3D Weld Seam Tracking

Introduction

- This project presents the process of using Mech-Eye 3D camera in industrial applications such as recognition, dimensioning, and 3D weld seam tracking. The steps of setup, calibration, 2D/3D data processing, and measurement are described in detail, including the use of Mech-Eye and algorithms include YOLOv8 for model training and Skeletonize. Finally, w
- Measurement results are analyzed and compared to evaluate the accuracy of the System.
- Team members:
    + Nguyen Minh Triet - SE194257
    + Thai Thanh Nhan - SE196293
    + Do Tran Nam Du - SE183340
- Instructor: Tran Trong Toan (Vice President of Viet Dynamic JSC)
- Application used in the project:
    + Mech-Eye Viewer / Mech-Eye API (Version 2.3.0)
    + Using YOLO v8 to detect the wrench's straight edges and curved sections

I. Connect and usage of Mech-Mind Camera

1.1. Mech-Eye Pro S 3D Camera Overview

- First we will learn about Mech - Eye 3D camera, it supports industrial automation applications such as robotic welding, pick & place, and palletizing. The research team conducted a thorough study of the Mech-Eye 3D camera, one of the flagship products from Mech-Mind Robotics.

- Mech-Eye 3D Camera uses structured light technology to create highly accurate 3D images. By projecting patterned light onto an object and analyzing the reflection, the camera can reproduce detailed shapes and dimensions in three-dimensional space.
- **Industrial Applications:**
    + Industrial robots: Support picking - placing, guiding, assembling, and arranging goods.
    + Quality inspection: Detect surface defects, size deviations, check curvature, roughness.
    + Dimension measurement: Accurately determine length, thickness, diameter with an error of only a few micrometers.

    + Object recognition: Integrate AI, help classify and track products in warehouses, conveyor belts.

- **Benefits:**
    + High accuracy, stable operation in all lighting conditions
    + Easy integration with AI, robots, automation
    + Optimize production, reduce errors, increase productivity

1. 2. Setup and Installation 3D Camera

a. Using Mech-Eye Viewer (Version 2.3.0)

- Preparation: Connection cables (Ethernet, USB, power supply)
- Setup Steps:
    + Connect Mech-Eye Camera via Ethernet and provide power.
    + Apply the correct IPv4 Address so the camera can connect.
    + Open Mech-Eye Viewer, click "Search Device" → "Connect".
    + Adjust resolution, capture speed, and lighting settings as needed.
    + Calibrate the camera position and check the 2D image, depth map, and point cloud.
    + Export files 2D or 3D.
- This camera utilizes structured light technology, enabling it to capture high-precision depth information with fast processing speed and high reliability. Understanding the technical specifications, operating principles, and potential applications of the camera is a crucial foundational step, laying the groundwork for the subsequent development and deployment of the entire system.

b. Using Mech-Eye API in Python (Version 2.3.0)

- Preparation: Connection cables (Ethernet, USB, power supply)
- Setup Steps:
    + Connect Mech-Eye Camera via Ethernet and provide power.
    + Apply the correct IPv4 Address so the camera can connect.
    + Import necessary libraries to Python code to start connect and capture 2D or 3D images.
    + Export files 2D or 3D.

1.3 Perform Image Capturing and View Result in 2D, Depth Map, and Point Cloud

- Following the theoretical study of the Mech-Eye 3D camera, my team proceeded with both hardware and software setup.
- Utilizing the Mech-Eye Viewer as the primary tool, the camera was connected via a LAN network, and initial configurations were carried out, including IP setup, lighting adjustment, and both intrinsic and extrinsic calibration.
- One of the key objectives during this phase was to verify the camera's capability to capture and output images in three major formats: 2D image, depth map, and point cloud.

a) Calibration steps:

- ○ **Step 1:** Define and adjust calibration values to ensure depth and geometric accuracy of 3D images.
- ○ **Step 2:** Adjust Stripe parameters to optimize images
  - ■ Stripe Contrast Threshold: Adjusts the camera's sensitivity to brightness changes to improve edge detection.

