Cureus Journal of Computer Science

Open Access Original Article

Compact Muon Solenoid Detector Quality Assurance Using Machine Learning

Ayush R. Gerra [1] , Shashikant Dugad [2] , Richard Joseph [1] , Anisha Rohra [1] , Sneha Tanna [1] , Preethika Shetty [1] , Gagan Mohanty [3] , Irfan Mirza [3] , Chetan Mandloi [3] , Kameswara Rao [3] , Pruthvi Suryadevara [3]

1. Computer Engineering, Vivekanand Education Society's Institute of Technology, Mumbai, IND 2. Department of High Energy Physics, Indian Institute of Science Education and Research, Mohali, IND 3. Department of High Energy Physics, Tata Institute of Fundamental Research, Mumbai, IND

Corresponding author: Ayush R. Gerra , 2021.ayush.gerra@ves.ac.in

Received 03/25/2025
Review began 04/17/2025
Review ended 09/16/2025
Published 10/03/2025

DOI:
https://doi.org/10.7759/s44389-025-03601-9

Abstract

Background

Precision and accuracy in quality assurance are critical for high-energy physics experiments. The Compact Muon Solenoid experiment at the Large Hadron Collider is undergoing an upgrade with the High-Granularity Calorimeter, requiring the inspection of numerous silicon detectors and hexaboards. Manual inspection methods, such as Coordinate Measuring Machines, are time-consuming and labor-intensive. This study investigates an automated quality assurance framework using flatbed scanners and deep learning algorithms.

Methods

The EPSON v850 flatbed scanner was selected for its high resolution and large scan area. Three approaches were explored for extracting stepped holes from hexaboard scans: Circular Hough Transform (CHT), printed circuit board edge detection, and bottom-most hole detection with CHT. Signal pad detection was performed using the YOLOv8 architecture, while concentricity offsets were measured with CHT. The pixel-to-real-world coordinate mapping was calibrated using dots per inch measurements, and the framework was optimized to reduce processing time.

Results

The automated inspection framework successfully localized 92 stepped holes per hexaboard and detected signal pads with high accuracy. The concentricity offset was calculated with a precision of 303 microns. The optimized system reduced processing time to approximately 2 m 6 s ± 692 ms at 3,200 dots per inch, compared to several minutes with manual methods. However, noise from dust and scanner skewness introduced occasional inaccuracies.

Conclusions

The proposed framework offers a scalable and efficient solution for quality assurance in the Compact Muon Solenoid experiment's High-Granularity Calorimeter upgrade. By leveraging deep learning and advanced image processing techniques, the system significantly enhances inspection speed and accuracy. Future improvements, including noise mitigation and enhanced scanner hardware, could further optimize performance. This methodology holds promise for broader industrial applications requiring high-precision inspections.

Categories: AI applications, Computer Vision, Machine Learning (ML)
Keywords: high granularity calorimeter, automated visual inspection, cms, object detection, hough transform, circular hough transform (cht), yolov8, flatbed scanner, pcb quality assurance

Introduction

In high-energy physics experiments, precision and accuracy are critical, particularly in the quality assurance (QA) of detector components. The Compact Muon Solenoid experiment at the Large Hadron Collider requires the production and inspection of numerous silicon detectors and hexaboards for its High-Granularity Calorimeter upgrade [1,2]. Traditional inspection methods, such as using a Coordinate Measuring Machine at the Tata Institute of Fundamental Research, are time-consuming and labor-intensive, as each hexaboard requires stepwise measurement of 92 stepped holes. This slow throughput creates a significant bottleneck in the production pipeline, potentially delaying detector assembly schedules. Moreover, manual inspections may introduce variability due to operator fatigue or subtle inconsistencies in measurement procedures. Coordinate Measuring Machines and Optical Gaging Products machines are much more expensive, further adding to the limitations of relying on these traditional methods.

How to cite this article
Gerra A R, Dugad S, Joseph R, et al. (October 03, 2025) Compact Muon Solenoid Detector Quality Assurance Using Machine Learning. Cureus J Comput Sci 2 : es44389-025-03601-9. DOI https://doi.org/10.7759/s44389-025-03601-9

The inefficiency of manual inspection has driven the shift towards automation, inspired by previous studies on automated visual inspection systems. Giri et al. [3] demonstrated the effectiveness of deep learning-based object detection methods, such as YOLOv4 tiny and VGG16, for inspecting wire bonds in printed circuit boards (PCBs). Similarly, Zhou [4] introduced a geometric characteristic-based vision detection method using the Hough Transform to identify regular hexagons with high accuracy, showcasing its potential for industrial applications. The Hough Transform, as discussed by Hassanein et al. [5], is a widely used technique in image processing for detecting geometric shapes and patterns. These studies highlight the promise of advanced computer vision techniques in improving inspection processes.

Building on these advancements, this study explores the use of flatbed scanners combined with deep learning algorithms to address the challenges of inspecting hexaboards at scale. Initial tests with the HP MFP M277SDN scanner revealed limitations in depth resolution, prompting the use of the EPSON Perfection v850 Pro scanner, which offers high resolution (up to 6,400 dots per inch [DPI]), a large scan area (8″ × 11″), and efficient scan times (8-9 minutes at 3,200 DPI). This setup enables the extraction of signal pads and concentricity measurements of stepped holes using a YOLOv8-based framework and Circular Hough Transform (CHT) [5-7].

The primary inspection criteria for hexaboards include:

1. Extraction of stepped holes from the PCB scan

2. Orientation of signal pads in the stepped holes

3. Area of signal pads in the stepped holes

4. Offset in the concentricity of stepped holes

By automating these processes, this study aims to develop a scalable, efficient, and reliable QA framework, paving the way for streamlined production of components critical to the Compact Muon Solenoid detector upgrade [1,2].

Materials And Methods

Scanner selection

The EPSON v850 flatbed scanner was selected for Hexaboard inspections due to its high resolution (up to 6,400 DPI), large scan area (8″ × 11″), and efficient scanning capabilities. Preliminary tests with other scanners confirmed its suitability. Inspections were conducted at Epson India, Andheri Branch.

Study design and approach

The Hexaboard's physical blueprint was analyzed to determine the real-world coordinates of each stepped hole. A calibration factor was derived from the scanner's DPI to map real-world dimensions to pixel dimensions.

1. Image Coordinates: The image origin is at the top-left corner, with x-coordinates increasing left-to-right and y-coordinates top-to-bottom.

2. Coordinate Transformation: Real-world coordinates were adjusted relative to a reference point (h, k) using:

$$(x', y') = (|x + h|, |k - y|)$$

3. Calibration Factor: Calculated as $\left(\dfrac{\text{DPI}}{25.4}\right)$ to convert real-world units (mm) to pixels, where DPI represents dots per inch and 25.4 is the number of millimeters in one inch.

