input conditions.

C. Monitoring Interface

A lightweight monitoring interface was used to visualize live inspection status. The interface displays camera feed, crack counts, frame identifiers, confidence levels, and runtime metrics such as CPU usage and inference time. This consolidated view improves operator awareness and supports post-run analysis.

IV. PROPOSED METHOD

The crack detection problem can be formulated as a frame-wise classification task with temporal consistency constraints. Let I_i denote the i-th input frame and $f(I_i)$ represent the model output confidence for crack detection.

In a conventional single-frame approach, a detection is confirmed if

$$f(I_i) \geq \tau \quad (1)$$

where τ is a predefined confidence threshold. However, due to temporal fluctuations, this approach may result in unstable detections. To address this, the proposed method introduces a temporal aggregation function over N consecutive frames:

$$C_{avg} = \frac{1}{N} \sum_{i=1}^{N} f(I_i) \quad (2)$$

A detection is confirmed only if

$$C_{avg} \geq \tau \quad (3)$$

This formulation ensures that only persistent detections across multiple frames are considered valid, thereby improving reliability.

A. INT8 Quantization for Efficiency

The baseline YOLO crack detector uses FP32 precision. While numerically expressive, FP32

increases memory bandwidth and arithmetic cost on CPU-based embedded platforms. To reduce computational load, the trained model is converted to INT8 precision.

Inference speed is computed as

$$FPS = \frac{1000}{T_{inf}} \quad (4)$$

where T_{inf} is the average inference time in milliseconds. CPU improvement is measured by

$$CPU_{reduction}(\%) = \left(\frac{CPU_{FP32} - CPU_{INT8}}{CPU_{FP32}} \right) \times 100 \quad (5)$$

B. Multi-Frame Confidence Confirmation

Single-frame decisions can be unstable when confidence fluctuates due to noise or lighting variation. To improve reliability, crack confirmation is performed over a short sequence of consecutive frames. Let C_i denote the crack confidence in frame i. For a window of N frames,

$$C_{avg} = \frac{1}{N} \sum_{i=1}^{N} C_i \quad (6)$$

A crack is confirmed only when C_{avg} exceeds a predefined threshold. In this study, N =5 provided a practical balance between stability and response delay. False-positive reduction is calculated as

$$FP_{reduction}(\%) = \left(\frac{FP_{single} - FP_{multi}}{FP_{single}} \right) \times 100 \quad (7)$$

C. Joint Trade-Off Perspective

Quantization improves throughput and resource efficiency, while multi-frame confirmation improves decision stability. Evaluating them together is important: speed alone cannot ensure trustworthy inspection outputs, and reliability methods alone may be impractical if runtime load remains excessive.

V. EXPERIMENTAL SETUP

Two experiments were performed. The first compared FP32 and INT8 models for computational efficiency. The second compared single-frame detection with fiveframe confirmation for reliability. For both experiments, the same trained model, resolution settings, detection threshold, and inspection video sequences were used. Each condition was run for several hundred frames under controlled indoor lighting. The recorded metrics were average inference time, CPU utilization, average detection confidence, confirmed detections, false positives, and false negatives.

A. Quantization Evaluation Procedure

For quantization evaluation, FP32 and INT8 versions of the same detector were deployed sequentially on the same Raspberry Pi unit. No architecture changes, retraining modifications, or threshold differences were introduced between the two modes. This ensured that observed performance differences were attributable to numeric precision and runtime computation path. Each model was executed continuously over identical inspection frame sequences. Per-frame inference time was captured and aggregated into an average value for each configuration. CPU utilization was monitored throughout continuous operation, and confidence output for detected cracks was averaged across the sequence. Each experiment was repeated and averaged.

B. Reliability Evaluation Procedure

Reliability was evaluated using two decision strategies:

(i) immediate single-frame confirmation and (ii) five frame average confirmation. For both strategies, video inputs and detector thresholds remained fixed. In the single-frame case, a crack was reported as soon as confidence crossed threshold in one frame. In the multi frame case, confidence values from five consecutive frames were averaged before confirming a crack.

Detection logs were manually reviewed to count confirmed detections, false positives, and false negatives. This comparison quantified whether temporal confirmation reduces unstable reporting during robot motion.