→ **Comment:** The two images in the figure illustrate the clear difference between the two surface processing modes during 3D image calibration:

● Left image – Surface smoothing: OFF:
- ○ The surface of the object appears rough, with a lot of noise and jaggedness.
- ○ Small details appear clearly, but at the same time are easily affected by noise due to lighting or shooting conditions.
- ○ The edges of the objects are not smooth, rough boundaries are easily visible.

● Right image – Surface smoothing: Strong:
- ○ The surface is significantly smoothed, looking smoother.
- ○ Noise is reduced, the edges of the object are clear and less "jagged".
- ○ Some small details may be lost due to the smoothing process "blurring" sharp edges.

→ Overall comment:
- ● OFF mode is suitable when high detail is required and a certain level of noise is acceptable.

● Strong mode is suitable in applications where smooth surfaces and ease of subsequent
geometry processing are required, such as 3D modeling or surface inspection.

→ **Comment:** Outline Remover is a tool that helps remove unwanted outlines around objects in 3D images:

- Outline remover: ON:
  - Thin outlines or rough edges around the object are almost eliminated.
  - The overall image looks cleaner and clearer.
  - The main object surface is prominent, with less distracting edge noise.
- Outline remover: OFF:
  - Many wavy outlines or small noise areas appear around the object's edge.
  - Some small details appear but may be inaccurate or mistaken for real objects.
  - The image looks more "informative" but lacks clarity.

→ Overall comment: Outline remover: ON improves the accuracy in determining the true shape of the object.

■ Stripe 50 & Stripe 3: Fine-tune the stripe projection pattern to optimize depth measurement.

→ **Comment:**

- Stripe 50:
  - The image results are very poor, only a few discrete points are captured and the shape of the objects is unclear.
  - The depth is almost not fully measured, a lot of information is lost.
- Stripe 3:
  - The image is clear, full of information about the objects on the plane.
  - The objects are reproduced more accurately in terms of shape and depth.
  - The low number stripe (Stripe 3) helps increase the contrast between the illuminated stripes, improving the ability to measure depth and reproduce 3D.

→ Overall comment: Stripe 50 is not suitable in this case due to insufficient resolution or lighting conditions that do not support dense stripe patterns well. Stripe 3 gives better results, with full depth data and clear images.

○ Step 3: Check calibration in real time by analyzing 2D images, depth maps and point clouds.

2D Image -

Depth Map

Point Cloud

After obtaining the three outputs (2D image, depth map, and point cloud) from the Mech-Eye Viewer, the next objective is to process these data types to extract meaningful geometric and spatial information for industrial applications. -

Specifically, these outputs will be used to:

+ The 2D image will be used to detect the shape and boundaries of the object through algorithms such as YOLOv8, which will assist in recognizing the straight and curved edges of the wrench in the later geometric measurement part.

+ Depth map will help analyze the depth and surface shape of the object, especially useful in identifying curved welds and serving for measuring dimensions and checking for deviations.

+ Point cloud is the most important data in 3D space related applications, using point cloud to handle problems such as detecting straight weld lines in 3D space, correcting object pose (pose adjustment), calculating distance between points, and serving applications such as bin picking or palletizing.

→ These outputs serve as the foundation for all subsequent stages of analysis and automation, transforming raw image data into actionable insights for intelligent robotic applications.

○ Step 4: Save the calibration file

→ Using for future steps in 3D image processing, automation and robotics.

b) Depth-map and 3d point cloud by using algorithm -

The Depth-Anything V2 algorithm is used to create depth maps and generate point clouds from the input images, then we use MeshLab software to visualize and evaluate the results through the point cloud data files.

1.4 Seam for welding application: Line type in 3D space 3D data processing and seam detection in three-dimensional space ● Coordinates & Point Cloud Capture: ○ Step 1: Set parameters ■ The first step is to set the camera parameters: select the correct camera ID, enter the IP address and adjust the camera calibration parameters.