4. Conversion Formula: Real-world coordinates (x, y) were converted to image coordinates (x', y') using the equations:

$$x\#\# = x * \text{Calibration Factor}$$

$$y\#\# = \text{Image Height} - (y * \text{Calibration Factor})$$

where x and y are real-world coordinates in millimeters, Calibration Factor converts millimeters to pixels, and Image Height is the height of the image in pixels.

Extraction of stepped holes

Three approaches were explored to extract 92 stepped holes from the Hexaboard scans:

Approach I: Stepped Hole Detection via CHT

1. Detect all stepped holes in the scanned image using CHT [5,7].

2. Map real-world coordinates to pixel coordinates.

3. Create key-value pairs linking coordinates to conventional names.

4. Crop and save detected stepped holes as .jpg files.

Initial testing observations: The CHT produced false positives, including detections along edges and on circuit components, as shown in Figure *1*. Additionally, the algorithm encountered computational challenges due to the high resolution of images, which exceeded 15,000 pixels in dimension [5,7]. The approach achieved a perfect recall of 100%, detecting all true holes, but showed a precision of ~88%. This indicates strong sensitivity for QA applications where missing defects is critical, but further post-processing or confidence threshold tuning became necessary.

FIGURE 1: Examples of False Positive Detections Recorded During Initial Testing of Approach I. A Total of 13 False Positives Were Observed, Primarily Along PCB Edges or Irregular Contours, Representing Approximately 12.38% of Total Detections

PCB, Printed Circuit Board

Approach II: Hexaboard Edge Detection for Stepped Hole Localization

This approach uses OpenCV to detect and mark the edges of the hexagonal PCB, with the following steps:

1. Detect and mark edges of the hexagonal PCB using Hough Transform [4].

2. Select a reference vertex for consistency.

3. Map real-world coordinates to pixel coordinates using the calibration factor.

4. Perform the steps from Approach I to extract and save the holes.

Challenges and observations: Inspired by Zhou [4], we applied Hough Transform to detect the PCB edges and map vertices for hole extraction. However, inaccuracies arose due to image rotation and noise.

Rounding of pixel coordinates to integers caused discrepancies, as seen in Figure 2, where purple lines deviated from the expected red line.

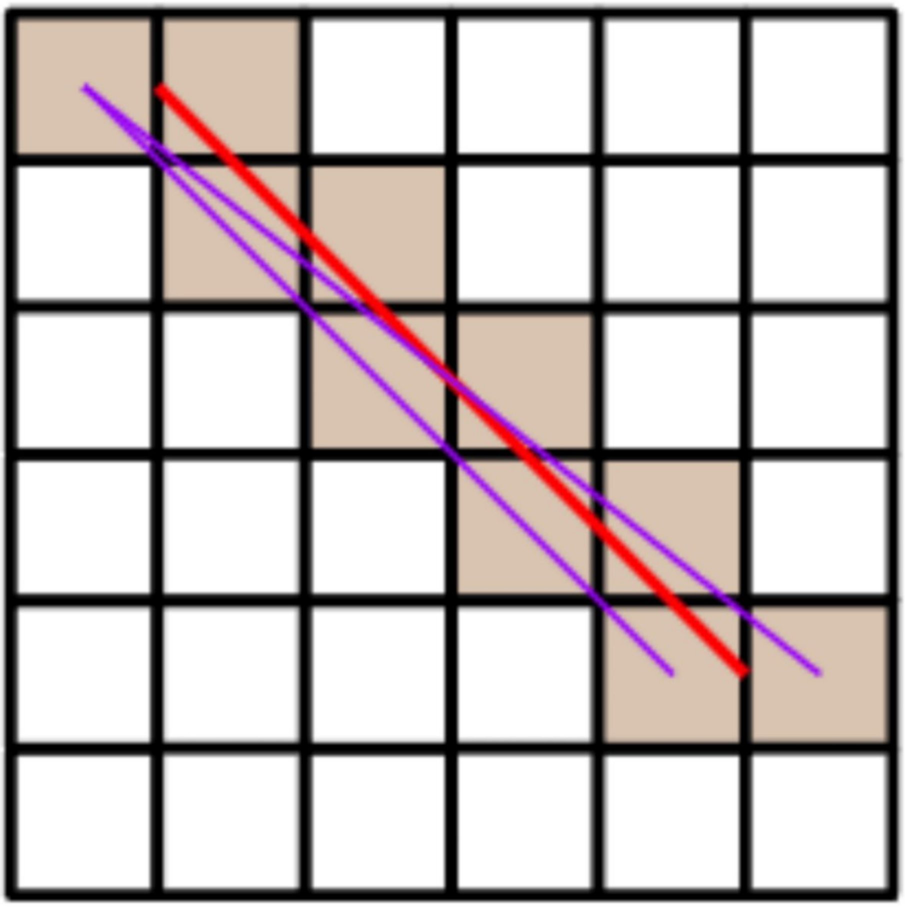

FIGURE 2: Comparison of Detected (Purple) and Required (Red) PCB Edge Lines, Showing Deviation

PCB, Printed Circuit Board

Increasing the offset for hole extraction helped mitigate some errors in detecting hole centers but created complications for subsequent YOLO model analysis [6]. Furthermore, as shown in Figure 3, the detected hexagonal boundary of the PCB is highlighted, while Figure 4 represents the corresponding schematic representation of the detected segments, which were used for alignment and feature extraction.