Table I: Quantization Performance Comparison

Model	Avg. Inference Time (ms)	Avg. CPU Usage (%)	Avg. Detection Confidence
FP32	186	88	0.81
INT8	104	63	0.78

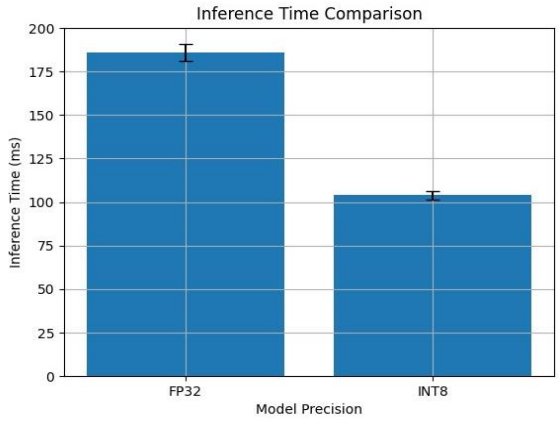

Figure 1: Comparison of average inference time for FP32 and INT8 model execution across multiple runs. Error bars indicate standard deviation.

VI. RESULTS AND DISCUSSION

A. Quantization Performance

Table I summarizes computational comparison between FP32 and INT8 execution from the extracted source results. INT8 reduced average inference time by about 44% and CPU usage by about 25%. Using (1), throughput increased from approximately 5.3 FPS to 9.6 FPS. The confidence decreases from 0.81 to 0.78 is modest and remains acceptable for practical crack detection. Fig. 1 illustrates the comparison of average inference time between FP32 and INT8 model execution across multiple runs. The INT8 model demonstrates a substantial reduction in inference latency, consistent with the reported average decrease of approximately 44%. Error bars represent standard deviation across repeated trials, indicating stable performance improvements under quantized execution.

Fig. 2 presents CPU utilization comparison between FP32 and INT8 execution modes. The quantized model reduces average CPU load by approximately 25%, thereby increasing computational headroom for concurrent processes and improving thermal stability during prolonged operation on the embedded platform.

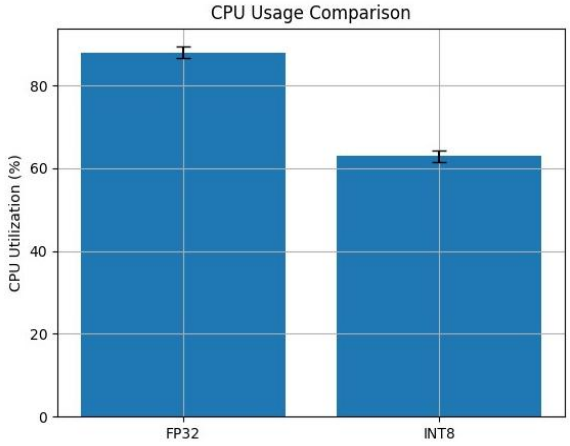

Figure 2: CPU utilization comparison between FP32 and INT8 execution modes.

Table II: Multi-Frame Confirmation Comparison

Method	Confirmed Detections	False Positives	False Negatives
Single-frame detection	39	13	6
Five-frame confirmation	36	8	7

B. Multi-Frame Reliability

Table II report's reliability comparison between single frame and five-frame confirmation. From (4), false positives decrease by roughly 38% with multi-frame confirmation. A slight increase in false negatives is observed, which is expected under stricter temporal confirmation. In inspection practice, reduced false alarms often improves usability and trust in automated reporting.

Fig. 3 compares false-positive and false-negative counts between single-frame detection and multi-frame confirmation strategies. A reduction of approximately 38% in false positives is observed with temporal confirmation, indicating improved detection stability. A marginal increase in false negatives is noted, reflecting a stricter confirmation criterion.

Fig. 4 illustrates temporal smoothing of detection confidence using a five-frame averaging window. Raw confidence values exhibit fluctuations due to noise and illumination variation, whereas the averaged confidence curve demonstrates improved stability, enabling more consistent crack confirmation decisions.

Figure 3: Comparison of false-positive and false-negative counts for single-frame detection and multi-frame confirmation.

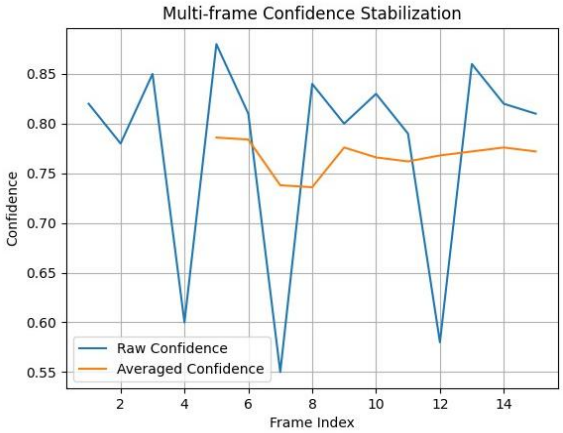

Figure 4: Temporal smoothing of detection confidence using a five-frame averaging window.