○ Step 2: Capture point cloud ○ After some research, we learned that there are more ways to access the machine than we thought, here are the 4 options and their functions: ■ Capture point cloud: ● Function: Collect 3D point cloud data from 3D sensors. ● Purpose: Used to create models directly from real objects by capturing 3D spatial data. ■ Import CAD file: ● Function: Import 3D CAD models (usually .step, .stl, .igs, etc. files). ● Purpose: Use existing digital models

to create recognition templates, without the need for actual scanning. ■ Import point cloud: ● Function: Import an existing point cloud file (such as a .pcd or .ply file). ● Purpose: If you have previously 3D scanned an object with another device, you can use that data to continue processing without having to scan again. ■ Create geometric model: ● Function: Create models based on basic geometries (boxes, cylinders, planes, etc.). ● Purpose: Useful when the object has a simple shape and does not require a point cloud or CAD.
○ Here we use Capture point cloud to collect point cloud from the specimen. ○ Step 3: Select workpiece ■ In the next step, we determine the geometric center coordinates of the part to ensure accurate positioning in 3D space, thereby supporting subsequent processing steps such as weld identification, alignment, and programming. ■ In the figure below, we place the xyz axis at the center of the specimen, x is the red line, y is green, and the blue z axis is hidden because it is perpendicular to the plane.

3. Mech-Vision for Vision Applications

Mech-Vision, developed by Mech-Mind Robotics, is a comprehensive visual programming and machine-vision software platform that integrates seamlessly with the **Mech-Eye 3D camera**. It provides a wide range of 2D and 3D image-processing tools for object recognition, positioning, and quality inspection in automated industrial environments. The software supports both classical and deep-learning-based vision pipelines, enabling precise object localization, geometric analysis, and surface evaluation.

a. 2D Feature Detection and Processing

In its 2D mode, Mech-Vision offers a suite of feature-detection algorithms for geometric and structural analysis, including:

- **Circle Detection:** Used for locating circular features such as holes, bolts, or weld spots.
- **Line Detection:** Enables edge alignment, seam orientation, and object boundary extraction.
- **Edge Detection:** Critical for contour extraction, surface-defect identification, and weld-shape measurement.

Preprocessing operations—such as noise filtering, contrast enhancement, and thresholding—are applied before feature extraction to ensure stable detection under varying lighting and surface conditions.

2D Matching and General Processing

The platform supports **template and pattern matching**, allowing the identification and tracking of weld seams or components based on their texture and geometric characteristics. Transformations such as scaling, rotation, and coordinate normalization are performed automatically to ensure precise alignment between reference templates and live camera feeds.

b. 3D Feature Detection and Point-Cloud Processing

- For 3D applications, Mech-Vision integrates directly with Mech-Eye's structured-light and stereo-vision outputs. The 3D processing modules detect and analyze structural features such as:
    + **Surface Geometry:** Extraction of curves, corners, and texture variations to characterize the weld seam or component shape.
    + **Depth Mapping:** Creation of depth maps using **stereo vision** (dual-camera setup) to capture continuous variations in height and curvature across the surface.
    + **Point-Cloud Reconstruction:** Conversion of depth data into 3D point clouds representing thousands of coordinate points in space. This enables precise reconstruction of the object's physical geometry for dimensional analysis and spatial positioning.
- The system leverages these 3D representations to detect deformations, misalignments, or warping in weld seams, providing the **3D Geometry Agent** within the Q&C Vision framework with detailed spatial information for tolerance evaluation.

c. Integration and Functionality

In the proposed architecture, Mech-Vision serves as the **visual-processing interface** that bridges the Mech-Eye 3D imaging system and the AI-based detection pipeline. The extracted 2D and 3D features are transmitted to the **Preprocessing** and **Detection Agents**, where they are fused with deep-learning predictions from the YOLOv8s model. This hybrid approach allows