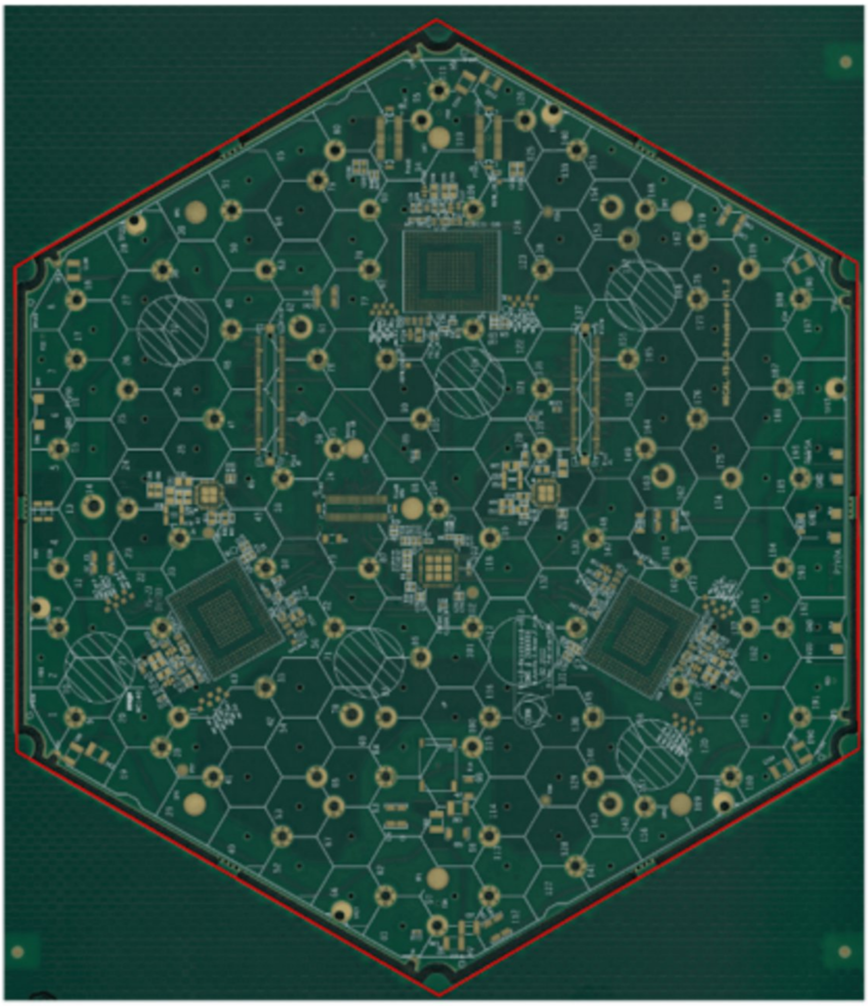

FIGURE 3: Detected Hexagonal Boundary on PCB

PCB, Printed Circuit Board

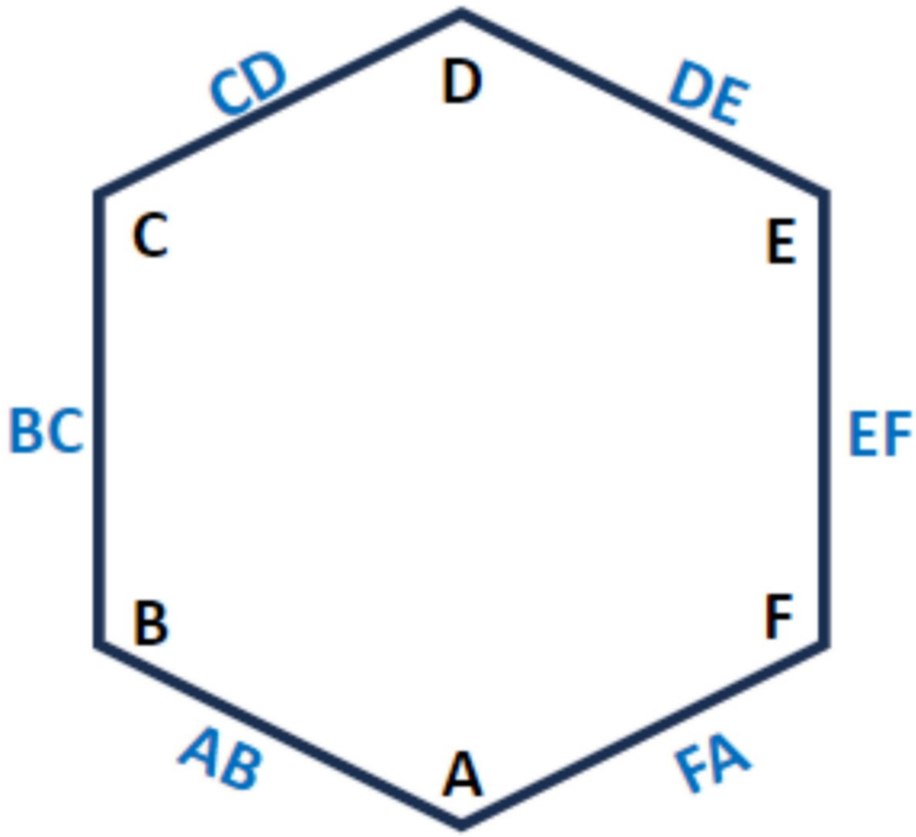

FIGURE 4: Corresponding Schematic Representation of Detected Segments

Inaccuracy example (Tables 1 and 2): Vertices A and D did not align vertically as expected, and diagonal intersections did not form a perfect circle through all vertices, which led to inconsistencies.

Line Type	Line	Length (in px)	Slope	Y-Intercept
Edge	AB	4,662	0.5776	7,361
Edge	BC	4,687	∞	294
Edge	CD	4,694	-0.5776	3,014
Edge	DE	4,645	0.5774	-2,021
Edge	EF	4,704	∞	8,382
Edge	FA	4,678	-0.5776	12,365
Diagonal	AD	9,367	-334.53	7,702
Diagonal	BE	9,360	-0.582	1,458,737
Diagonal	CF	9,343	0.578	2,674

TABLE 1: Hexaboard Edge Detection Results Using the Hough Transform

Vertices	Coordinates
A	(4331, 9863)
B	(294, 7531)
C	(294, 2844)
D	(4359, 496)
E	(8382, 2819)
F	(8382, 7523)

TABLE 2: Calculated Coordinates of Hexaboard Vertices

Approach III: Stepped Hole Detection via CHT in Limited FOV

This approach focuses on detecting the bottom-most stepped hole within a dedicated image window using the CHT. The bottom-most hole is used as a reference point to locate the positions of other holes. The procedure follows the steps in Approach II for further processing.

By concentrating on the bottom-most hole, this approach enhanced accuracy by addressing noise and pixel rounding issues, but challenges persisted when running the YOLO model [6]. The approach achieved hole extraction in 41.7 s ± 175 ms for images at 3,200 DPI. However, resizing the image to 600 DPI and then processing it to obtain pixel coordinates, followed by scaling back to 3,200 DPI, reduced the extraction time to 21 s ± 330 ms.

Steps for reproducing results:

1. Process a window containing the bottom-most hole using CHT.

2. Extract the bottom-most hole (S001), which serves as the reference point for locating other holes. For our example, the pixel coordinates obtained were (8,550, 18,523).

3. Translate to align the bottom-most hole with the origin of the real-world coordinate system. For our example, this involves subtracting the values (h, k) = (4.237, 9.189) from the x- and y-coordinates, respectively, as shown in Table *3*. The resulting translated coordinates are presented in Table *4*.

Point	Coordinate
S001	(4.237, 9.189)
S077	(-9.716, 154.199)
S086	(4.236, 178.379)

TABLE 3: Real-World Coordinate Values Extracted from Hexaboard Blueprint

Point	Coordinate
S001	(0, 0)
S077	(-13.953, 145.01)
S086	(0.001, 169.19)

TABLE 4: Translated Real-World Coordinates

a) x', the pixel x-coordinate of the stepped hole, is calculated by scaling the transformed real-world x-

coordinate, $x_{\text{TRANSFORMED}}$, to pixel units using $\frac{\text{DPI}}{25.4}$, then adding the reference point's pixel x-coordinate, x_{PIXEL}, and taking the absolute value:

$$x' = \left|\left(x_{\text{TRANSFORMED}} * \left(\frac{\text{DPI}}{25.4}\right)\right) + x_{\text{PIXEL}}\right|$$

$$x' = |(0 * 94.488) + 8550| = 8550$$

b) y', the pixel y-coordinate of the stepped hole, is calculated by scaling the transformed real-world y-coordinate, $y_{\text{TRANSFORMED}}$, to pixel units using $\frac{\text{DPI}}{25.4}$, subtracting the result from the reference point's pixel y-coordinate, y_{PIXEL}, and taking the absolute value:

$$y' = \left|y_{\text{PIXEL}} - \left(y_{\text{TRANSFORMED}} * \left(\frac{DPI}{25.4}\right)\right)\right|$$

$$y' = |18523 - (0 * 94.488)| = 18523$$

Coordinates of other holes are calculated using the same method, as illustrated in Table 5.

