

# CAAP Final Report

**Date of Report:** September 29, 2022

**Prepared for:** *U.S. DOT Pipeline and Hazardous Materials Safety Administration*

**Annual Period: From** *(September 29, 2019)* **to** *(September 29, 2022)*

**Contract Number:** 693JK31950002CAAP

**Project Title:** AI-Enabled Interactive Threats Detection using a Multi-Camera Stereo Vision System

**Prepared by:** Dr. Yongming Liu (PI), Dr. Yang Yu (Co-PI), Mr. Rahul Rathnakumar, Ms. Sampriti Neog, Mr. Gowtham Dakshnamoorthy, Mr. Rakesh Balamurugan

**Contact Information:**

Dr. Yongming Liu (PI), Email: [Yongming.Liu@asu.edu](mailto:Yongming.Liu@asu.edu)

Dr. Yang Yu (Co-PI), Email: [yangyu18@asu.edu](mailto:yangyu18@asu.edu)

## **Table of Contents**

1	EXECUTIVE SUMMARY.....	15
1.1	Project Status – Proposed vs. Delivered.....	15
1.2	Summary of Accomplishments.....	17
2	DEPTH MAP GENERATION AND CAMERA CALIBRATION PROCEDURES.....	19
2.1	Working Principle of Stereovision .....	19
2.2	Camera Calibration and Preliminary Case Studies on the Aptina AR0330 Camera ....	21
2.3	Design Criteria and Hardware Prototypes.....	27
2.3.1	Hardware prototype 1.....	27
2.3.2	Hardware prototype 2 .....	29
2.4	Depth Map Generation and Evaluation of the Intel RealSense D435i Camera .....	35
2.4.1	Analysis of the generated depth map for depth resolution limits .....	35
2.4.2	Surface roughness profile – Signal vs Noise at high resolution .....	38
2.5	Disparity Map Post-Processing Methods for Improving Depth Quality.....	40
2.5.1	Post-Processing of the Disparity map using Gaussian Pyramids.....	40
2.5.2	Disparity Map Post Processing using a Hole Filling Algorithm.....	42
2.6	Extremum Seeking Controller for Depth Map Optimization .....	43
3	DATA-DRIVEN TECHNIQUES FOR DEFECT DETECTION .....	50
3.1	Fully Supervised RGB-D Semantic Segmentation with Uncertainty.....	50
3.1.1	Feature Fusion using Fully Convolutional Networks .....	50
3.1.2	Epistemic and Aleatoric Uncertainty Quantification using MC-Dropout .....	58
3.1.3	Results and Discussion .....	65
3.2	Semi-Supervised Semantic Segmentation using Activation Map Interpolation .....	74
3.2.1	Background.....	74

3.2.2	Related Work.....	76
3.2.3	Methodology.....	78
3.2.4	Results and Discussion .....	82
4	DAMAGE PROGNOSTICS AND PHYSICS-BASED MODELS FOR INTERACTING THREATS.....	89
4.1	Semi-Empirical models for failure pressure and remaining useful life estimates .....	89
4.2	Kriging model trained on FEM data for failure analysis .....	91
4.2.1	Methodology.....	92
4.2.2	Results and Discussion .....	92
4.3	Assessment of X65 Gas Pipeline using FEA for Interacting Corrosion Pits and Comparison with ASME B31G .....	94
4.3.1	Verification and Validation of FEA analysis.....	94
4.3.2	Parametric Study.....	95
5	3D RECONSTRUCTION, SYSTEM INTEGRATION AND OPERATIONAL PROCEDURES.....	100
5.1	RGB-D 3D Reconstruction of a pipe using Simultaneous Localization and Mapping (SLAM).....	100
5.2	Orientation Estimation using a Kalman Filter versus a Complementary Filter.....	107
5.3	Robot Operation Procedure.....	113
5.4	Integrating semantic segmentation and risk assessment.....	114
6	CONCLUSIONS.....	117
7	APPENDIX A.....	119
7.1	Empirical evaluation of stereo-matching parameters for the Intel RealSense D435i Camera .....	119
7.1.1	Sensitivity analysis of object area measurements to the camera parameters.....	120

7.2	Sensitivity analysis of object depth measurements to the camera parameters.....	134
7.2.1	Methodology .....	134
7.2.2	Results and Discussion .....	136
8	APPENDIX B .....	146
8.1	Hardware prototype 1 .....	146
8.2	Hardware prototype 2 .....	150

## LIST OF FIGURES

Figure 2.1: (a) Projection from the image plane to 3D world coordinates is only known up to scale (b) Triangulation approach to find the corresponding point on the second camera for a point on the first camera.....	19
Figure 2.2: Dual Lens Aptina AR0330 USB Camera.....	23
Figure 2.3: Calibration images.....	23
Figure 2.4: Reprojection errors of calibration.....	24
Figure 2.5: Visualization of multi-plane camera calibration .....	24
Figure 2.6: Image pair captured by the stereo camera for Case Study 1: (a) Left camera image (b) Right camera image (c) Edge length of the cube (d) Disparity map for case study 1 .....	25
Figure 2.7: Location and 2D coordinates of the vertices of L1: (a) rectified left image; (b) rectified right image .....	26
Figure 2.8: Image pair captured by the camera for case study 2: (a) Left image (b) Right image (c) Computed disparity map (d) Computed depth map.....	27
Figure 2.9: (a) Camera housing module (b) Vehicle tank assembly (c) Robot arm assembly (d) Assembled robot carrier.....	28
Figure 2.10: (a) Original RGB image (b) Computed depth map.....	29
Figure 2.11: Demonstration of the angled view (a) RGB image (d) Depth map (original) (c) Depth map (filled).....	30
Figure 2.12: (a-b) CAD model of the robot prototype platform demonstrating the height adjustment mechanism using the PA-07 actuator. (c) Stereo camera mounted on the platform.....	31
Figure 2.13: Schematic diagram of the integrated hardware electronics.....	34
Figure 2.14: Integrated hardware prototype.....	34
Figure 2.15: (a) Sensitivity of the depth map to regions at various distances in (mm) to perturbations in disparity, focal length, and baseline. (b) Sensitivity of the depth to disparity perturbations for objects at various distances .....	36
Figure 2.16: Relating the length of an object in pixels to its real-world length .....	37
Figure 2.17 (a-c) RGB images for smooth, roughness level 1 and roughness level 2 surfaces respectively, Bottom Row (d-f) Depth maps in white to black scale for smooth, roughness level 1 and roughness level 2 surfaces respectively. (g-i) Depth histograms for smooth, roughness level 1 and roughness level 2 surfaces respectively .....	39

Figure 2.18: Filling missing values using a hierarchical pyramid approach (Adelson et al., 1984)	40
Figure 2.19: (a) Original Disparity Map (b) Disparity Map after Post-Processing with Gaussian Pyramids	42
Figure 2.20: Depth map hole filling algorithm demonstration (a) RGB Image (b) Depth map with holes (c) Post-processed depth map	43
Figure 2.21: Extremum seeking control loop	44
Figure 2.22: Demonstration 1: Baseline test case in a well-lit scene with the factory D435i parameter settings	47
Figure 2.23: Demonstration 2: Poor lighting conditions with bad parameter settings: Low exposure, gain and IR laser power	48
Figure 2.24: Demonstration 3: Test case with varying external illumination conditions starting with a dark environment with poor initial condition setting	49
Figure 3.1: Overview of the proposed network architecture	51
Figure 3.2: RGB-D Fusion Module	55
Figure 3.3: RGB-DNC Data Pre-processing Module	55
Figure 3.4: Demonstration of the curvature and normal maps for idealized surface formulations	57
Figure 3.5: DNC data derived from the point cloud representation	58
Figure 3.6: Bayesian Linear Model baseline uncertainties with Gaussian prior for (Top) 1 Training sample (Middle-1) 3 Training samples (Middle-2) 20 Training samples (Bottom) 100 Training samples: Left column – Total uncertainty; Middle column – Epistemic uncertainty; Right column – Aleatoric uncertainty	62
Figure 3.7: MC-Dropout uncertainties for (Top) 1 Training sample (Middle-1) 3 Training samples (Middle-2) 20 Training samples (Bottom) 100 Training samples: Left column – Total uncertainty; Middle column – Epistemic uncertainty; Right column – Aleatoric uncertainty	63
Figure 3.8: (a) Visualization of the two moons dataset with a visualization of the decision boundary when using MC-Dropout (b) Classification with uncertainty in the two moons dataset - Variation of uncertainties with sample size	65
Figure 3.9: (a) CrackForest dataset and (b) ASU Pipe Dataset	66
Figure 3.10: Uncertainty Evaluation for images from the ConcreteCrack dataset with a model	

trained on the CrackForest dataset: (a) Aleatoric Uncertainty and (b) Epistemic Uncertainty ....	66
Figure 3.11: Defect measurement demonstration on the pipeline sample .....	68
Figure 3.12 : Classwise Epistemic Uncertainty - F1 score calibration performance for the ASU Pipe RGB-D validation set across classes: Top row - Small ASU Pipe Dataset with a test set held-out (Left) Background (Middle) Cracks (Right) Pits, Bottom Row - Augmented ASU Pipe Dataset with test samples augmented from the training samples for verifying model calibration assuming the test set distribution is very close to the training data distribution (Left) Background (Middle) Cracks (Right) Pits .....	70
Figure 3.13: Demonstrative detections from the ASU pipe dataset: (Left to Right) Image, Ground Truth, Prediction and Total Uncertainty .....	71
Figure 3.14: Classwise Epistemic Uncertainty - F1 score calibration performance for the ConcreteCrack dataset by class: Top-Row: Original unaltered data (a) Background (b) Crack, Bottom-Row: Data with added Gaussian noise and random brightness, contrast, hue and saturation jitter for (c) Background (d) Crack .....	72
Figure 3.15: Demonstrative examples of detections in the CrackForest (Top-3 rows) and the ConcreteCrack Dataset (Bottom-3 rows). Images from Left to Right: Input, Prediction, Ground Truth (GT), Total Uncertainty .....	73
Figure 3.16: (a) Epistemic Uncertainty across various dropouts and input data types (b) Aleatoric Uncertainty across various dropouts and input data types (c) Mean-F1 across various dropouts and input data types .....	74
Figure 3.17: Cluster assumption demonstration by computing the patch wise Euclidean distance between a patch and its neighbors. Higher values are indicated with yellow: The left-most image is the input, followed by the distance maps (DM) of the following: the input image, the activation map from the end of convolutional Block 1, and the activation map from the end of Block 5....	79
Figure 3.18: Overall methodology for semi-supervised segmentation using interpolation consistency in the probability space: Blocks with the same colors denote related elements. ....	80
Figure 3.19: Demonstrative detections on the NEU validation set: (Left to Right) Image, Ground Truth, Predictions: 5% labeled training data, 10% labeled training data, 20% labeled training data, 50% labeled training data, 100% labeled training data.....	83
Figure 3.20: Demonstrative detections illustrating the effect of adding additional unlabeled data to the set and using the consistency loss. In this case, the comparison is made between the fully	