Table III: Derived Trade-Off Metrics

Metric	Value
Inference-time reduction (FP32 → INT8)	≈ 44%
CPU-usage reduction (FP32 → INT8)	≈ 25%
Throughput gain (5.3 FPS → 9.6 FPS)	≈ 81%
False-positive reduction (single → multi-frame)	≈ 38%
Confidence changes after quantization	0.81 → 0.78

C. Derived Performance View

To present the trade-off more clearly, Table III reports derived metrics computed from Tables I and II. Throughput gain and CPU savings quantify runtime benefits, while false-positive reduction and confidence shift indicate reliability implications.

D. Combined Effect and Practical Implications

Applying both methods together produced smoother runtime behavior and more stable crack reporting. Lower CPU load reduced processing stress on the embedded platform, while temporal averaging reduced fluctuating frame-to-frame alerts. The overall behavior demonstrates that deployment-level optimization can yield meaningful gains without changing robot mechanics or introducing complex new model architectures. From a system engineering perspective, the results support a staged deployment strategy for low-cost inspection robots. Quantization should first be used to secure runtime headroom and predictable CPU behavior. Temporal confirmation can then be tuned according to inspection priorities: a shorter window for faster alerts, or a longer window for stricter reliability. The measured outcomes also show that moderate confidence reduction after quantization does not necessarily translate into unacceptable inspection quality. When temporal consistency checks are added, overall reporting quality can improve despite reduced numeric precision. This is useful for real deployments where operational stability is often more valuable than maximizing isolated frame confidence.

VII. EXTENDED DISCUSSION

A. Operational Deployment Considerations

For indoor pilot runs, controlled lighting and moderate robot speed produced stable detector outputs. In field settings, periodic reflections, dust, or wet surfaces may cause transient confidence dips. A fixed five-frame window works well in the tested setup, but adaptive windows can further improve robustness. For example, when motion blur increases, increasing the confirmation window by one or two frames can reduce instability; when motion is steady, returning to five frames preserves responsiveness.

Another practical consideration is CPU thermal behavior on Raspberry Pi during long runs. Quantization reduces average load, which indirectly helps thermal stability and minimizes frequency throttling risk. This effect supports more predictable real-time inspection over extended pipeline segments.

B. Inspection-Quality Perspective

Inspection systems are judged not only by frame-level precision but also by operator trust and consistency of alarms. In this study, the multi-frame method produced fewer false alerts, which improves usability during long monitoring sessions. Even with a slight increase in false negatives, the net behavior is more stable for maintenance planning, especially when repeated passes are available.

In scenarios where missing any possible crack is unacceptable, confirmation thresholds can be tuned downward while preserving temporal averaging. Conversely, when false alarms are costly, thresholds can be raised slightly with the same confirmation window. This demonstrates that the proposed method is configurable without retraining the model.

C. Scalability to Other Embedded Platforms

Although this study uses Raspberry Pi, the framework is hardware-agnostic. Any CPU-limited edge platform running a quantized detector can adopt the same confirmation strategy. The key requirement is access to consecutive confidence outputs and low-overhead buffering for temporal averaging. Therefore, the approach can be extended to other mobile inspection robots and industrial edge devices.

VIII. IMPLEMENTATION NOTES FOR REPRODUCIBILITY

A. Runtime Configuration

The embedded runtime was configured to execute a continuous inference loop with frame capture, preprocessing, model inference, and logging performed in sequence. For practical deployment, maintaining deterministic preprocessing and fixed confidence thresholds is important when comparing FP32 and INT8 modes. Any change in preprocessing pipeline may influence confidence values and can mask the true effect of quantization.

B. Data Logging Strategy

A structured logging format was used for each frame: timestamp, model mode, confidence score, inference time, and CPU usage. This structure supports postrun filtering and allows direct pairing of detected events across single-frame and multi-frame confirmation pipelines. In deployment settings, this log can be synchronized with robot position estimates to produce location-aware crack reports.

C. Guidelines for Parameter Tuning

Based on observations from the extracted manuscript, three parameters are most influential: confidence threshold, temporal window size N, and frame rate. Increasing threshold and window size reduces false positives but may increase missed detections and response delay. Lowering threshold improves recall but may increase unstable alerts. A practical workflow is to first tune threshold for acceptable sensitivity, then tune N to suppress temporal noise while preserving operator response needs.