Stepped Hole	Real-world Coordinate	Transformed Coordinate	Pixel Coordinate
S001	(4.237, 9.189)	(0, 0)	(8550, 18523)
S077	(-9.716, 154.199)	(-13.953, 145.01)	(7231, 4821)
S086	(4.236, 178.379)	(0.001, 169.19)	(8550, 2536)

TABLE 5: Mapping Between Real-World and Pixel Coordinate Systems

Analysis of the stepped holes

Data Generation

The stepped holes in the hexagonal PCB contain two types of pads: Signal and Ground. Signal pads, which can be distinguished from ground pads by the presence of a notch, were detected using an instance segmentation approach with the YOLOv8 architecture [6,8]. The signal pads were annotated as a single class, "signalPad," using Roboflow [9].

To enhance model generalization, data augmentation techniques such as blurring with a maximum threshold of 0.7 pixels were applied, expanding the dataset from 375 to 824 images. The dataset was split into 720 training, 34 testing, and 70 validation images, each resized to 800 × 800 pixels. The model was trained using Ultralytics [10] for 100 epochs with a batch size of 16.

Orientation of Signal Pads

Following signal pad detection via YOLOv8, bounding boxes were generated around each identified pad [6,8,10]. As illustrated in Figure 5, each stepped hole contains one or more such bounding boxes. To compute the orientation of a signal pad, the center of the stepped hole - previously identified using the CHT - was geometrically compared to the centroid of the corresponding bounding box [5,7]. A vector was then drawn between these two points, and the gradient of this vector was used to define the pad’s orientation, measured in the clockwise direction relative to the center.

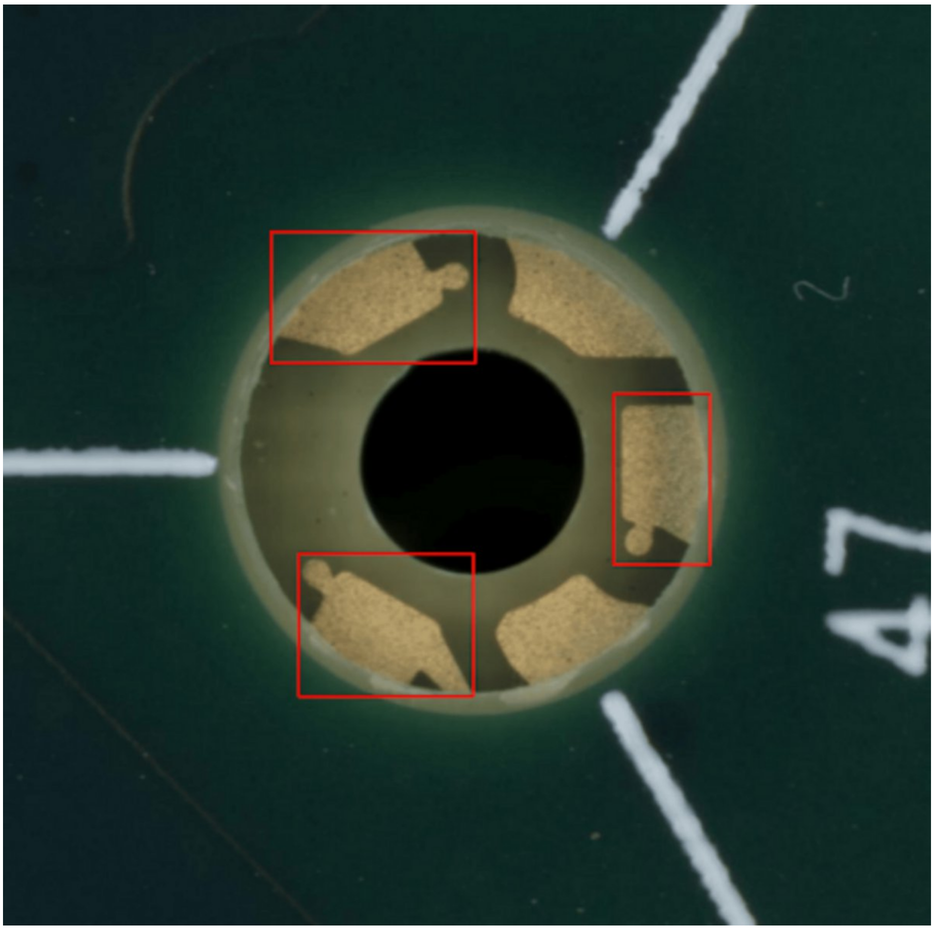

FIGURE 5: YOLOv8 Detection Results with Bounding Box

The steps to measure the vector gradient are as follows:

1. Determine the Approximate Center: The approximate center (x_c, y_c) of the stepped hole is obtained from the detection data, where x_c and y_c are the real-world x- and y-coordinates of the center.

2. Compute the Angle: The angle θ between the signal pad and the center of the stepped hole is calculated using the arctangent function:

$$\theta = \tan^{-1}\left(\frac{y_c - y_p}{x_c - x_p}\right)$$

where (x_p, y_p) represents the real-world coordinates of the signal pad, and (x_c, y_c) represents the real-world coordinates of the hole center. The numerator ($y_c - y_p$) is the vertical difference, and the denominator ($x_c - x_p$) is the horizontal difference between the center and the signal pad.

3. Convert to a Positive Angle: If the computed angle is negative, it is adjusted by adding 360° to ensure a positive value:

$$\theta = 360 + \theta \text{ if } \theta < 0$$

This ensures that the angle lies within the range [0°, 360°].

4. Adjust for Reference Alignment: To align with the system's counterclockwise reference, the angle is adjusted by adding 270° and taking the result modulo 360°:

$$\theta' = (\theta + 270) \bmod 360$$

where θ' is the adjusted angle used for orientation calculation.

5. Convert to Clock-Hour Representation: The final orientation is expressed as a clock-hour value using the formula:

$$H = 12 - \left\lfloor \frac{\theta'}{30} \right\rfloor$$

where H is the hour on a 12-hour clock, θ' is the adjusted angle in degrees, and 30° represents the angle between each hour position (360°/12 = 30°).

This approach enables the determination of a signal pad’s orientation relative to the center of the stepped hole (Figure 6).