supervised case with 30 labeled samples, and the semi-supervised case with 30 labeled samples and 570 unlabeled samples. ....	84
Figure 3.21: Demonstrative detections on the CrackForest validation set: (Left to Right) Image, Ground Truth, 5% Labeled, 10% Labeled, 20% Labeled, 50% Labeled, Fully Labeled .....	86
Figure 4.1: Prediction of failure pressure with depth to wall thickness ratio for a carbon steel pipe .....	89
Figure 4.2: Comparing the cumulative failure probability of the surrogate model and the ASME B31G model .....	93
Figure 4.3: FEA simulation result where the limit load predicted by it is more conservative than the modified ASME B31G formula. ....	95
Figure 4.4: (a) True stress-strain curve for X65 (b) Plastic strain vs true stress in ANSYS .....	96
Figure 4.5: Von Mises Failure Criterion Across the ligament of the defect .....	97
Figure 4.6: Interaction distance between pits vs burst pressure .....	97
Figure 5.1: Overall system diagram for RTAB-Map (Labbé & Michaud, 2019) .....	100
Figure 5.2: RTAB-Map SLAM: Visual Odometry system components (Labbé & Michaud, 2019) .....	101
Figure 5.3: Frame stitching using SLAM in a feature-sparse pipeline environment .....	102
Figure 5.4: Demonstrative odometry measurements and inlier detection in real-time from the stationary test on the plastic pipe: (a) Angle data (b) Inlier ratio (c) Frame with defect features .....	104
Figure 5.5: Rotation of the camera on the axis of the plastic pipe causes an intermittent loss of odometry due to the sparsity of feature matches between subsequent frames .....	105
Figure 5.6: (a) Pitch angle of the camera (b) Inlier ratios for a rotation cycle on the corroded steel pipe sample. ....	106
Figure 5.7: Demonstration of pipe scan with rotation of the camera about the axis of the pipe, and translation of the robot along the pipe. The trajectory of the camera system is estimated and shown in white .....	107
Figure 5.8: Pitch, yaw and roll angles computed using the motion module .....	110
Figure 5.9: (a) Angular velocity measurement data from gyroscope (b) Ground truth orientation for the sensor .....	110
Figure 5.10: (a) Orientation prediction and ground truth using Kalman Filter (b) Orientation	



prediction and ground truth using Complementary Filter.....	111
Figure 5.11: Flow chart of Hardware-Software Integration .....	112
Figure 5.12: Demonstration of inspection in a sample pipe with pit and crack defects. (a) Image (b) Detection (c) Uncertainty (d) Remaining Useful Life estimate– Corrosion .....	115
Figure 5.13: Converted model in ANSYS and Von-Mises stress around the internal defect.....	116
Figure 7.1: Intel Realsense D435i camera system (Intel RealSense, 2022) .....	119
Figure 7.2: Original RGB image.....	121
Figure 7.3: Contour extraction from the depth map .....	121
Figure 7.4: (a) Original RGB image of a coin (b) Image with overlaid contour .....	122
Figure 7.5: Geometry of a camera lens.....	122
Figure 7.6: Contour area calculation for depth map .....	124
Figure 7.7: Representative contours for various values of gain.....	126
Figure 7.8: Contour area error variation with respect to camera gain .....	126
Figure 7.9: Depth frame for gain in range (16-55 dB).....	127
Figure 7.10: Representative contours for various values of gains .....	128
Figure 7.11: (a) Contour area error variation for varying exposure $wm2s$ (b) Distorted raw depth frame for exposure $>45000 W$ .....	128
Figure 7.12: Representative contours for various values of Laser Power .....	129
Figure 7.13: Contour area error variation for varying laser power.....	129
Figure 7.14: Representative contours for various values of second peak threshold.....	130
Figure 7.15: (a) Contour area error variation for varying the second peak threshold (b) Depth map for second peak threshold beyond 600 pixels .....	130
Figure 7.16: Representative contours for various values of neighbor threshold .....	131
Figure 7.17: (a) Contour area error variation for varying neighbor threshold (b) Distorted depth map for neighbor threshold above 400 pixels.....	131
Figure 7.18: Representative contours for various values of disparity shift .....	132
Figure 7.19: (a) Contour area error variation for varying disparity shift (b) Distorted raw depth map beyond disparity shift of 120 pixels .....	132
Figure 7.20: Contour capture at an object thickness of (a) 1.8cm and (b) 1cm.....	134
Figure 7.21: Object of thickness 18 mm thickness used for sensitivity analysis.....	135
Figure 7.22: RGB and depth map corresponding to the 3D map.....	135

Figure 7.23: Contour extraction process for thickness estimation.....	136
Figure 7.24: Contours across gains (dB) (a) 16 (b) 36 (c) 56 (d) 76 (e) 96 (f) 116.....	136
Figure 7.25: Plot of error in thickness estimation versus camera gain .....	137
Figure 7.26: Contours across exposures ( $Wm2s$ ): (a) 0 (b) 500 (c) 1000 (d) 7500 (e) 14500 (f) 16000.....	137
Figure 7.27: Plot of exposure versus error in thickness estimation .....	138
Figure 7.28: Contours and depth map for exposure below 2000 $Wm2s$ .....	138
Figure 7.29: Contours and depth map for exposure between 2000-16000 $Wm2s$ .....	139
Figure 7.30: Contours and depth map for exposure above 16000 $Wm2s$ .....	139
Figure 7.31: Contours across second peak thresholds (pixels):.....	140
Figure 7.32: Plot of second peak threshold versus error in thickness estimation.....	140
Figure 7.33: Contours across disparity shift (pixel): (a) 0 (b) 50 (c) 100 (d) 150 (e) 200 (f) 250 .....	141
Figure 7.34: Plot of disparity shift versus error in thickness estimation .....	141
Figure 7.35: Contours across neighbor threshold (pixel): (a) 0 (b) 50 (c) 100 (d) 150 (e) 200 (f) 250.....	142
Figure 7.36: Plot of neighbor threshold versus error in thickness .....	142
Figure 7.37: Contours across laser power (W): (a) 0 (b) 50 (c) 100 (d) 150 (e) 200 (f) 250.....	143
Figure 7.38: Plot of laser power versus error in thickness.....	143
Figure 7.39: (a) 3-D visualization of a 5 mm thickness coin (b) 3-D visualization of a coin of a thickness of 1.75 mm .....	144
Figure 7.40: Contour extraction from the depth map for the object of 5 mm thickness.....	145
Figure 7.41: Contour extraction from the depth map for the object of 1.74 mm thickness.....	145
Figure 7.42: (a) 3-D RGB textured point cloud; (b) Depth map for the object of 8 mm diameter (c) 3-D RGB textured point cloud; (d) Depth map for the object of 7.5 mm diameter.....	146
Figure 8.1: Fisheye stereo camera .....	150

## LIST OF TABLES

Table 2.1: Distance estimation results .....	26
Table 3.1: Comparing the performance of various types of input data at a constant dropout value .....	69
Table 3.2: Dataset sizes.....	83
Table 3.3: Metrics computed for the validation set at various proportions of labeled samples, with a performance comparison between the loss ramp-up approach and the equally weighted losses approach.....	85
Table 3.4: Metrics computed for the validation set at various proportions of labeled samples, with a performance comparison between the semi-supervised model and the fully supervised model	87
Table 3.5: Comparison of the results obtained in the test set for the NEU dataset .....	87
Table 3.6: Comparison of results obtained in the test set for the CrackForest dataset .....	88
Table 4.1: Input parameters for the surrogate model for the baseline experiment .....	92
Table 4.2: Experimental details for the defect size configuration .....	94
Table 4.3: Summary of datapoints used for the parametric study .....	98
Table 4.4: Comparison of the ASME B31G burst pressure and FEA burst pressure .....	99
Table 7.1: Contour area at various distances in pixel coordinates and world coordinates .....	125
Table 7.2: Variation of error in area computation with resolution .....	133
Table 7.3: Variation of error in area computation with surface area change.....	133
Table 7.4: Variation of error in area computation with change in object thickness .....	134
Table 7.5: Error in thickness estimation versus Resolution.....	144
Table 8.1: Components list for the height adjustment mechanism.....	150
Table 8.2: Technical specifications of the 12V brushed DC motors .....	151
Table 8.3: Technical specifications of the PA-07 linear actuators .....	151
Table 8.4: Truth Table for L298N Motor Driver Module for Direction Control .....	151
Table 8.5: Technical specifications of the DRV8834 Stepper Motor Driver Module.....	152
Table 8.6: Truth Table for DRV8834 Stepper Motor Driver Module for Micro stepping .....	152
Table 8.7: PS4 Controller button mappings to robot functions. ....	152

## LIST OF ABBREVIATIONS

<b>Abbreviation</b>	<b>Definition</b>
ASIC	Application-Specific Integrated Circuit
ASME	American Society of Mechanical Engineers
ASU	Arizona State University
BLUP	Best Linear Unbiased Predictor
BRIEF	Binary Robust Independent Elementary Features
CAM	Class Activation Map
CFD	CrackForest Dataset
CNN	Convolutional Neural Network
DM	Distance Map
DNC	Depth Normal Curvature
DOF	Degrees of Freedom
ESC	Extremum Seeking Control
FAST	Features from Accelerated Segment Test
FCN	Fully Convolutional Network
FEA	Finite Element Analysis
GAN	Generative Adversarial Network
GAP	Global Average Pooling
GFTT	Good Features To Track
GPU	Graphics Processing Unit
HPF	High Pass Filter
ICT	Interpolation Consistency Training
ILI	In-Line Inspection

IMU	Inertial Measurement Unit
IR	Infrared
KL	Kullbak Leibler
LBP	Local Binary Pattern
LTM	Long Term Memory
MC	Monte Carlo
MLE	Maximum Likelihood Estimate
MSE	Mean Squared Error
NEU	Northeastern University (China)
NLL	Negative Log Likelihood
NNDR	Nearest Neighbor Distance Ratio
NYU	New York University
ORB	Oriented FAST and Rotated BRIEF
PLA	Polylactic Acid
PS2	PlayStation2
PS4	PlayStation4
PWM	Pulse Width Modulation
RANSAC	Random Sample Consensus
RMSE	Root Mean Squared Error
RTAB	Real-Time Appearance Based
SDK	Software Development Kit
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SMYS	Specified Minimum Yield Strength

STM	Short Term Memory
SVD	Singular Value Decomposition
SVM	Support Vector Machine
USB	Universal Serial Bus
VGG	Visual Geometry Group
WM	Working Memory
YOLO	You Only Look Once

# 1 EXECUTIVE SUMMARY

## 1.1 Project Status – Proposed vs. Delivered

Tasks	Proposed	Delivered
Task 1.1	Reports and codes for stereo vision algorithm developed for generating a depth map of pipeline surface	<ol style="list-style-type: none"><li>1. RealSense SDK is used for the baseline results in depth map generation.</li><li>2. Extremum-seeking control algorithm codes are delivered to improve depth map quality automatically.</li></ol>
Task 1.2	Reports and drawings for optimal design criteria and calibration procedures of the prototype device	<ol style="list-style-type: none"><li>1. Hardware calibration was performed, and design criteria were outlined for in-line inspection with a single rotating camera system.</li></ol>
Task 1.3	Reports and specifications for the hardware requirements for desired performance and future commercialization plans	<ol style="list-style-type: none"><li>1. Sensitivity analysis and experiments for depth camera performed and results documented.</li><li>2. Analytical analysis of depth camera to determine depth and area resolution documented and contextualized in the framework of ILI.</li></ol>
Task 1.4	Full-scale prototype device and reports on preliminary testing in laboratory experiments	<ol style="list-style-type: none"><li>1. Proposed hardware components have been specified for the robotic platform with codes delivered. The prototype is built as a single unit and evaluated in the pipeline environment.</li></ol>

<p>Task 2.1/Task 2.2</p>	<p>Reports and codes for the trained deep learning model and detection performance for pipeline anomaly detection</p>	<ol style="list-style-type: none"> <li>1. Dataset collection for ILI defect detection model.</li> <li>2. Proposed a fully supervised RGB-D fusion network for semantic segmentation with MC Dropout uncertainty.</li> <li>3. Proposed a point-cloud derived data stream for the CNN to leverage the geometrical features of the pipe – called the DNC representation.</li> <li>4. Empirical comparison between various encoder backbones for semantic segmentation.</li> <li>5. Defect quantification using semantic segmentation results for crack and corrosion defects.</li> <li>6. Proposed a semi-supervised learning algorithm for defect detection based on interpolation consistency training.</li> </ol>
<p>Task 2.3</p>	<p>Reports and tools for the physics-based models developed for assessing interactive threats and damage prognostics</p>	<ol style="list-style-type: none"> <li>1. Utilized the ASME B31G and NG-18 models to estimate remaining useful life.</li> <li>2. Application of a Kriging-based surrogate model trained on FEM data to estimate remaining useful life for crack and corrosion defects.</li> <li>3. Interacting threat assessment for corrosion pits using FEM and comparison with ASME B31G.</li> </ol>