D. Practical Use in Inspection Workflows

For routine maintenance screening, the combined approach can run in an assistive mode where potential defects are flagged for review rather than immediately treated as critical faults. For urgent safety checks, threshold and confirmation windows can be adjusted to prioritize early detection. This adaptability is one reason the proposed framework is suitable for low-cost deployments where one static setting may not fit all operating scenarios.

IX. ABLATION AND SCENARIO-BASED ANALYSIS

A. Ablation View of the Pipeline

To better understand contribution of each component, the inspection workflow can be viewed as three progressively constrained configurations: baseline FP32 single-frame inference, INT8 single-frame inference, and INT8 with five-frame confirmation. The first configuration gives a reference for model confidence under highest precision but also highest CPU load. The second isolates computational effects of precision reduction. The third adds temporal decision logic to stabilize outcomes.

From this ablation perspective, quantization is primarily responsible for resource efficiency, while multi-frame confirmation is primarily responsible for reliability stabilization. The results indicate that these contributions are largely complementary rather than conflicting. In other words, quantization creates runtime headroom, and temporal confirmation consumes only modest additional logic while improving reporting consistency.

B. Scenario 1: Low-Texture Pipe Surfaces

In low-texture or relatively clean pipe sections, crack cues are weaker and detector confidence values may remain close to threshold. Under these conditions, singleframe decisions are sensitive to minor noise. Temporal averaging acts as a filter, allowing persistent weak signals to accumulate while transient spikes are suppressed. This behavior is consistent with reduced false positives observed in the extracted results.

C. Scenario 2: Motion-Induced Blur

During faster wheel motion or brief mechanical vibration, motion blur may reduce instantaneous crack saliency. In such frames, detector confidence can dip momentarily even when a real crack remains in view. A short temporal window compensates for this by aggregating evidence over nearby frames. The trade-off is a small detection delay; however, for pipeline inspection where the robot travels continuously, this delay remains acceptable if it improves reporting stability.

D. Scenario 3: Illumination Fluctuation

Pipeline lighting may vary due to LED angle shifts, reflective surfaces, or localized dark zones. In a purely single-frame pipeline, these fluctuations can trigger unstable confidence crossings. Multi-frame confirmation reduces this effect by requiring consistency over time.

For practical operation, maintaining approximately uniform onboard lighting and using temporal confirmation together provides a stronger reliability baseline than either method alone.

E. Discussion on Practical Threshold Selection

A practical thresholding workflow can be summarized as follows. First, establish a baseline threshold using FP32 outputs that balances recall and false alarms in representative clips. Second, switch to INT8 and recheck the same clips to account for minor confidence shift. Third, apply multi-frame confirmation and tune only one variable at a time (window size or threshold) to avoid coupled effects. This sequential approach simplifies commissioning in industrial settings. In many maintenance contexts, operators prefer stable actionable alerts over noisy high-frequency triggers. The extracted results support this preference: although confirmed detections reduced slightly in multi-frame mode, false positives reduced more substantially. This can lower unnecessary inspection stops and improve overall operator trust in the robotic system.

F. Computational Budget Interpretation

For embedded deployments, average CPU usage is only part of the story; margin to peak CPU load is also important. With FP32 operation near high utilization, temporary processing spikes can disrupt frame timing. INT8 reduces average usage and improves margin, making timing behavior more predictable. This predictability is important when synchronized logs, video review, or downstream control tasks must run concurrently on the same processor.

The observed FPS improvement from approximately 5.3 to 9.6 indicates that frame processing becomes less of a bottleneck relative to robot motion. In practical terms, higher throughput supports denser visual sampling per meter of pipe traversal, which can help reduce missed visual evidence in difficult segments.

X. LIMITATIONS AND FUTURE SCOPE

Although the results are encouraging, the current evaluation has limitations inherited from the source study. First, experiments were conducted in controlled indoor conditions with relatively stable lighting. Actual industrial pipelines may include reflections, moisture, deposits, and varying illumination that can alter confidence behavior.

Second, the dataset and trajectory length were limited compared with full-scale field deployment. Extended testing over longer pipe segments, including bends and vibration-heavy motion, is required to evaluate long duration stability. Third, the present method focuses on crack confirmation reliability and does not yet include higher-level severity ranking or maintenance-priority classification.

Future work should include larger and more diverse pipeline datasets, dynamic lighting experiments, and adaptive temporal windows that change with robot speed and scene quality. Additional lightweight optimization methods, such as pruning or selective region inference, may further improve real-time embedded performance.

XI. CONCLUSION

This paper presents an efficiency and reliability enhancement framework for a Raspberry Pi based

pipeline inspection robot. The study shows that INT8 quantization substantially improves inference speed and reduces CPU usage, while five-frame confidence confirmation reduces unstable detections and false positives during continuous inspection.