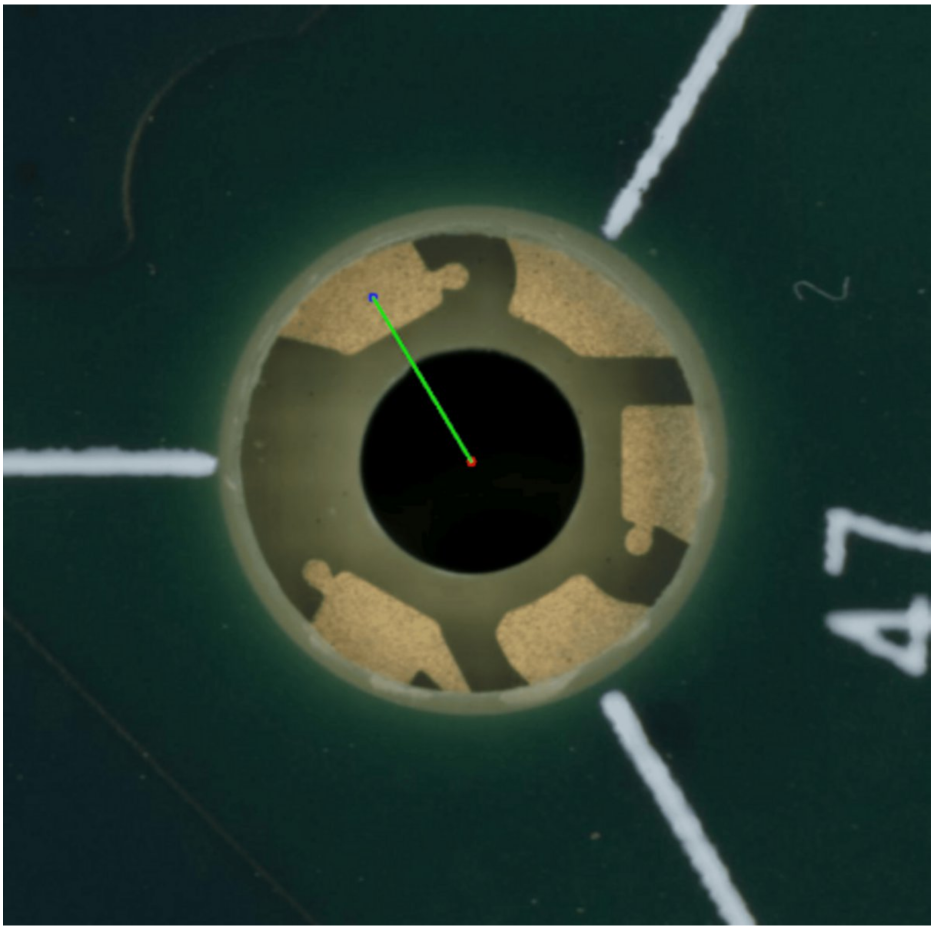

FIGURE 6: Orientation Estimation Using the Bounding Box and Stepped Hole Center

Area of Signal Pads

To compute the area of each detected signal pad, the segmentation mask was processed using OpenCV functions [11]. A mask was created from the bounding box to isolate the signal pad region. This mask was then multiplied with the original image to extract the exact region of interest. Using cv2.findContours(), the contours of each signal pad were identified (Figure 7) [11]. The function cv2.contourArea() was subsequently used to calculate the area of each pad, and the results were stored in a CSV file for further analysis [11].

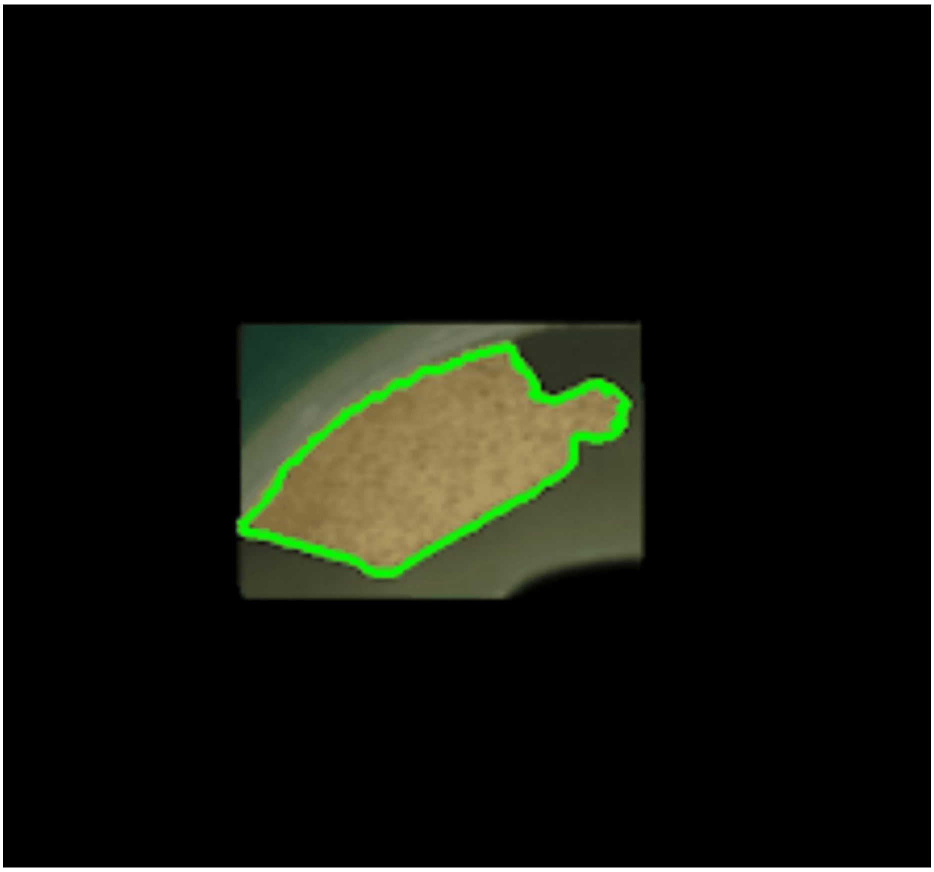

FIGURE 7: Contour Detection Results for Extracted Signal Pad Region

Offset in Concentricity

Offset detection involved measuring the centers of two concentric circles in the stepped hole images. The CHT was used to detect these circles, yielding their radii R_1 and R_2, and centers $C1 = (C1_x, C1_y)$ and $C2 = (C2_x, C2_y)$ for the outer and inner circles, respectively [5,7]. A significant offset between the centers indicated a potential design flaw in the PCB holes.

The preprocessing pipeline included Gaussian Blur and Canny Edge Detection before applying the CHT [12]. The Euclidean distance between the centers was then calculated as the offset:

$$d = \sqrt{(C1_x - C2_x)^2 + (C1_y - C2_y)^2}$$

where $C1_x$ and $C1_y$ are the x- and y-coordinates of the outer circle's center, and $C2_x$ and $C2_y$ are the x- and y-coordinates of the inner circle's center. The resulting distance d represents the magnitude of the offset.