Task 2.4	Reports on demonstration study of the hardware-software integrated prototype device for ILI of actual pipeline systems	1. Reporting, demonstration, and code for the integrated functioning of the prototype.
----------	--	--

## 1.2 Summary of Accomplishments

### Journal Papers

- Epistemic and Aleatoric Uncertainty Quantification for Defect Detection using an RGB-D Fusion Network (*Revisions in progress*)
- Semi-Supervised Surface Defect Segmentation with Activation Map Interpolation (*In Preparation*)

### Poster presentations

- AI-Enabled Interacting Threats Detection using a Multi-Camera Stereo-Vision System, PHMSA R&D Forum 2020
- Threat Detection using Active Stereo for In-Line Inspection in Gas Pipelines, PRCI Fall Technical Meeting 2022

### Doctoral Symposium

- AI-Enabled Interacting Threat Detection using a Multi-Camera Stereo-Vision System, PHM Society Conference, 2020

### Educational Accomplishments

- Sampriti Neog (MS) Mechatronics, Robotics and Automation Engineering (2021)

### Student Contributors

- Rahul Rathnakumar (PhD student)
- Rakesh Balamurugan (MS student)
- Chinmay Dixit (MS student)
- Utkarsh Pujar (MS student)
- Omar Serag (Undergraduate student)

- Kailing Liu (MS student)
- Sampriti Neog (MS Student)
- Karthikeya Vemullapalli (MS student)
- Rohith Kalyan Kavadappu (MS student)
- Gowtham Dakshnamoorthy (MS student)
- Abhishek Srinivas Loganathan (MS student)

## 2 DEPTH MAP GENERATION AND CAMERA CALIBRATION PROCEDURES

### 2.1 Working Principle of Stereovision

Stereovision relies on matching corresponding points in one camera to another camera. This requires camera calibration, followed by triangulation. We briefly explain this process here to help readers from fields outside computer vision. Existing literature in photogrammetry has extensive descriptions that can be referred to for further details (Hartley & Zisserman, 2004).

It can be inferred from Figure 2.1 that a point in an image can be projected along a ray to a point in the 3D space and vice versa. The point determines the ray equation on the image plane and the camera's optical center. The projection of a pixel on the image plane onto the 3D space is only determined by the equation of the ray projected from the camera. In other words, we know the position of the 3D point subject to a scaling parameter for the depth. On the other hand, we can completely determine the position of a projected pixel from the 3D space provided we know the location and orientation of the camera's image plane. In Figure 2.1, there are two calibrated cameras. In this context, calibration is the process of determining the camera's internal parameters and the camera's position and orientation. The calibration is represented with calibration matrices  $P_1$  and  $P_2$  for the first and second cameras, respectively. If this matrix is known, it is possible to project a 3D point in the scene to a pixel on the camera, provided the camera's field of view is high enough.

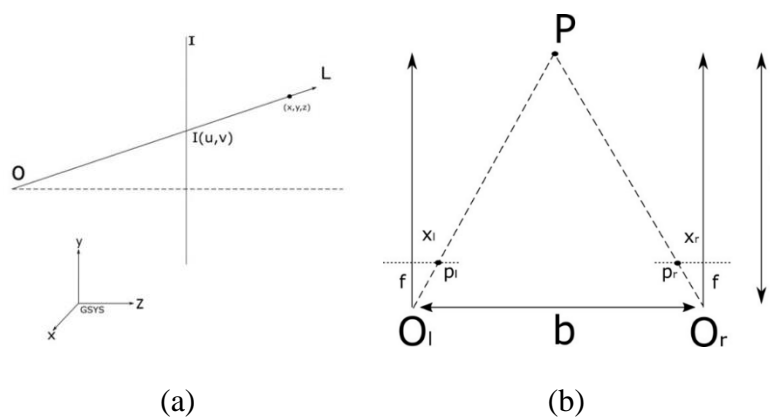


Figure 2.1: (a) Projection from the image plane to 3D world coordinates is only known up to scale (b) Triangulation approach to find the corresponding point on the second camera for a point on the first camera.

$$x = Px \tag{2.1}$$

$$P = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_1 & r_2 & r_3 & t_1 \\ r_4 & r_5 & r_6 & t_2 \\ r_7 & r_8 & r_9 & t_3 \end{bmatrix} \tag{2.2}$$

In Equations 2.1 and 2.2 the pixel coordinate  $x$  is obtained by linearly transforming the world coordinates  $X$  using the calibration matrix  $P$ . The calibration matrix consists of the intrinsic parameters represented by  $f, p_x$  and  $p_y$  and the relative rotation matrix and translation vector from a common coordinate system given by the  $r_i$  and  $t_i$  respectively.

The inverse problem involves computing a point in the world corresponding to a point in the image. Given two cameras with known calibration matrices, an estimate without scale ambiguity can be made using triangulation. A point in the image plane of one camera can lie anywhere on a ray projected from the image plane. The intersection of the ray from the second camera with the ray from the first camera provides the corresponding match. The 3D point and the rays to the 3D point from each camera plane constitute the epipolar plane. The epipolar plane is constrained to be hinged to the line joining the optical centers of the two cameras, known as the baseline. In general, the correspondences of a point in the first image plane lie on a line called the epipolar line on the second image plane. This epipolar line is used for the matching point search. Determining the equation of this epipolar line requires a transformation from a point in the first image plane as shown in Equations 2.3 and 2.4:

$$l' = Ex, x'^T l' = 0 \tag{2.3}$$

$$E = t X R \tag{2.4}$$

$t$  and  $R$  are the relative translation vector and rotation matrix, respectively, for each image plane to obtain the essential matrix  $E$ .

The search space for the matching process is made smaller by rectification - projecting the images from both cameras onto a common image plane. This puts the epipoles of both image planes at infinity. All epipolar lines are, therefore, parallel. The consequence of this projection is that the second image is shifted along with a line relative to the first image, removing the 3 relative rotational degrees of freedom and 2 translational degrees of freedom. Figure 2.1 (b) shows the top-

down view of the rectified configuration of the two-camera system. Using similar triangles, we can derive the depth of a point using Equations 2.5 and 2.6:

$$\frac{b + x_l - x_r}{z - f} = \frac{b}{z} \quad 2.5$$

$$z = \frac{fb}{d} \quad 2.6$$

where  $b$  is the baseline,  $f$  is the focal length,  $z$  is the depth of the point in the world frame,  $x_l$  and  $x_r$  are the horizontal coordinates in the image plane, and  $d$  is the disparity.

## 2.2 Camera Calibration and Preliminary Case Studies on the Aptina AR0330 Camera

A camera projects 3D points in the real-world to 2D points on an image plane by using the following mapping:

$$u = K[R|T]X \quad 2.7$$

where  $\mathbf{K}$  is the 3-by-3 camera intrinsic matrix comprised of intrinsic parameters, including the focal length and principle point;  $\mathbf{R}$  and  $\mathbf{T}$  are the 3-by-3 rotation matrix and 3-by-1 translation vector (extrinsic parameters), respectively; the vector  $\mathbf{u} = [x \ y \ 1]$  represents the 2D coordinates of the points in the image plane;  $\mathbf{X} = [X \ Y \ Z \ 1]$  represents the 3D coordinates of the points in the real world. Equation 2.7 can be simplified to a form similar to Equation 2.1 as:

$$u = PX \quad 2.8$$

where  $\mathbf{P}$  is the 3x4 camera projection matrix. In a stereo vision system, an object point in the real world is mapped onto image planes by the two cameras as:

$$u = P_1X \quad 2.9$$

$$v = P_2X \quad 2.10$$

where  $\mathbf{X}$  is the 3D coordinate of the object point;  $\mathbf{P}_1$  and  $\mathbf{P}_2$  are the camera projection matrices of

the two cameras, respectively; and  $\mathbf{u}$  and  $\mathbf{v}$  are the 2D coordinates of the object point in the image plane of cameras 1 and 2, respectively. Recall that the two vectors in the same direction have a cross product of zero. Thus, we have:

$$\mathbf{u} \times P_1 \mathbf{X} = 0 \quad 2.11$$

$$\mathbf{v} \times P_2 \mathbf{X} = 0 \quad 2.12$$

which can be expressed as:

$$\mathbf{u} \times P_1 \mathbf{X} = \begin{bmatrix} y_u p_1^{3T} - p_1^{2T} \\ p_1^{1T} - x_u p_1^{3T} \\ x_u p_1^{2T} - y_u p_1^{1T} \end{bmatrix} \mathbf{X} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad 2.13$$

$$\mathbf{v} \times P_2 \mathbf{X} = \begin{bmatrix} y_v p_2^{3T} - p_2^{2T} \\ p_2^{1T} - x_v p_2^{3T} \\ x_v p_2^{2T} - y_v p_2^{1T} \end{bmatrix} \mathbf{X} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad 2.14$$

where  $\mathbf{p}_1^{nT}$  and  $\mathbf{p}_2^{nT}$  are the transpose of the  $n$ th row of the camera projection matrix  $\mathbf{P}_1$  and  $\mathbf{P}_2$ , respectively. Combining Equation 2.13 and Equation 2.14 gives:

$$\begin{bmatrix} y_u p_1^{3T} - p_1^{2T} \\ p_1^{1T} - x_u p_1^{3T} \\ y_v p_2^{3T} - p_2^{2T} \\ p_2^{1T} - x_v p_2^{3T} \end{bmatrix} \mathbf{X} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad 2.15$$

Equation 2.15 represents a homogeneous linear system that can be solved using Singular Value Decomposition (SVD) to obtain the 3D coordinate  $\mathbf{X}$ . Once the coordinates of two critical points,  $\mathbf{X}_1$  and  $\mathbf{X}_2$  are known, we can then compute the distance between the two points as:

$$dist = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2 + (Z_1 - Z_2)^2} \quad 2.16$$

For the experiment, a dual-lens USB webcam shown in Figure 2.2 is adopted. Each camera has a resolution of 640x480. To obtain the camera projection matrix, multi-plane calibration is conducted. The calibration image set includes 15 pairs of images of a chessboard placed at different

angles and distances from the camera, as shown in Figure 2.3. The calibration is done using Zhang’s Camera Calibration Algorithm (Z. Zhang, 2000). The reprojection errors of the calibration are plotted in Figure 2.4. The reprojection errors are small, with an average error of approximately 0.15, indicating that accurate calibration was performed. A visualization of the calibration is shown in Figure 2.5. The calibrated camera parameters, including the camera projection matrix, the relative orientation between cameras, and the rectification rotation matrix, were stored for later use in distance estimation.



Figure 2.2: Dual Lens Aptina AR0330 USB Camera

The purpose of the experiment is to verify that the developed algorithm can be used to characterize the pipeline defect in terms of its size and depth. For this purpose, two case studies were designed, including one case for size estimation and the other case for depth estimation. A Rubik's cube was adopted to represent the ‘defect’ in this experiment.