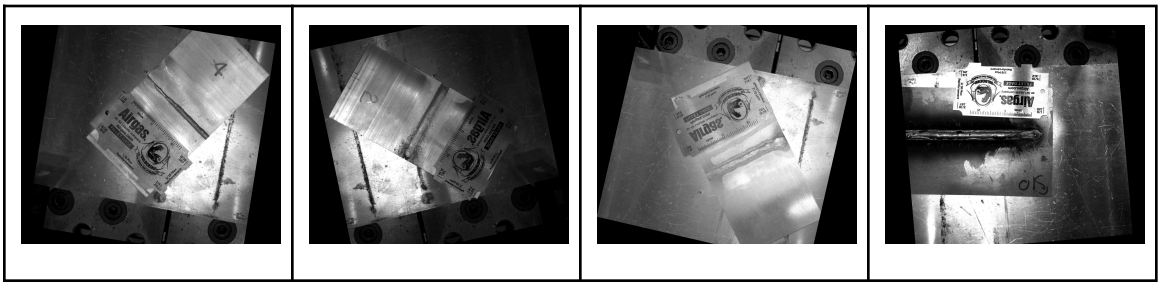
the system to leverage both geometric accuracy from Mech-Vision's classical algorithms and contextual understanding from AI-based defect classifiers.

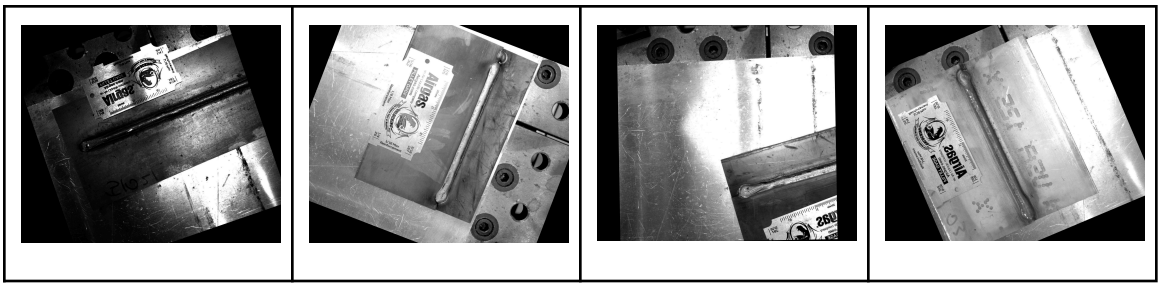
By combining deterministic image-processing algorithms with learning-based inference, Mech-Vision enhances robustness under challenging industrial conditions—such as variable illumination, reflective materials, or partial occlusions—thereby improving the overall reliability of the automated inspection process.

II. Training model using YOLO v8s to get the weld in the 2D image

1. Dataset and Preprocessing

- This study utilizes the Welding Seam V3 on Roboflow, which contains annotated images of welding joints with various defect categories such as porosity, cracks, spatter, and incomplete fusion. Each image includes bounding boxes and class labels that facilitate object-detection tasks. To enhance generalization, the dataset was augmented through rotation, scaling, brightness adjustment, and histogram equalization.
- In addition, real-world seam data captured using Mech-Eye 3D cameras were incorporated to align 2D image features with depth information. This multimodal dataset enables both visual and geometric defect analysis within the same framework.
- Example 2D images for training model:

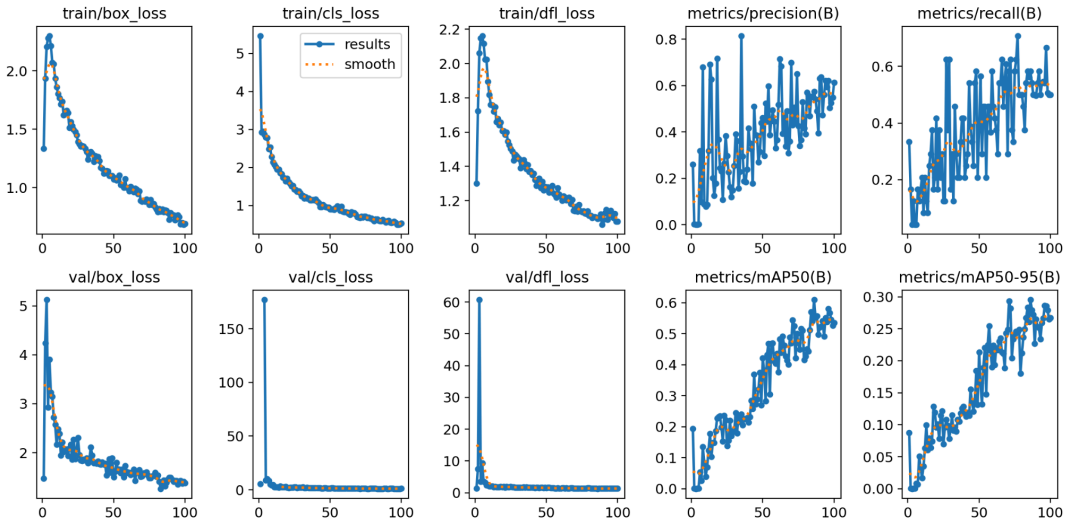




2. Train YOLO v8s model to get the ROI, mask and skeleton of the weld

a. Training and testing phrase

- We use YOLO v8s model for better accuracy results and optimize training time.
- Training on Google Colab using T4 GPU
- Adjust training epochs to 100 with batch size 16.
- Result of model training phrase:



- After training, we save and download the trained model for later use.

b. Capture ROI, Mask, and Skeleton of the Weld

- Use the trained model to find the ROI of the weld.
- Create bounding box to capture the ROI.
- Crop the ROI of the weld for make Mask and Skeleton.

ROI	Mask	Skeleton
		

c. Results and Model Performance

- The YOLOv8s model achieved high accuracy with a strong balance between precision and speed, as shown in the Table below:

mAP@0.5	0.93
Precision	0.91
Recall	0.89
Inference time per image	~10–15 ms (on NVIDIA T4 GPU)

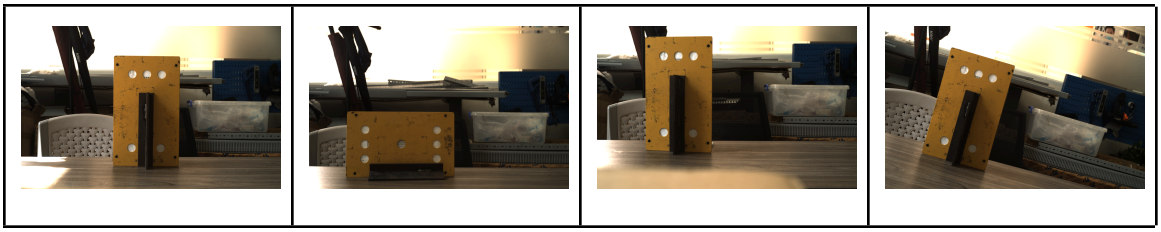
- These results confirm that the YOLOv8s model is well-suited for real-time seam localization, providing accurate weld detection while maintaining low latency for integration with the Q&C Vision pipeline. The trained model is subsequently used to generate weld ROIs during the seam-tracking process and for 3D fusion with Mech-Eye data.

III. Train the model to track the path for welding in 2D images

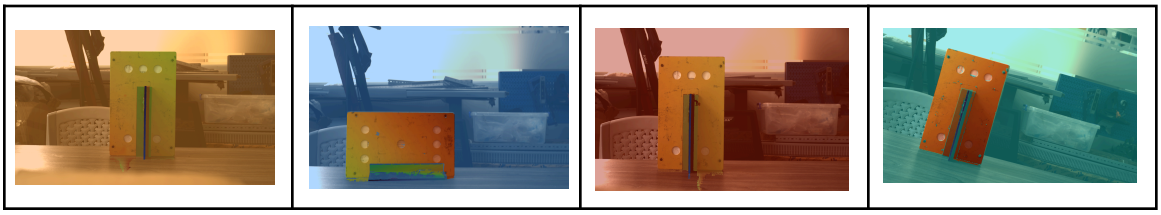
1. Capture Dataset from Mech-Eye 3D Camera

- After using YOLO v8s model to detect a weld path, we move on to seam tracking method.