The computed offsets were stored in a CSV file for further inspection and quality control of the PCB holes (Figure 8).

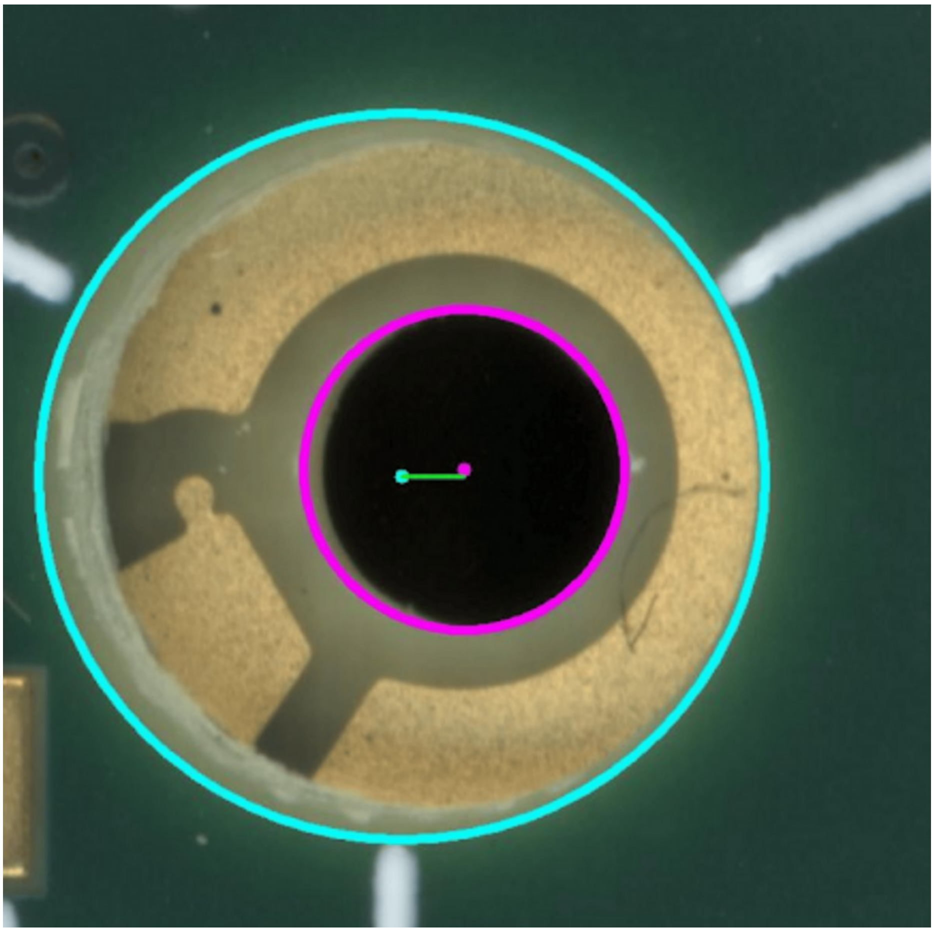

FIGURE 8: Concentric Circle Offset Measurement via Circular Hough Transform

Results

The processed images showcased precise localization and segmentation of features. Figure *9* represents a raw extracted stepped hole before processing, highlighting the initial detection. The application of YOLOv8 and the CHT further enhanced the accuracy of detecting the stepped holes, as illustrated in Figure *10* [6-8].