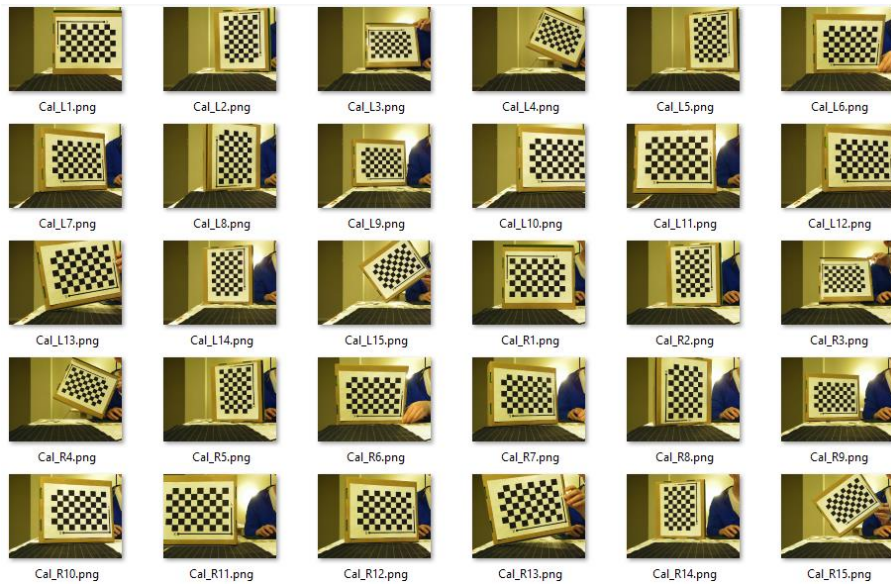


Figure 2.3: Calibration images

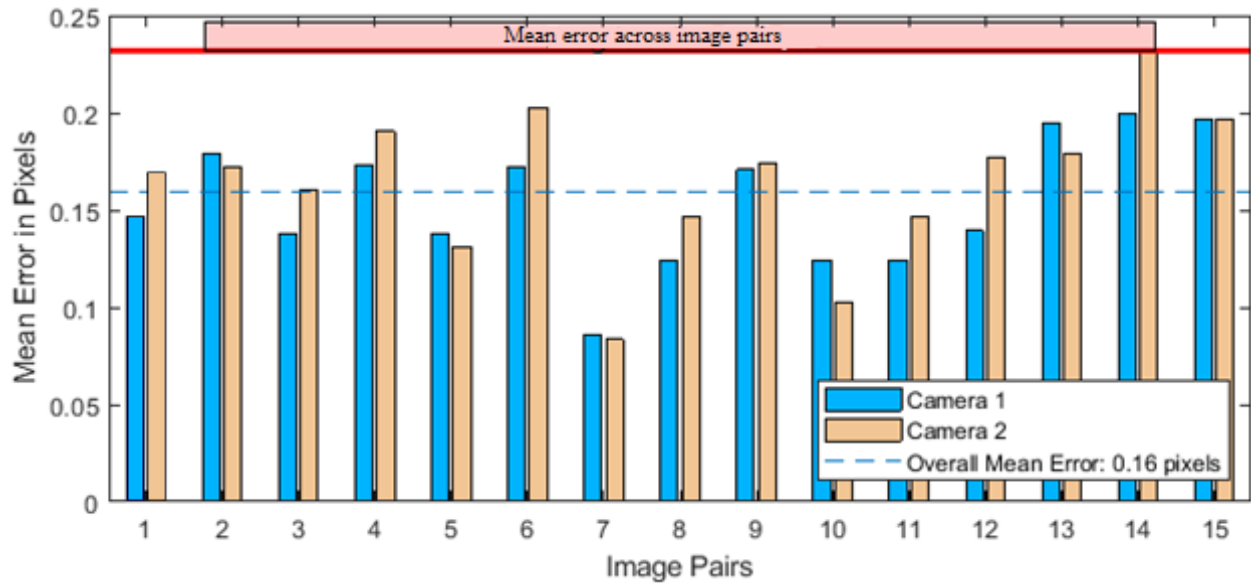


Figure 2.4: Reprojection errors of calibration

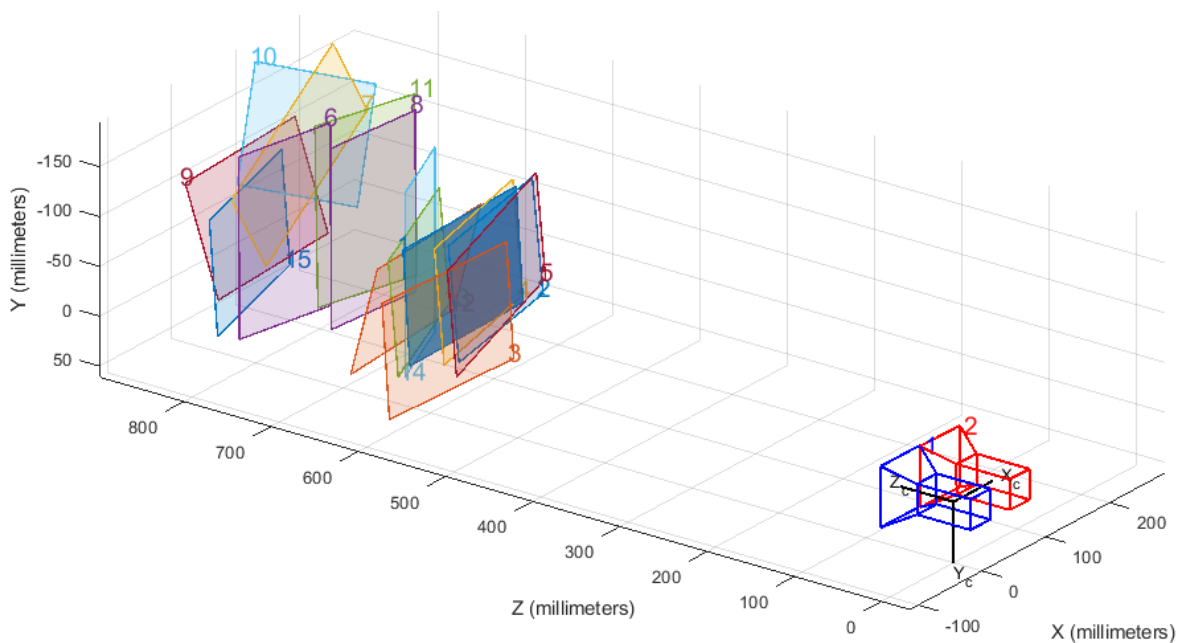


Figure 2.5: Visualization of multi-plane camera calibration

### Case study 1

For Case study 1, the Rubik's cube is placed on a shelf, as shown in top row of Figure 2.6. The top two layers of the cube are rotated to form two inclined surfaces. The objective is to estimate the edge length of each layer of the cube as denoted by  $L_1$ ,  $L_2$ , and  $L_3$ , shown in Figure 2.6 (c).



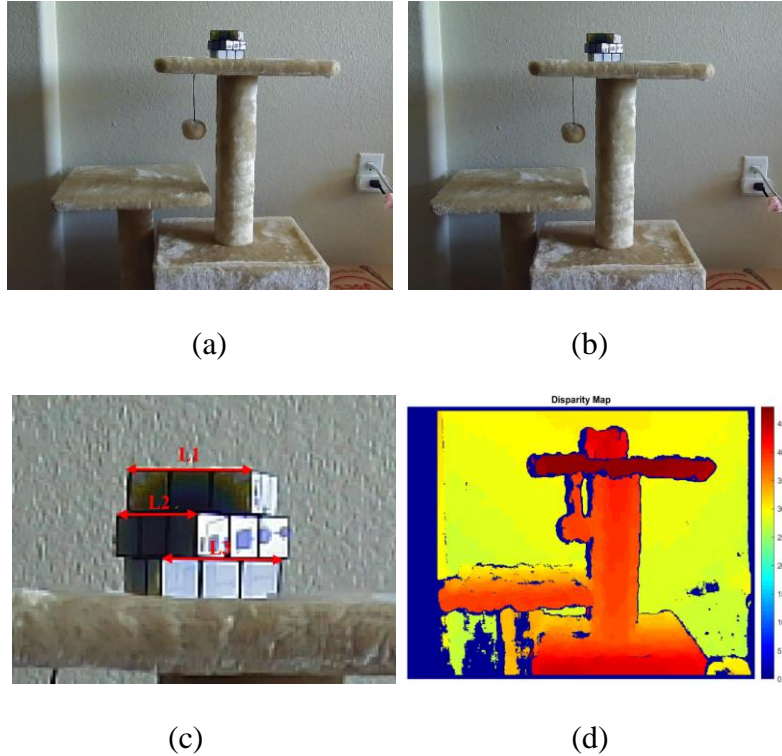


Figure 2.6: Image pair captured by the stereo camera for Case Study 1: (a) Left camera image (b) Right camera image (c) Edge length of the cube (d) Disparity map for case study 1

The captured image pair is rectified, and the disparity map is obtained using semi-global matching algorithm as shown in Figure 2.6 (d), where the outline of the shelf and cube can be seen from the disparity map. The procedures for distance estimation are illustrated here by using L1 as an example. First, the 2D coordinates of the vertices are obtained from the rectified images, as shown in Figure 2.7. It can be seen that: (1) the 2D pixel coordinates of the left vertex of L1 are [331 42] and [288 42] in the left and right images, respectively; (2) the 2D pixel coordinates of the right vertex of L1 are [376 42] and [333 42] in the left and right images, respectively. Then, the 3D coordinates of the vertices are computed using the algorithm developed above. In this case, the 3D coordinates of the left and right vertices of L1 are estimated to be (21.75, -278.11, 1129.49) mm and (80.01, -277.14, 1125.21) mm, respectively. Finally, the edge length is computed using Equation 2.16. The edge length L1 is estimated to be 5.84 cm, while the true length is measured at 5.62 cm. The same procedures are repeated for all three edges shown in Figure 2.6 (d), and the results are summarized in Table 2.1. The developed algorithm can estimate the edge length with good accuracy. The error for L2 is larger due to the higher degree of inclination.



Figure 2.7: Location and 2D coordinates of the vertices of L1: (a) rectified left image; (b) rectified right image

Table 2.1: Distance estimation results

Edge	Left vertex coordinate (mm)	Right vertex coordinate (mm)	Distance (cm)	Error (%)
L1	(21.75, -278.11, 1129.49)	(80.01, -277.14, 1125.21)	5.84	3.97
L2	(16.91, -244.43, 1154.33)	(51.72, -233.52, 1102.62)	6.33	12.61
L3	(37.30, -236.19, 1127.57)	(94.17, -235.51, 1123.99)	5.70	1.40

### Case study 2

The purpose of Case study 2 is to demonstrate that the developed algorithm can estimate the depth of 'defect'. For this purpose, the Rubik's cube is placed against the wall to simulate a defect protruding out of the pipeline surface, as shown in Figure 2.8 (a) and (b). In this case, the depth to be estimated is equal to the edge length of the cube.

From the camera calibration, the focal length is 869.31 pixels, and the baseline distance between the two cameras is measured at 59.77 mm. The depth of all pixels can be obtained, and the depth map is plotted in Figure 2.8 (c). The depth of four representative points is denoted in Figure 2.8 (d). It can be seen the depth of the wall is approximately 143.1 cm while the depth of the cube is approximately 137.4 cm, based on which the depth of the 'defect' is estimated to be 5.7 cm which is very close to the actual value of 5.62 cm. It should be noted that the estimated depth is approximate values because the plane of the camera is aligned parallel to the wall only by hand. Therefore, more rigorous alignment is performed to ensure that the obtained depth reflects the shortest distance from the camera plane to the defect surface.

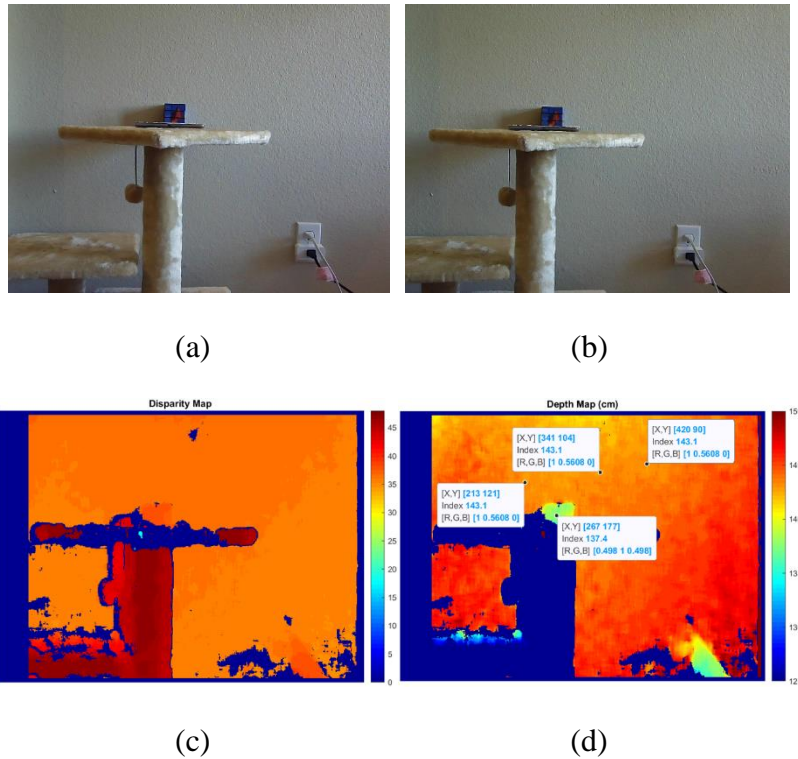


Figure 2.8: Image pair captured by the camera for case study 2: (a) Left image (b) Right image  
(c) Computed disparity map (d) Computed depth map

## 2.3 Design Criteria and Hardware Prototypes

### 2.3.1 Hardware prototype 1

The initial hardware prototype was in line with the proposed multi-camera configuration. The inspection device hosts 3 stereo-cameras, each covering a portion of the pipe's Field of View (FoV). A housing module holds these cameras in place. The housing module is fixed onto a rotating arm connected to a tracked robot base.