Taken together, these methods provide a practical efficiency–reliability balance for low-cost embedded pipeline monitoring without requiring complex architectural redesign. The proposed approach is particularly relevant for low-cost inspection systems deployed in developing regions, where computational resources are limited and reliable automated monitoring is critical for infrastructure maintenance. Future work will focus on larger datasets, longer inspection paths, variable lighting conditions, and additional lightweight optimization techniques for field deployment.

REFERENCES

[1] J. Zhang, X. Niu, A. J. Croxford, and B. W. Drinkwater, "Strategies for pipeline inspection using mobile robots: Review of robot locomotion types, tradeoffs, and constraints," *Sensors*, vol. 21, no. 13, 2021.

[2] K. Murata and A. Kakogawa, "Development of a practical articulated wheeled in-pipe robot for force main inspection of sewer pipes (AIRo-7.1)," in *Proc. 29th Int. Symp. Artificial Life and Robotics (AROB)*, 2024.

[3] R. S. Elankavi, D. Dinakaran, A. S. A. Doss, R. K. Chetty, and M. Ramya, "Design and motion planning of a wheeled type pipeline inspection robot," *Int. J. Mechanical Engineering and Robotics Research*, vol. 10, no. 10, 2021.

[4] S. Elankavi R, D. Dinakaran, A. S. A. Doss, R. M. K. Chetty, and M. M. Ramya, "Design of a wheeled-type in-pipe inspection robot to overcome motion singularity in curved pipes," *J. Ambient Intelligence and Smart Environments*, vol. 16, no. 1, pp. 43–55, 2023.

[5] E. Islas-García, M. Ceccarelli, R. Tapia-Herrera, and C. R. Torres-SanMiguel, "Pipeline inspection tests using a biomimetic robot," *Biomimetics*, vol. 6, no. 1, p. 17, 2021.

[6] M. Kamezaki *et al.*, "SPIRA: Small gas pipeline inspection robot with spiral leg-wheel mechanism and single bending joint," *IEEE Robotics and Automation Letters*, vol. 10, no. 12, pp. 12676–12683, 2025.

[7] P. Chang, M. Wang, Z. Shao, A. Maimaiti, and W. Zhou, "Modular design of multi-unit pipeline inspection robot for low pressure gas pipeline," in *Proc. 10th Int. Conf. Automation, Control and Robotics Engineering (CACRE)*, 2025, pp. 370–375.

[8] R. A. Mangi *et al.*, "FOPID controlled self-propelled pipeline inspection robot for internal mapping of small-diameter pipelines," *IEEE Access*, vol. 13, 2025.

[9] N. M. Syahrian, P. Risma, and T. Dewi, "Vision-based pipe monitoring robot for crack detection using Canny edge detection method as an image processing technique," *KINETIK*, vol. 2, 2017.

[10] T. Wang, Y. Li, Y. Zhai, W. Wang, and R. Huang, "A sewer pipeline defect detection method based on improved YOLOv5," *Processes*, vol. 11, no. 8, 2023.

[11] P. Liu *et al.*, "Enhancing urban infrastructure: Water supply and drainage pipeline defect detection with the GLIP pre-trained model and YOLOv5 model," *Engineering Applications of Artificial Intelligence*, vol. 126, 2023.

[12] P. Rakhmetova, G. Sergazin, Y. Altay, D. Dauletiya, and L. Kurmangaliyeva, "Development of in-pipe defects detection and classification system," *Eastern-European Journal of Enterprise Technologies*, vol. 1, no. 9(133), pp. 80–89, 2025.

[13] Mukherjee, S. K. Pal, U. Paswan, N. R. Mandal, and T. Roy, "Expert-aware multi-stage AI-assisted image processing pipeline for accurate weld defect detection from radiographs," *Nondestructive Testing and Evaluation*, pp. 1–63, 2025.

[14] X. Xiao, M. Su, B. Guo, J. Wu, J. Wang, and J. Liang, "Design and experimental validation of pipeline defect detection in low-illumination environments based on bionic visual perception," *Biomimetics*, vol. 10, no. 9, p. 569, 2025.

[15] Daniyan, V. Balogun, B. Oladapo, O. Ererughurie, and L. Daniyan, "Development of an inline pipe inspection robot for the oil and gas industry," *Int. J. Automation and Smart Technology*, vol. 12, 2021.

[16] D. J. Seo, T. G. Kim, and S. W. Noh, "Underground pipeline tracking robot

development based on magnetic inductive sensor," in *Proc. 16th Int. Conf. Control, Automation and Systems (ICCAS)*, 2016, pp. 338–340.