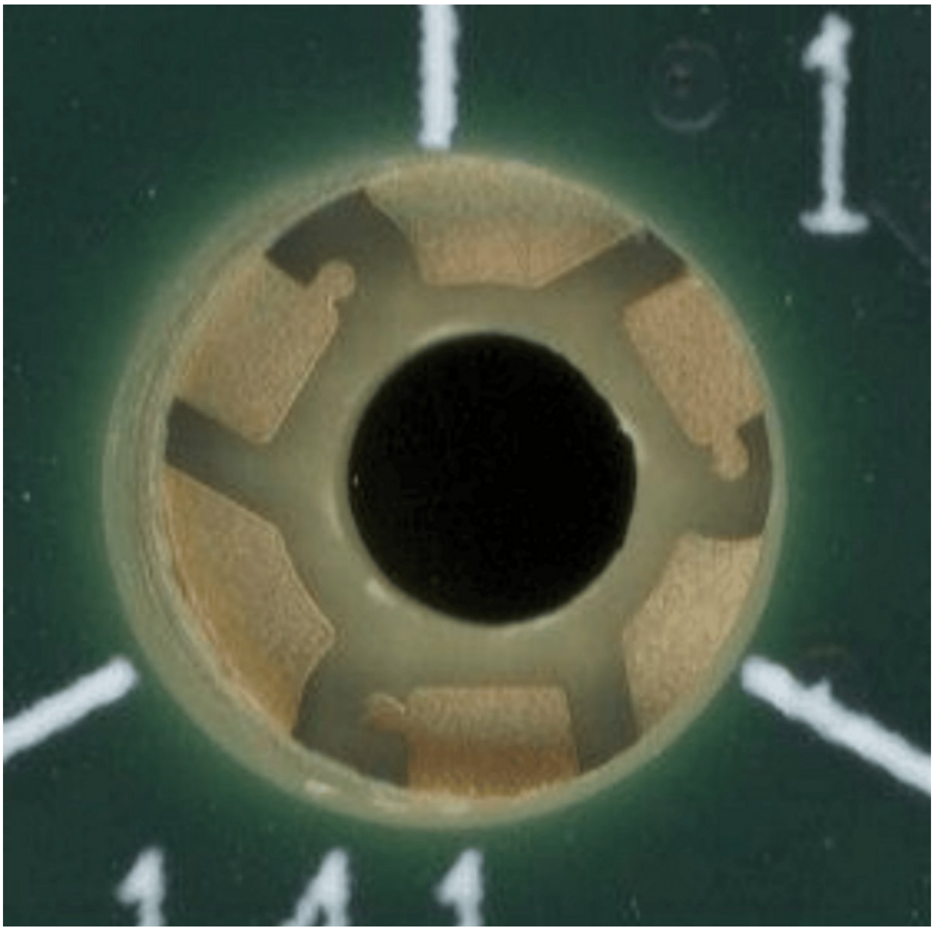

FIGURE 9: Raw Extracted Stepped Hole

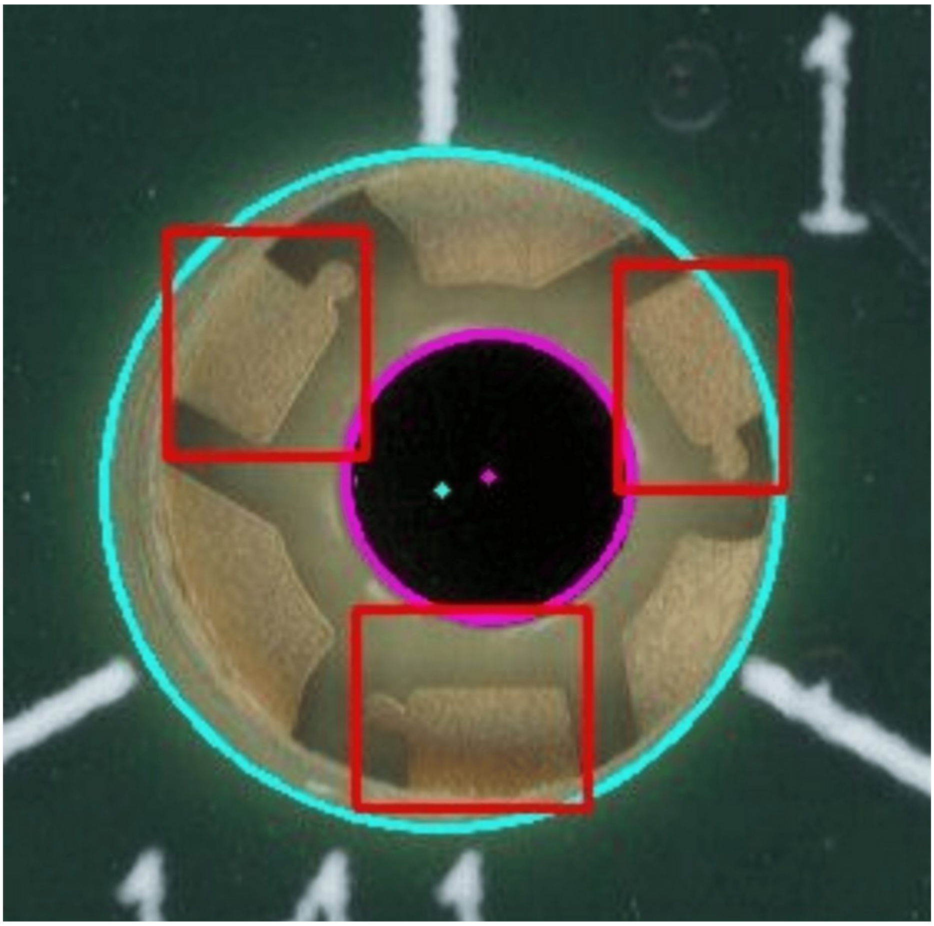

FIGURE 10: Processed output of YOLOv8 and Hough Transform

Feature descriptions

• **Local Centers:** The columns local_outer_centre_x, local_outer_centre_y, local_inner_centre_x, and local_inner_centre_y represent the centers of the stepped hole relative to the extracted image.

• **Global Centers:** The columns global_outer_centre_x, global_outer_centre_y, global_inner_centre_x, and global_inner_centre_y indicate the centers of the stepped hole relative to the original scanned image.

• **Offset:** The positional offset, calculated in pixels (offset), is converted to microns (offset_microns).

Signal pad detection

Clockwise-oriented signal pad areas are labeled as Area0 through Area12, with Area1 representing the pad area at the 1 o'clock position, and so on. If no signal pad is detected, the area value remains 0. For example, Tables *6* and 7 present the positional features, areas, and orientations of signal pads for the stepped hole S006_114_113_098.

Parameter	Value
image	S006_114_113_098
actual_x (in px)	1,579
actual_y (in px)	4,715
local_outer_centre_x (in px)	133
local_outer_centre_y (in px)	145
local_inner_centre_x (in px)	147
local_inner_centre_y (in px)	142
global_outer_centre_x (in px)	1,572
global_outer_centre_y (in px)	4,720
global_inner_centre_x (in px)	1,586
global_inner_centre_y (in px)	4,717
outer_centre_x (in mm)	-68
outer_centre_y (in mm)	80
inner_centre_x (in mm)	-68
inner_centre_y (in mm)	80
offset (in px)	14.31782106
offset_microns	303.0605458

TABLE 6: Positional for Image S006_114_113_098

Parameter	Value
image	S006_114_113_098
Area0	0
Area1	0
Area2	0
Area3	177,014.1875
Area4	0
Area5	0
Area6	128,500.1875
Area7	0
Area8	0
Area9	0
Area10	0
Area11	0
Area12	0

TABLE 7: Area Measurements for Image S006_114_113_098

Performance observations

The framework demonstrated robust performance under ideal conditions but was susceptible to noise caused by dust on the PCB surface or scan bed. This noise occasionally affected detection accuracy. However, it was effectively mitigated by carefully cleaning the scan bed with a microfibre cloth prior to each scan. Nevertheless, implementing preprocessing techniques such as background normalization and de-skewing algorithms remains a potential future improvement to further enhance robustness.

Timings

To evaluate the computational efficiency of the proposed automated inspection framework, the image processing pipeline was tested on a system with the following specifications. The primary goal of this evaluation is to measure the time taken to extract stepped holes from the hexaboard scans and perform subsequent geometric analysis at different scan resolutions. The results show that higher scan resolutions, such as 3,200 DPI, significantly increase processing time (from 1 second at 600 DPI to 21 seconds at 3,200 DPI for hole extraction alone) but also improve detection accuracy for small and detailed features like concentric stepped holes. This indicates that while lower resolutions enable faster inspection throughput, higher resolutions are necessary when precise geometric measurements are critical for QA workflows. Therefore, selecting an optimal resolution requires balancing processing speed with the accuracy demands of specific inspection tasks. The computational times for each resolution are summarized in Table *8*.

Specifications of the test machine:

• Processor - i5 10400F (6 Cores 12 Threads)

• RAM - 16 GB 2,666 MHz

• 7,200 rpm hard disk drive

	Time Taken to Obtain Holes	Time Taken to Obtain Holes + Analysis
600 DPI	1 s ± 200 ms	1 m 41 s ± 262 ms
1,200 DPI	2.6 s ± 400 ms	1 m 48 s ± 262 ms
3,200 DPI	21 s ± 330 ms	2 m 6 s ± 692 ms

TABLE 8: Computational Time for Hexaboard Image Processing

YOLOv8 training performance

To evaluate the detection capabilities of the proposed framework, a YOLOv8 model was trained on a dataset of 824 annotated hexaboard images. The training was conducted over 100 epochs, and the loss convergence and evaluation metrics are summarized in Figure *11*.