The camera housing module will be used to host three pairs of stereo cameras. Figure 2.9 (a) shows the camera setup with one camera. The camera bed where the cameras were to be placed was designed according to this area's requirement to accommodate them. The field of view of the camera is  $180^{\circ}$  and the angle at which the cameras were placed was to be calculated. Since the field of view of the camera was  $180^{\circ}$ , we placed it at an optimal angle so that it would cover a major portion of the pipe's inner surface. To accommodate this, the cameras are placed at  $37.5^{\circ}$

with respect to the pipe's inner surface. This was because we need to stitch the images obtained from each pair of stereo cameras, and each pair of the stereo camera needed to cover at least  $120^\circ$ . We must also cover the region directly perpendicular to the camera on the pipe surface. Thus, we chose an angle higher than  $120^\circ$ , which was  $142.5^\circ$ . We selected a commercial eight degrees of freedom (DOF) vehicle robot with 8-Axis RC robotic arm for the robot carrier. Details of this carrier and camera are listed in Appendix B. The camera module and the robot platform are shown in Figure 2.9.

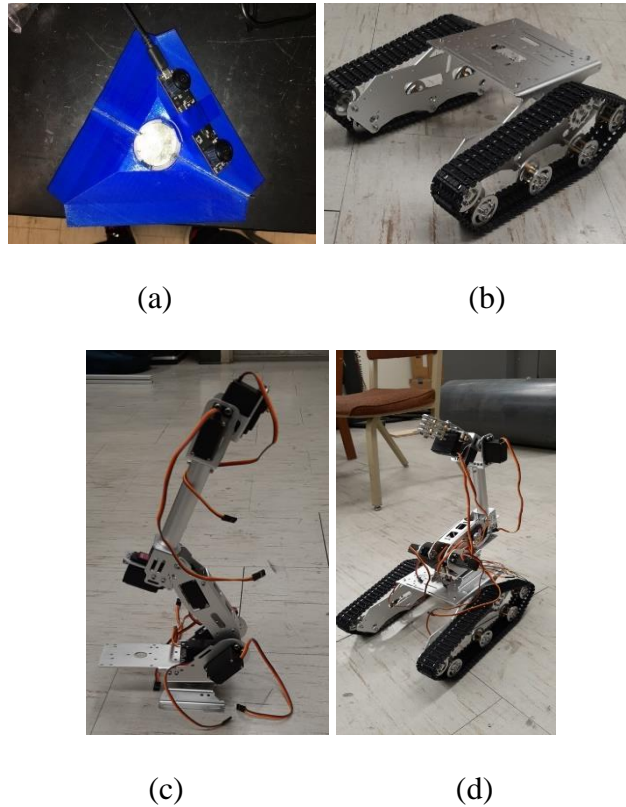


Figure 2.9: (a) Camera housing module (b) Vehicle tank assembly (c) Robot arm assembly (d) Assembled robot carrier

A demonstration with simulated defects is conducted to visualize the pipe surface with simulated defects. For the demonstration, the camera housing module was elevated to the axis center of the pipe, and a ruler was placed at the end which was to be observed to show different contours. The surface also had an irregularity to simulate what rust would look like. This was done to demonstrate that the depth map generated could identify both contour types, i.e., the crack propagating inwards and rust protruding outwards. Once the setup was complete, the stereo vision code was run, and

the results were generated. The RGB image of the pipe is shown in Figure 2.10 (a). This shows the two types of contours, i.e., the ruler and the surface irregularity. It should be noted that this image is distorted around the edges as a fisheye camera was used. The depth map was generated and shown in Figure 2.10 (b). The contours of the ruler and two surface irregularities can be clearly seen from the depth map. The ruler was placed to simulate protruding irregularities in the pipe, and its edges worked as cracks to show the depth in the cracks. The other irregularity shown worked as simulated rust in the pipe. In addition, the depth map generated is sensitive to the lighting condition.

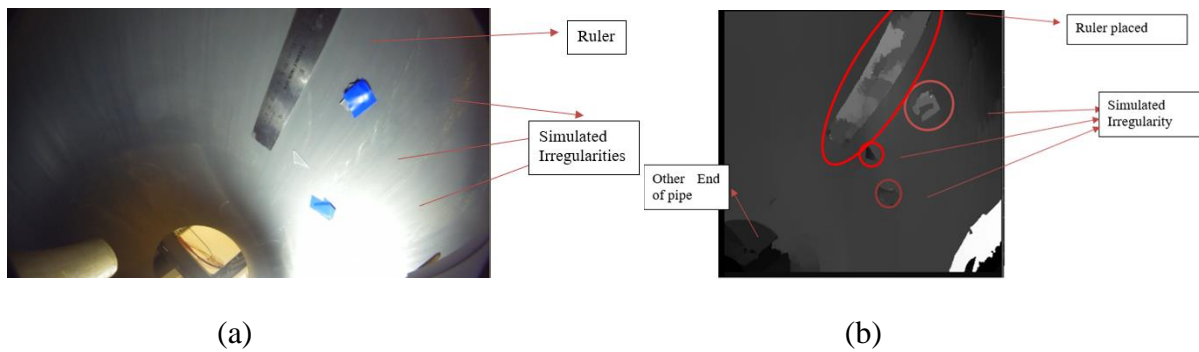


Figure 2.10: (a) Original RGB image (b) Computed depth map

### 2.3.2 *Hardware prototype 2*

This prototype is a significant modification of Hardware prototype 1. A multi-camera system would have been useful to cover an effective field of view greater than any one camera, provided the placement of these cameras was made correctly. However, In-Line Inspection (ILI) tools with cameras not oriented normally towards the pipe wall reduce the detection capabilities in the depth map, as shown in Figure 2.11. Figure 2.11 has a nut that is 20mm long and has a maximum depth of approximately 10mm. The depth map does not indicate the object is present, despite it being much larger than the lowest pixel resolution provided by the depth map. Furthermore, the oblique viewing angle of the camera presents difficulties for extracting normal and curvature features from the point cloud that can indicate bumps in the surface. A simplified design is therefore proposed. A rotating single-camera system replaces the fixed multi-camera system. The rotation is about an axis parallel to the pipe's axis, so the camera is always facing normally to the pipe surface. This means that any surface defect that changes the pipe radius will be more visible, as the radial change in depth is captured directly by the depth map, which maps the z-component of the point cloud.

Additionally, the proposed design guarantees that the entire pipe section can be covered with a rotation about the axis and a translation along the length of the pipe. It is also possible to add additional degrees of freedom, such as translation vertically or horizontally with respect to the camera mount so that the robot can navigate closer to or away from a particular defect on the pipe wall to get clearer images.

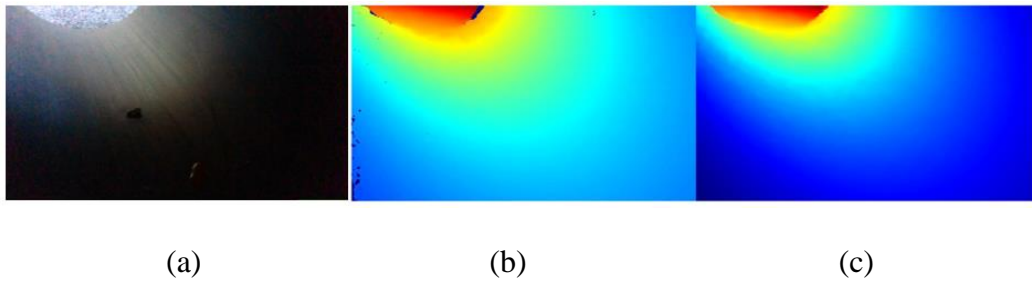


Figure 2.11: Demonstration of the angled view (a) RGB image (b) Depth map (original) (c) Depth map (filled)

Figure 2.11 has a nut that is 20mm long and has a maximum depth of approximately 10mm. The depth map does not indicate the object is present, despite it being much larger than the lowest pixel resolution provided by the depth map. Furthermore, the oblique viewing angle of the camera presents difficulties for extracting normal and curvature features from the point cloud that can indicate bumps in the surface. A simplified design is therefore proposed. A rotating single-camera system replaces the fixed multi-camera system. The rotation is about an axis parallel to the pipe's axis, so the camera is always facing normally to the pipe surface. This means that any surface defect that changes the pipe radius will be more visible, as the radial change in depth is captured directly by the depth map, which maps the z-component of the point cloud. Additionally, the proposed design guarantees that the entire pipe section can be covered with a rotation about the axis and a translation along the length of the pipe. It is also possible to add additional degrees of freedom, such as translation vertically or horizontally with respect to the camera mount so that the robot can navigate closer to or away from a particular defect on the pipe wall to get clearer images. The proposed design is shown in Figure 2.12. For the modification of the robotic platform, the following criteria were kept in mind:

- 3D-Printable: The custom parts were designed to be 3D-printable to reduce the time needed to prototype and take advantage of the lab resources.



- Co-axiality of the stereo camera: During the initial calibration procedures of the camera, we assessed the disparity of the camera for different heights, and it was found that the disparity shift was manageable when the camera was between 12-15 cm from the pipe surface. A height adjustment mechanism was designed and fabricated to fulfill this requirement for a range of pipeline sizes (16-20 inches).

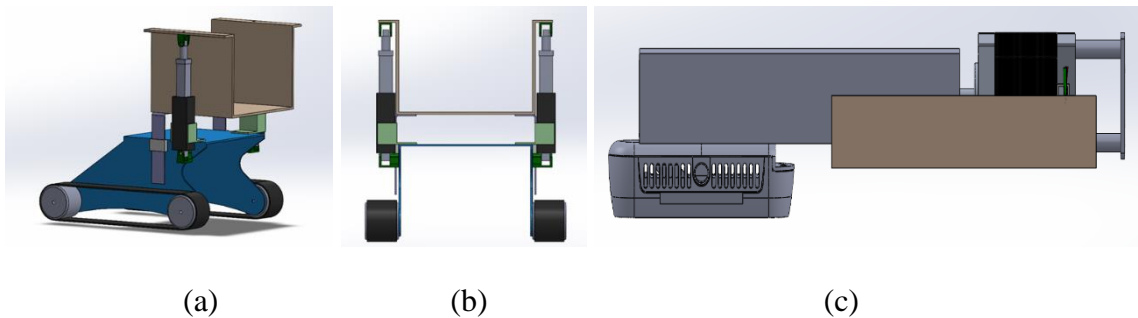


Figure 2.12: (a-b) CAD model of the robot prototype platform demonstrating the height adjustment mechanism using the PA-07 actuator. (c) Stereo camera mounted on the platform

**Precise actuation and feedback of the angular component of the pose of the stereo camera:**

A stepper motor was used to precisely actuate the stereo camera around the pipe, as shown in Figure 2.12 (c). To complement the roll angle data from the stereo camera IMU, an AS5600 encoder was attached to the back of the stepper camera. A mount had to be designed for it such that the AS5600 is placed at a distance between 3-5mm from the permanent magnet. During the trial runs outside the pipe sample, an external AC/DC adapter with a range of 12V was used. During the pipeline run, 4-rechargeable 18650 Li-Ion batteries were connected in series, generating a total of 14.4V. The power supply capacity was estimated to be 3300mAh as the connection is made in series. These batteries are held in a bracket along the frame of the robot.