- Images in this dataset are captured by using Mech-Eye Pro S 3D camera, which gives practicality.
- Example 2D images for training model:



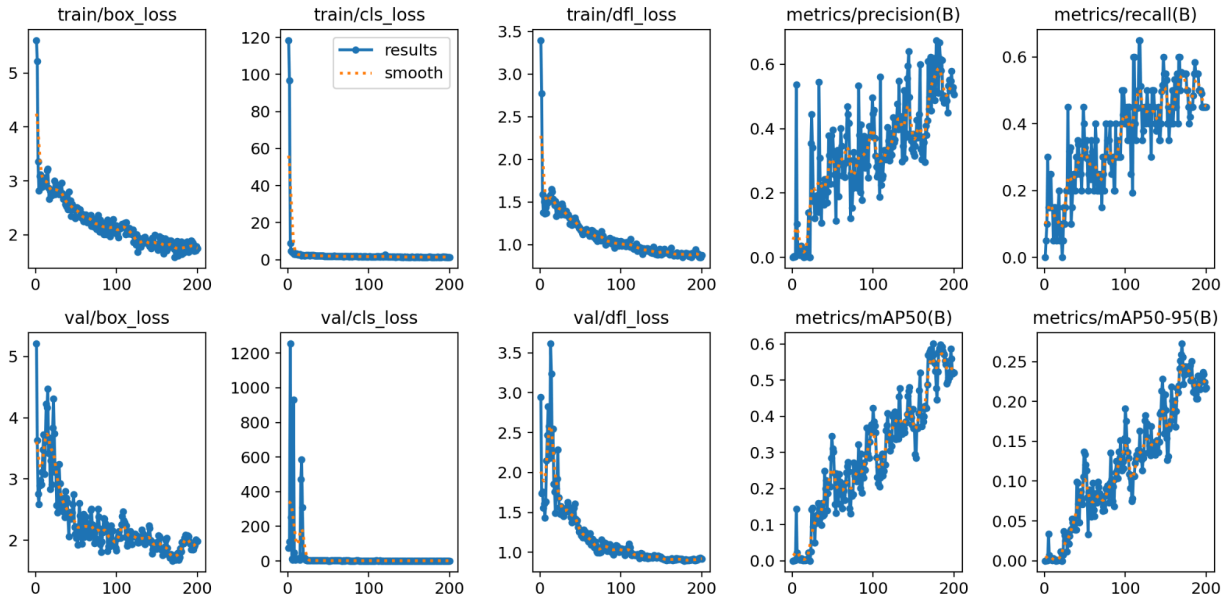
- Depth captures:



2. Train YOLO v8s model to get the ROI for 2D images

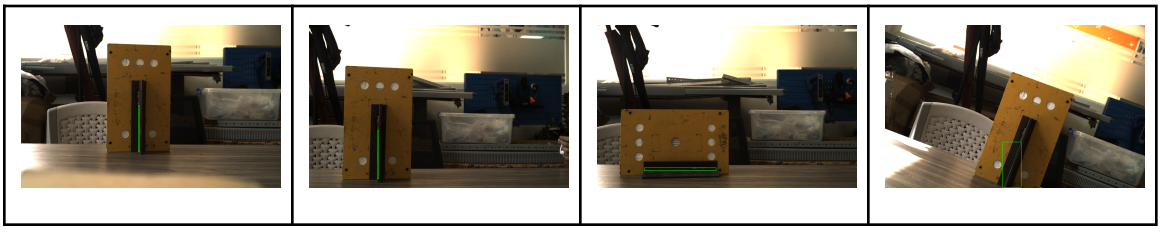
a. Training and testing phrase

- We use YOLO v8s model for better accuracy results and optimize training time.
- Training on Google Colab using T4 GPU
- Adjust training epochs to 200 with batch size 16.
- Result of model training phrase:



b. Capture ROI

- Use the trained model for capture ROI of the track in the original image.
- Example tests:

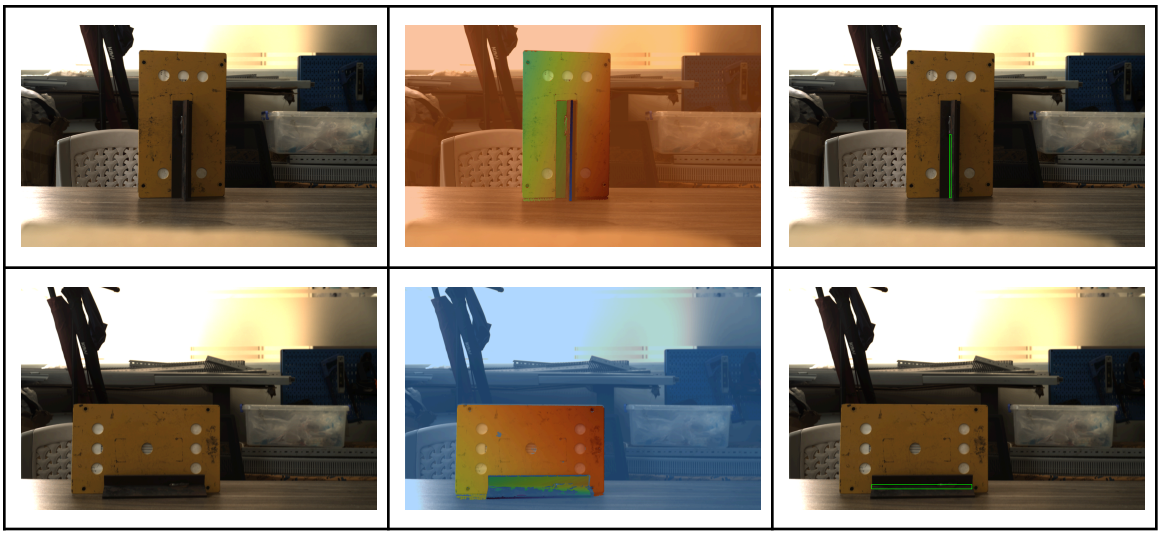


3. Train model to detect the track in the depth map image

a. Training and Testing phase

- The results of using 2D images for detecting seam track depend on the quality of the images captured, include brightness and camera angle. Which means the model cannot detect the track if the 2D images aren't captured correctly.
- With the help of depth map images, the model can read and detect the track correctly.

Original	Depth	Detected track



IV. Compare Measurements

1. Data Overview:

- The data includes 4 types of wrench sizes: 16, 18, 19, 24 (mm).
- Each type of wrench has 3 different predicted values -> It is the measurement parameters on various parts of the wrench.
- The error rate ranges from very low (~0.43%) to very high (54.00%).

2. Analysis of Error

a. In case of small error (< 1%): These predictions are quite accurate, with little deviation from the actual value:

- 16mm Wrench: Predicted 20.10 vs Actual 20.00 (0.50%)
- 18mm wrench: Predicted 21.90 vs actual 22.00 (0.45%)
- 19mm wrench: Predicted 15.86 vs actual 16.00 (0.88%)
- 24mm wrench: Predicted 28.12 vs actual 28.00 (0.43%)
- Remarks: This error may be due to the accuracy of the measurement method or rounding the number. In practice, an error of less than 1% is generally acceptable in industrial measurement.

b. In case of average error (1% - 10%)

The predictions have moderate errors, which may be due to slight differences in the measurement method:

- 16mm wrench: 13.83 vs 14.20 (2.61%)
- 18mm wrench: 15.68 vs 16.00 (2.00%)
- 19mm wrench: 21.82 vs 23.00 (5.13%)
- 24mm wrench: 20.17 vs 20.50 (1.61%)
- Remarks: These errors can come from the way the wrench is placed when measuring or errors in the image recognition algorithm.

c. Cases of Large Errors (> 10%)

- Wrench 16mm: Predicted 2.51, actual 2.00 → Error 25.50%
- Wrench 18mm: Predicted 2.90, actual 2.40 → Error 20.83%
- Wrench 19mm: Predicted 2.67, actual 2.50 → Error 6.80%
- Wrench 24mm: Predicted 4.62, actual 3.00 → Error 54.00%
- Comments: An error of 54.00% is very high, possibly due to measuring errors or the prediction algorithm not being accurate with small sizes. Large errors mainly occur in small measurements (2.00 - 4.62). This could be because small objects are more susceptible to errors when extracted from images.
- Solution direction:
  ➔ Since the measurement of the straight length part of the wrench is the smallest, we thought of separating the wrench into 3 parts: the straight line, the wrench head, and the circle for measurement.
  ➔ For the circular part of the wrench, we apply the equation of a circle, from which we deduce the diameter, plus the length of the original wrench line.

VI. Reference