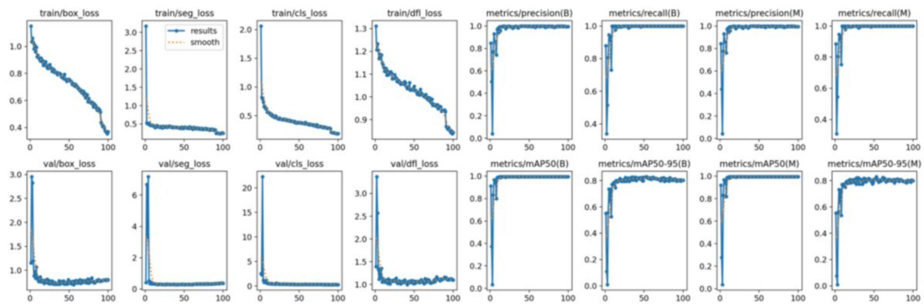

FIGURE 11: Training and Validation Loss Curves Along With Precision, Recall, and mAP Metrics for YOLOv8 Trained on 824 Images

mAP, Mean Average precision

As shown in Figure *11*, the model achieved robust convergence across the box loss, segmentation loss, classification loss, and distribution focal loss components. Evaluation on the validation set resulted in Precision: 0.98, Recall: 0.99, and F1 Score: 0.985.

The F1 score was computed as:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} = 2 \times \frac{0.98 \times 0.99}{0.98 + 0.99} \approx 0.985$$

Additionally, the model attained a mean average precision of ~0.98 at IoU = 0.5 and ~0.82 across IoU = 0.5-0.95, demonstrating high segmentation and detection performance suitable for deployment in automated inspection workflows.

These results indicate that the YOLOv8-based framework is capable of accurately detecting stepped holes and relevant features on hexaboard scans with minimal false positives or missed detections, satisfying the precision requirements for QA applications.

Discussion

The results demonstrate certain limitations inherent to the scanner's design and functionality. The skewness observed in the stepped holes, likely due to the scanner's reliance on reflected light beams captured by a sensor, introduced inaccuracies in offset validation. Addressing this issue in future work could involve exploring alternative scanner technologies or developing strategies to mitigate the skewness effect.

Additionally, the scanner's inability to adjust focal length hindered its ability to capture fine details, such as the layer containing wire bonds. Acquiring a scanner with adjustable focal length or enhanced resolution capabilities could significantly improve the project's outcomes.

Conclusions

This study presents an efficient and scalable QA framework for inspecting hexaboards leveraging advanced imaging technologies - such as the EPSON v850 scanner - and state-of-the-art machine learning techniques like YOLOv8 and the CHT. Key findings include a substantial reduction in inspection time, from several minutes per hexaboard with traditional methods to approximately 2 minutes and 6 seconds (±692 ms) using optimized image processing techniques. The automated framework effectively detects flaws in stepped holes and analyzes signal pads for accurate concentricity and orientation.

While the framework demonstrates high effectiveness, challenges remain - primarily image noise, scanner-induced skew, and limited visibility of fine wire bond layers. Future improvements in imaging hardware and algorithmic robustness are expected to address these limitations. Overall, this framework offers a promising solution for high-precision inspection in large-scale manufacturing, showcasing the value of integrating deep learning and computer vision in industrial quality assurance workflows.

Additional Information

Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of the work.

Concept and design: Ayush R. Gerra , Richard Joseph, Anisha Rohra, Sneha Tanna, Preethika Shetty, Gagan Mohanty, Irfan Mirza, Kameswara Rao, Shashikant Dugad, Chetan Mandloi, Pruthvi Suryadevara

Acquisition, analysis, or interpretation of data: Ayush R. Gerra , Richard Joseph, Anisha Rohra, Sneha Tanna, Preethika Shetty, Gagan Mohanty, Irfan Mirza, Kameswara Rao, Shashikant Dugad, Chetan Mandloi, Pruthvi Suryadevara

Drafting of the manuscript: Ayush R. Gerra , Richard Joseph, Anisha Rohra, Sneha Tanna, Preethika Shetty, Gagan Mohanty, Irfan Mirza, Kameswara Rao, Shashikant Dugad, Chetan Mandloi, Pruthvi Suryadevara

Critical review of the manuscript for important intellectual content: Ayush R. Gerra , Richard Joseph, Anisha Rohra, Sneha Tanna, Preethika Shetty, Gagan Mohanty, Irfan Mirza, Kameswara Rao, Shashikant Dugad, Chetan Mandloi, Pruthvi Suryadevara

Supervision: Richard Joseph, Gagan Mohanty, Irfan Mirza, Kameswara Rao, Shashikant Dugad, Chetan Mandloi, Pruthvi Suryadevara

Disclosures

Human subjects: All authors have confirmed that this study did not involve human participants or tissue. **Animal subjects:** All authors have confirmed that this study did not involve animal subjects or tissue. **Conflicts of interest:** In compliance with the ICMJE uniform disclosure form, all authors declare the following: **Payment/services info:** Authors acknowledge the financial support from the Department of Atomic Energy, Government of India, under an extramurral research grant. **Financial relationships:** All authors have declared that they have no financial relationships at present or within the previous three years with any organizations that might have an interest in the submitted work. **Other relationships:** All authors have declared that there are no other relationships or activities that could appear to have influenced the submitted work.

References

1. Gerwig H: Engineering challenges in mechanics and electronics in the world's first particle-flow calorimeter at a hadron collider: The CMS high-granularity calorimeter. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. 2022, 1044:167493. 10.1016/j.nima.2022.167493
2. Quast T: Status and plans for the CMS high granularity calorimeter upgrade project. Journal of Physics: Conference Series. 2022, 2374:012020. 10.1088/1742-6596/2374/1/012020
3. Giri N, Dugad S, Chhabria A, Manwani R, Asrani P: Automated visual inspection of silicon detectors in CMS experiment. arXiv. 2022, 10.48550/arXiv.2206.02572
4. Zhou ZF: An approach for detecting regular hexagon. Advanced Materials Research. 2011, 187:780-785. 10.4028/www.scientific.net/AMR.187.780
5. Hassanein AS, Mohammad S, Sameer M, Ragab ME: A survey on Hough transform, theory, techniques and applications. arXiv. 2015, 10.48550/arXiv.1502.02160
6. Yaseen M: What is YOLOv8: An in-depth exploration of the internal features of the next-generation object detector. arXiv. 2024, 10.48550/arXiv.2408.15857
7. Rizon M, Yazid H, Saad P, et al.: Object detection using circular Hough transform. American Journal of Applied Sciences. 2005, 2:1606-1609. 10.3844/ajassp.2005.1606.1609
8. Kukartsev VV, Ageev RA, Borodulin AS, Gantimurov AP, Kleshko II: Deep learning for object detection in images: Development and evaluation of the YOLOv8 model using Ultralytics and Roboflow libraries. Software Engineering Methods Design and Application. Silhavy R, Silhavy P (ed): Springer, Cham; 2024. 1118:629-637. 10.1007/978-3-031-70285-3_48
9. Roboflow. Accessed: October 20, 2023: https://blog.roboflow.com.
10. Ultralytics. Accessed: October 20, 2023: https://docs.ultralytics.com.
11. OpenCV: Contour Features. Accessed: October 22, 2023: https://docs.opencv.org/4.x/dd/d49/tutorial_py_contour_features.html.
12. OpenCV: Canny Edge Detection. Accessed: October 22, 2023: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html.