**Microcontrollers:** Initially, the Arduino Uno was used to control the elements, such as actuators and stepper motors. It has 26 I/O pins in total (Digital-14, PWM/Digital-6, Analog-6). This is not sufficient for all the components of the robot. To allow for more I/O pins, we upgraded the controller to the Arduino Mega 2560. With 70 pins (Digital-39, PWM/Digital-15, Analog-16), connecting all the hardware components with options for further expansions was possible. Next, we attempted to control the robot using an older PS2 controller. It had a receiver module that was connected to the Arduino for data transmission. One of the disadvantages of having PS2 controller as the control module of the existing circuit is that it needs an individual receiver module to get

signals from the controller. The receiver module is connected to the Arduino board using jumper cables, so the receiver module might get disconnected from the Arduino. We upgraded the PS2 controller to a DualShock 4 wireless controller to mitigate this.

DualShock 4 wireless controller is found to be better than PS2 Controller-Receiver module for the following reasons:

- Unlike the PS2 Controller, the DualShock PS4 controller does not have a separate receiver module
- The optimum range of PS2 receiver module is between 8-10 meters, whereas the range of DualShock PS4 controller has a range of 15.2 to 30.5 meters.

The Arduino Mega does not have an in-built Bluetooth module, but it could be attached externally e.g., HC-05 Bluetooth module.

To further improve the integration process, we used a Raspberry Pi as the hub computer on-board the robot, with an Arduino for minimal processing assistance. The earlier version of the robot code was running exclusively on Arduino, which presented several difficulties during integration. The Raspberry Pi has the following advantages over Arduino:

- The Raspberry Pi can be connected wirelessly to the server PC using the onboard Wi-Fi module.
- Arduino Mega 2560 has a flash memory of 256kb and cannot handle multiple processes. Raspberry Pi comes with 8GB of RAM and can handle multiple processes simultaneously.
- Raspberry Pi offers more flexibility when it comes to programming languages than Arduino for rapid prototyping of software solutions.

As part of the porting process of the developed Arduino functionality to Raspberry Pi, the code was converted from C to Python.

**Motors and drivers:** Our hardware consists of two linear actuators, PA-07, for adjustment of the camera axis about the dimension of the pipes. For the robot's movement along the pipe, two geared DC motors are installed on the rear end of the robot. Both elements work in the range of 12V, and L298N motor driver individually controls them. It has two techniques to control two motors simultaneously - PWM (Pulse Width Modulation) for speed and Dual H-Bridge for rotation. It can control DC motors with a voltage range of 5-35V with a minimum logic voltage requirement of



5V. Therefore, it is powered up directly by the Raspberry Pi board. The speed of the motors (DC motors) and Rate of Stroke (Actuators) could be altered by changing the frequency of the analog PWM signal. The rotational direction (DC motors) and the Direction of Stroke (Actuators) could be changed by reversing the inputs for motor drivers.

We use the NEMA-17 stepper motor for the rotation of the camera. The NEMA-17 has a step angle of  $1.8^{\circ}$  (200 steps for a complete revolution) with holding torque of 0.44Nm. It draws a current of 1.8A at 5V. The DRV8834 motor driver controls it. This motor driver is capable of micro-stepping, speed, and direction control. It has two micro stepping pins, enabling us to attain 1/32 of the initial step resolution. The base speed is set by means of a pulse signal, and then the stepping is done as per the requirement. The motor driver is capable of driving stepper motors from 2.5V to 10.8V with a minimum logic voltage requirement of 2.5V. Therefore, it is also powered up by the Raspberry Pi board. The technical specifications and the logic table of the elements used are listed in Appendix B. The overall schematic is summarized in Figure 2.13, and the integrated prototype used in the trial run is shown in Figure 2.14.

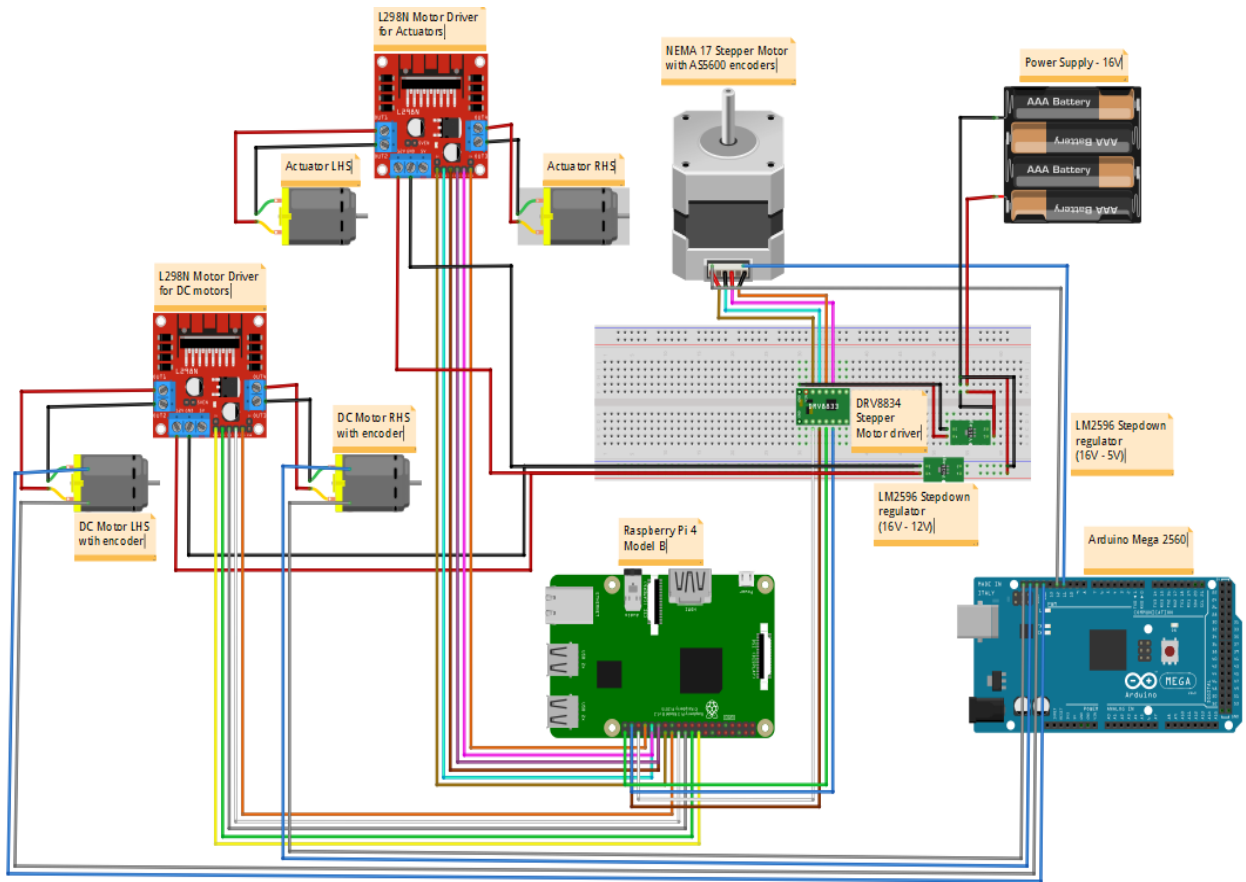


Figure 2.13: Schematic diagram of the integrated hardware electronics

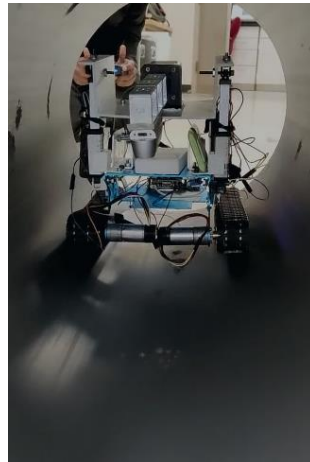


Figure 2.14: Integrated hardware prototype

## 2.4 Depth Map Generation and Evaluation of the Intel RealSense D435i Camera

### 2.4.1 Analysis of the generated depth map for depth resolution limits

For industrial inspection, the ideal outcome is to detect defects, measure them, and quantify their respective uncertainties. This is a multi-step process, with uncertainties propagating from the sensor into the detection model and finally manifesting themselves in the measurement. We start by modeling the depth sensor. Any sensor output can be expressed in the form of a mean signal,  $\bar{z}$  and a set of variances:

$$z = \bar{z} + \epsilon_s + \epsilon_t \quad 2.17$$

$$\epsilon = \epsilon_s + \epsilon_t \quad 2.18$$

where  $\epsilon_s$  and  $\epsilon_t$  are the spatial and temporal noise, respectively.

The overall uncertainty of the depth map depends on both the sensor's spatial and temporal noise. We limit the scope of this section to characterizing the noise sources at the signal output level. Explicit analytical modeling of the causes of this noise is not performed, as noise can result from several factors, such as lens distortion. In addition, the pixel resolution limitations lead to the quantization of disparity. The other noise source is the matching process and the image quality. This is dependent on the application in consideration. The matching cost function for finding corresponding points can suffer from ambiguities that lead to multiple matches with similar matching scores for in-line inspection. Filtering out the best match from a set of similar candidate matches has been done internally using the D435i stereo-matching algorithm.

A normalized metric called the subpixel Root Mean Squared Error (RMSE) is utilized to evaluate sensor performance, which is obtained from fitting a plane onto a depth map of a flat wall. The plane fit has an associated RMSE that estimates the spatial noise in the image, and the calibrated camera was found to have a temporally stabilized RMSE of 0.12%.

The subpixel RMS error is given in the form of disparity in pixels and is a function of the plane-fit projected depth  $z_p$  and the actual depth  $z_i$ , given by  $RMSE = \sqrt{z_i^2 - z_p^2}$ . Knowing the RMSE of a plane fit only provides a metric for the inherent noise level in the sensor. However, this does

not answer the crucial question of the minimum change in depth that the sensor can reliably detect, which is an important parameter that needs to be measured for optical metrology. For this, we examine the sensitivity of the depth function to infinitesimal changes in the variables it depends on:

$$dz = -\frac{z^2}{fb}dd + \frac{z}{b}db + \frac{z}{f}df \quad 2.19$$

The focal length and baseline are fixed for the D435i, so the sensitivity of the depth to small changes in disparity is proportional to the square of the depth. This means that an infinitesimal change in disparity for an object that is close to the camera would cause a smaller change in depth compared to an object that is farther away. The consequence of this relationship is that finer depth sensitivities are possible for the same perturbation in disparity for objects that are closer to the camera. This is demonstrated in Figure 2.15 (b).

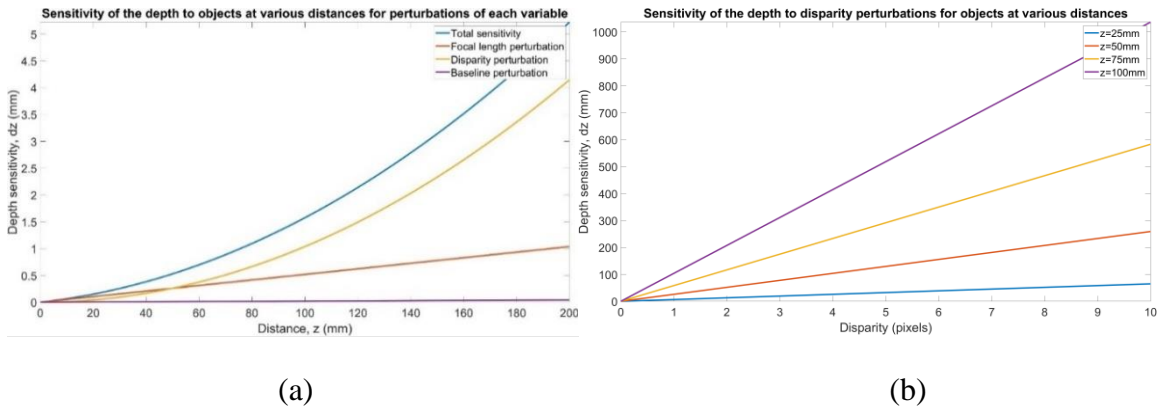


Figure 2.15: (a) Sensitivity of the depth map to regions at various distances in (mm) to perturbations in disparity, focal length, and baseline. (b) Sensitivity of the depth to disparity perturbations for objects at various distances

For the spatial resolving power of the depth camera, consider an object that is  $z$  mm away from the sensor and a camera with horizontal and vertical resolutions  $X_{res}$  and  $Y_{res}$ . Assuming a pinhole camera model, the length in pixels  $l_I$  for an object of length  $l_W$  in the world coordinates can be derived using similar triangles, as shown in Figure 2.16. To get the image length in pixels, the focal length also needs to be expressed in pixels, given by  $f_x$ . The camera field of view can be expressed as  $\theta_H$  for horizontal, and  $\theta_V$  for vertical. A similar relation can be derived between the

height of the object and the height in pixels.

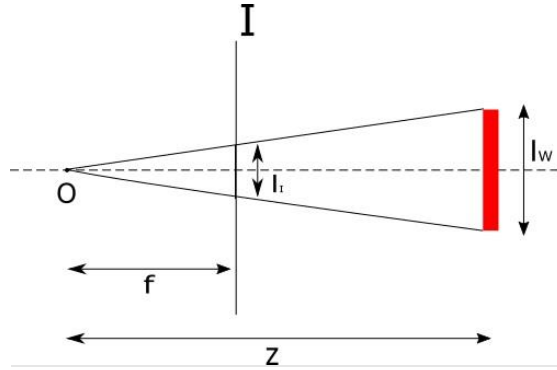


Figure 2.16: Relating the length of an object in pixels to its real-world length

The relationship between the length of an object in image coordinates and the world coordinates can be determined using similar triangles:

$$\frac{l_l}{f_x} = \frac{l_w}{z} \quad 2.20$$

$$f_x = \frac{\frac{1}{2} X_{res}}{\tan\left(\frac{\theta_H}{2}\right)} \quad 2.21$$

$$l_l = \frac{\frac{1}{2} X_{res}}{\tan\left(\frac{\theta_H}{2}\right)} \frac{l_w}{z} \quad 2.22$$

The D435i has a pixel size of  $3\mu m \times 3\mu m$ . From the equation above, the smallest object that the stereo-camera can detect can be found by computing  $\frac{2z l_l}{f}$  with a length in image space of 1 pixel. For a defect that is 200mm away, which would be the case for our experimental setup for a 16-inch diameter pipe, the length captured by 1 pixel is 0.621mm.

Additionally, an empirical evaluation of the Intel D435i depth camera to evaluate the quality of the depth map was performed. The goal was to evaluate the effect of changes in camera parameters on depth map quality, and to determine the parameter range at which the depth map most closely resembles the real-world object dimensions. Bad sensor parameters can lead to object boundary bleeding, missing values (holes) in the depth map, and noisy output. Detailed experimental results

are given in Appendix A.

#### ***2.4.2 Surface roughness profile – Signal vs Noise at high resolution***

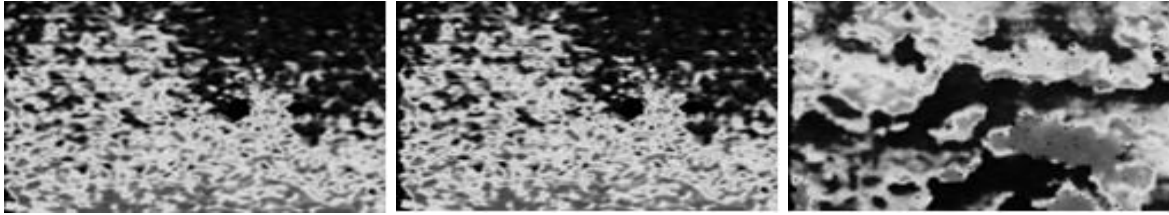
An interesting observation while capturing the images at the (1280x720) pixel resolution and depth unit  $1e^{-5}$  m was that the images captured by the depth sensor showed patterns like the ones found on the wall as shown in Figure 2.17. The dramatic contrast in the depth map occurs because the range has been limited to the minimum and maximum depth values to highlight differences in depth. The camera was placed directly perpendicular to the surface, and this was verified by ensuring that the depth did not vary substantially. To confirm whether this pattern was the surface roughness profile, or the sensor noise, a comparative analysis was performed between the rough wall and a baseline smooth wall. Three images were captured using the camera at a distance away from the surface, pointing perpendicular to it, with the same camera settings. Aggressive temporal smoothing was performed to average out temporally varying noise signals with large temporal filters. The stabilized image was analyzed using a histogram of depth values. The spatial domain is inspected using the depth histogram, and this would directly provide the distribution of depth information. The smooth image and roughness-level 1 image distributions show minimal differences between the two cases. Therefore, we can conclude that the detection of surface roughness from depth maps at the levels indicated by the middle image in Figure 2.17 will be unreliable. However, at a higher degree of roughness in the millimeter scale and above, it can be detected, as shown by the histogram in Figure 2.17 (i). The distribution on the far-right histogram has more pixels to the left of the highest mode of the distribution, indicating the peaks in the roughness profile.



(a)

(b)

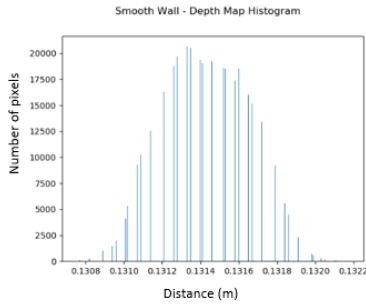
(c)



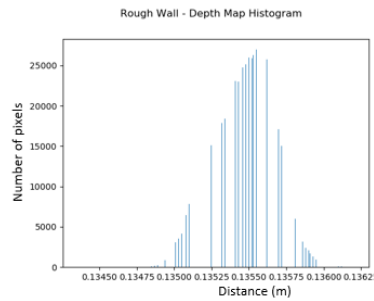
(d)

(e)

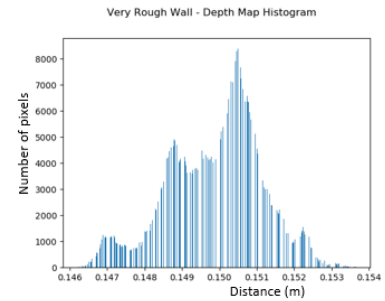
(f)



(g)



(h)



(i)

Figure 2.17 (a-c) RGB images for smooth, roughness level 1 and roughness level 2 surfaces respectively, Bottom Row (d-f) Depth maps in white to black scale for smooth, roughness level 1 and roughness level 2 surfaces respectively. (g-i) Depth histograms for smooth, roughness level 1 and roughness level 2 surfaces respectively

The resolution of the defect was shown to improve with the increase in camera resolution from 848x480 pixels to 1280x720 pixels, along with a higher sensor noise profile. To further attempt an improvement in the reconstruction, we tried to add a camera in series such that the common field of view from the two cameras had a higher IR dot density as each camera had its own IR dot projector. The D435i system is an “active-stereo” system that uses a combination of color imaging and an IR dot pattern to perform stereo matching, and the dot patterns provide additional texture to texture-less scenes, and these patterns are not visible. The color sensor detects these patterns and is substantial only in high laser power. An increase in dot pattern density was expected to

improve the matching quality by providing even more density of textures for the matching algorithm, however, no significant differences were observed.

## 2.5 Disparity Map Post-Processing Methods for Improving Depth Quality

### 2.5.1 Post-Processing of the Disparity map using Gaussian Pyramids

The disparity map obtained using the stereo block matching algorithm can contain invalid regions with no disparity values. This can compromise detection and measurement results. This section demonstrates a Gaussian Pyramid approach to post-process the generated disparity map in the previous section. A Gaussian pyramid is a hierarchical image representation scheme that downsamples the original image, followed by upsampling and interpolation. The overall method for filling missing values is shown in Figure 2.18.

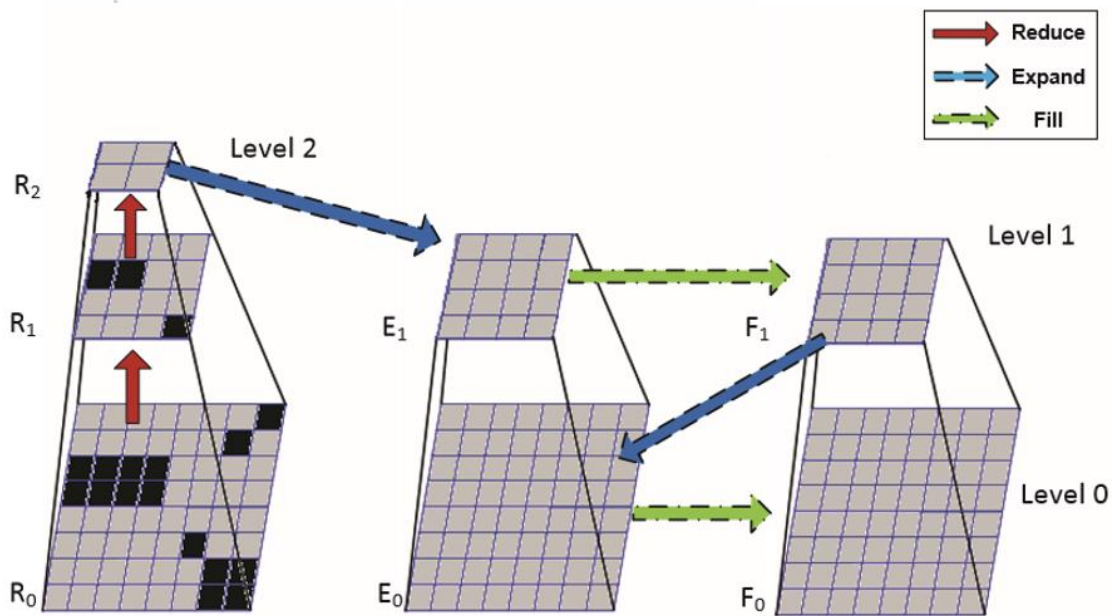


Figure 2.18: Filling missing values using a hierarchical pyramid approach (Adelson et al., 1984)

The original image, at its initial resolution, is considered level zero. The reduction output can be denoted as  $R_n$  where  $n$  is the current reduction level.  $R_0$  consists of the top-most level, and a weighting low pass filter kernel is convolved with the image and the image is subsampled by a factor of two such that:



$$I_k(i, j) = \sum_{n=-2}^2 \sum_{m=-2}^2 w(m, n) g_{k-1}(2i + m, 2j + n) \quad 2.23$$

$$R_0 \in R^{M \times N}, R_1 \in R^{\text{ceil}(\frac{M}{2}) \times \text{ceil}(\frac{N}{2})}$$

A ratio  $r = \frac{N_{\text{missing}}}{N_{\text{total}}}$  is computed for each scale, where  $N_{\text{missing}}$  is the number of missing values and  $N_{\text{total}}$  is the total number of pixels. As the resolution of the image reduces, the ratio parameter decreases. We downsample the image until  $r$  becomes zero or we get to the smallest possible representation. The lowest obtained representation has no missing values. This representation is then upsampled using the gaussian pyramid approach, where the pixels are upsampled as follows:

$$I_k(i, j) = \sum_{n=-2}^2 \sum_{m=-2}^2 w(m, n) g_{k-1}\left(\frac{i-m}{2}, \frac{j-n}{2}\right) \quad 2.24$$

$$R_k \in R^{M \times N}, R_{k-1} \in R^{2M \times 2N}$$

The weighting function that we used approximated the Gaussian shape as:

$$w = \left[ \frac{1}{4} - \frac{a}{2}, \frac{1}{4}, a, \frac{1}{4}, \frac{1}{4} - \frac{a}{2} \right], a = 0.375 \quad 2.25$$

The expansion back to the original size changes the pixel values in the original image. For replacing the invalid regions in the raw image, we have replaced the reduced representation pixels with the corresponding expansion image values in those indices if the reduced representation pixel is a missing value. The disparity map before and after post-processing is shown in Figure 2.19. There is no explicit assumption on the shape of the objects in the scene in this method.

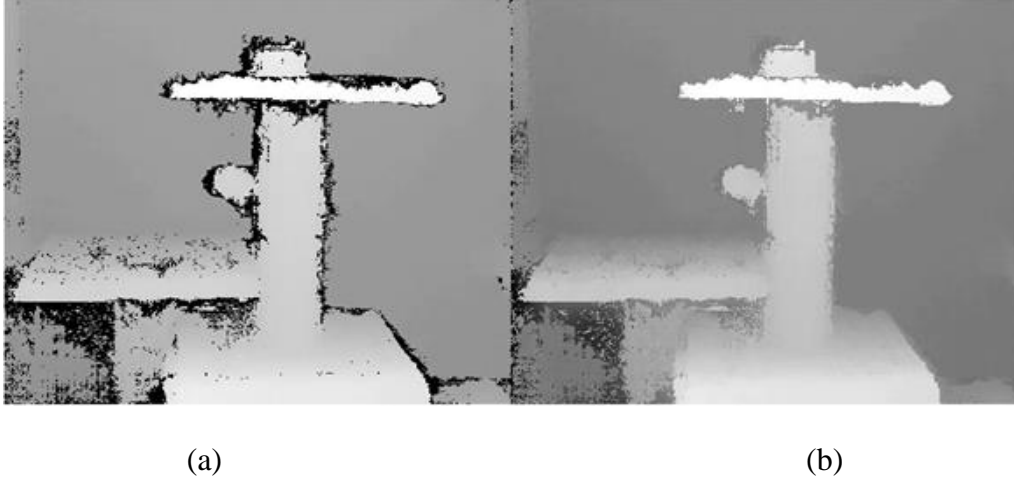


Figure 2.19: (a) Original Disparity Map (b) Disparity Map after Post-Processing with Gaussian Pyramids

### 2.5.2 Disparity Map Post Processing using a Hole Filling Algorithm

In the previous section, the disparity map obtained was post-processed using a Gaussian pyramid-based technique. The original image was progressively downsampled and then upsampled with a Gaussian filter kernel. In this section, we improve upon this approach by imposing a first-order smoothness assumption on the pixel values. The assumption makes it so that nearby pixels should have similar intensities. Levin originally published the method (Levin et al., 2004) as a proposal for an algorithm to colorize grayscale images. The assumption of having nearby pixels with similar intensities makes it so that the entire depth map gets smoothed globally by using this algorithm, and this is functionally like using Gaussian pyramids, although quantitatively different. The algorithm accomplishes this by minimizing the sum of the squared difference between the color at a particular pixel in a window and the weighted average of the neighbors within that window and applies this algorithm globally.

Let  $I(r)$  and  $I(s)$  be the intensities respectively of two neighboring pixels on the normalized intensity (greyscale) image, with  $I(r)$  being the center pixel of a sub-matrix window within the depth map. The window size is a user-defined parameter.

$$J(U) = \sum_r (I(r) - \sum_{s \in N} w_{rs} I(s))^2 \quad 2.26$$

The weighting function is large when  $Y(r)$  is closer to  $Y(s)$  and small when the two intensities are different.

$$w_{rs} = e^{-\frac{(I(r)-I(s))^2}{2\sigma_r^2}} \quad 2.27$$

Two sparse matrices  $A$  and  $G$  are constructed, where  $A$  is an affinity matrix with the pixel-wise values from the weighting function.  $G$  is a diagonal matrix. The new values for the depth map are constructed by solving the equation:

$$D' = (G - A)^{-1} * D \quad 2.28$$

This is equivalent to solving for the system  $Ax = b$ , where the system is  $(G-A)$  instead of  $(D-W)$  in the paper for obtaining normalized cut segmentation.

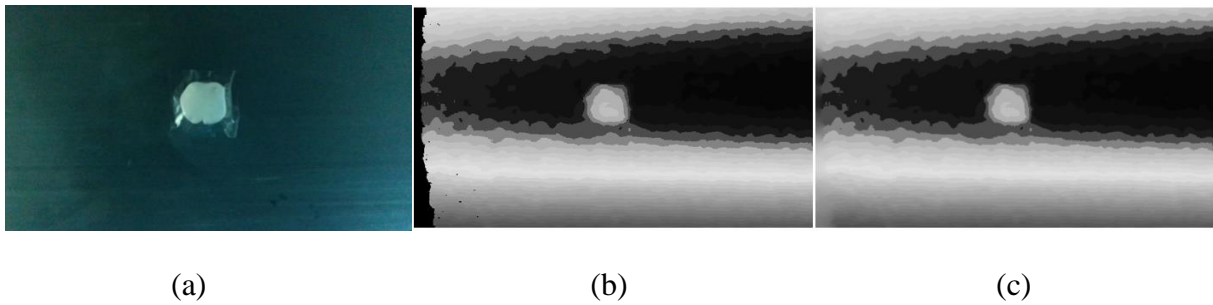


Figure 2.20: Depth map hole filling algorithm demonstration (a) RGB Image (b) Depth map with holes (c) Post-processed depth map

The resulting depth map is then multiplied by the maximum depth value in the original normalized depth map to obtain the final filled output, as shown in Figure 2.20.

## 2.6 Extremum Seeking Controller for Depth Map Optimization

There are two levels of calibration for the D435i. The first level of calibration involves tuning the camera's internal parameters and the extrinsic parameters' calibration to locate the camera in space. This has already been completed and verified within the D435i Application-Specific Integrated Circuit (ASIC) using onboard calibration algorithms (Intel RealSense, 2022). However, stereo-matching using the D435i requires additional parameter tuning. Extremum Seeking Control (ESC) is presented in this section as a solution for adaptive parameter calibration.

Extremum seeking is an online optimization technique that can be used for near real-time parameter tuning when the system model is not known. We have a cost function that measures the depth map quality, and this function is maximized (or minimized) iteratively in an online fashion. In our case, the depth map is known, however the stereo-matching parameters go into a grey-box matching algorithm that needs to be tuned. We can model the camera as a dynamical system, parameterized by a state vector  $x(t)$ . The state is updated iteratively according to a control input  $u(t)$ . Parameters are input into a depth map generator function  $h$ , which is iteratively evaluated according to a cost metric  $g$ . The goal of the method is to find  $x^*$  that minimizes this cost function. The entire loop is shown in and is summarized in this section.

$$\dot{x} = f(x(t), u(t)) \quad 2.29$$

$$y(t) = g(h(x(t))) \quad 2.30$$

$$x \in R^n$$

$$J: R^{W \times H} \rightarrow R$$

$$h: R^n \rightarrow R^{W \times H}$$

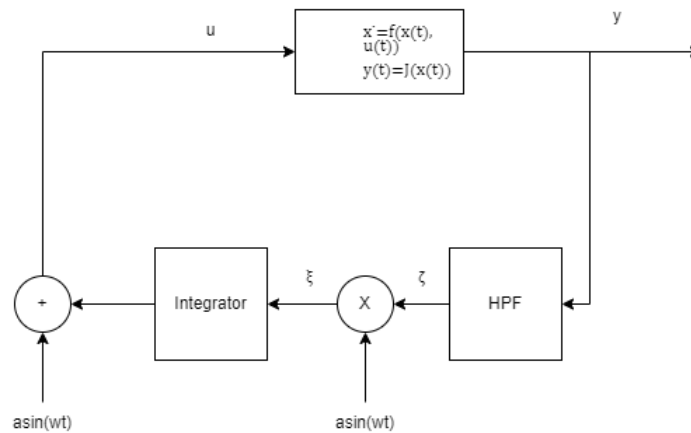


Figure 2.21: Extremum seeking control loop

The cost function for our demonstration relies on generating a continuous depth map with valid disparity pixels. With a changing environment that is unknown apriori, this would mean that the optimal parameter settings can keep changing, leading to discontinuous depth maps. We directly measure this using the following function:

$$h(x) = -\log\left(\frac{n_f}{WxH}\right) \quad 2.31$$

The number of filled pixels  $n_f$  in the frame or region of interest and the width and height of the region of interest, frame are the inputs to this function. The extremum-seeking control approach perturbs the parameters using a sinusoid to generate an AC signal of the form:

$$x = x_k + a\sin(\omega t) \quad 2.32$$

The resulting cost function evaluation is also perturbed, and this perturbed function is passed through a high pass filter (HPF) to remove the DC component of the signal. The extremum-seeking controller is seen to be an online form of hill climbing or gradient descent, as the Taylor series expansion of the cost function shows:

$$g(x) = g(x_k) + \nabla g_k(x - x_k) \quad 2.33$$

$$x = x_k + a\sin(\omega t) \quad 2.34$$

$$g(x) = g(x_k) + a\nabla g_k \sin(\omega t) \quad 2.35$$

The high pass filtered output gets attenuated with a phase shift and the result is:

$$\zeta = HPF(g(x)) \quad 2.36$$

$$\zeta = aA\nabla g_k \sin(\omega t + \phi) \quad 2.37$$

$$\frac{1}{\sqrt{2}} < A < 1$$

$$0 < \phi < 45^\circ$$

The high pass filter provides an estimate of the gradient with a sinusoidal component. To make this sinusoidal time-varying component strictly positive, we multiply with a sinusoid:

$$\xi = a\sin(\omega t) \zeta \quad 2.38$$

$$\xi = a^2 A \sin(\omega t) \sin(\omega t + \phi) \nabla g_k \quad 2.39$$

$$\xi = S(t)\nabla g_k \quad 2.40$$

Each parameter is updated as  $x_k = x_{k-1} - \alpha \nabla g_k$  according to gradient descent can then be derived using an integrator. The Z-Transform can illustrate how the transfer function of an integrator directly provides the next iterate. The Z-Transform is given by:

$$Z\{x(k)\} = \sum_{k=0}^K x(k)z^{-k} \quad 2.41$$

$$Z\{x(t)\} = X(z) \quad 2.42$$

The transfer function in z-domain of an integrator is given by:

$$I(z) = \frac{k}{1 - z^{-1}} \quad 2.43$$

The update rule when transformed into z-domain and rewritten reproduces the transfer function:

$$x_t = x_{t-1} + \alpha \nabla g \quad 2.44$$

Since  $Z(x(k-1)) = z^{-1}X(z)$ :

$$X(z)(1 - z^{-1}) = \alpha \nabla g \quad 2.45$$

The integrator therefore performs parameter updates, modified by a gain proportional to learning rate parameter  $\alpha$  in gradient descent.

For the demonstrations shown in this report, a perturbation sinusoid proposed by Scheinker et al (Scheinker et al., 2021) is used:

$$u_k = \sqrt{a_{ES}\omega_i} \cos(\omega_i k + k_{ES}J(x, k)) \quad 2.46$$

$$a_{ES} = \omega_i D^2 \quad 2.47$$

$$\omega = [\omega_1, \omega_2, \dots, \omega_n]$$

For each parameter in  $x$ , there is a corresponding dithering frequency. Another hyperparameter is

oscillation amplitude  $D$ . The final hyperparameter is the gain  $k_{ES}$ . The oscillation amplitude is varied until the parameter variation is high enough to demonstrate a corresponding change in the objective function value, and the gain is then tuned to provide a local optimum. The demonstrations use 3 critical parameters: Exposure, gain and IR dot projector power, that significantly influence the continuity of the depth map. Given that the functional relationship between these parameters and the cost function is a priori unknown, and there is no model-based learning involved, the knowledge of “over” and “under” exposed is not known to the controller. In all the forthcoming demonstrations,  $p1$  is the exposure,  $p2$  is the gain and  $p3$  is the IR laser power.

To test how the controller performs, we first set up a baseline case of a bright environment with the default settings of the camera, and the evolution of the cost function and parameters is seen in Figure 2.22. The depth map reaches a local optimum in a few iterations and the parameter oscillations are damped according to the hyperparameters specified earlier and a prespecified decay rate that is applied to the oscillation amplitude in each iterative step.

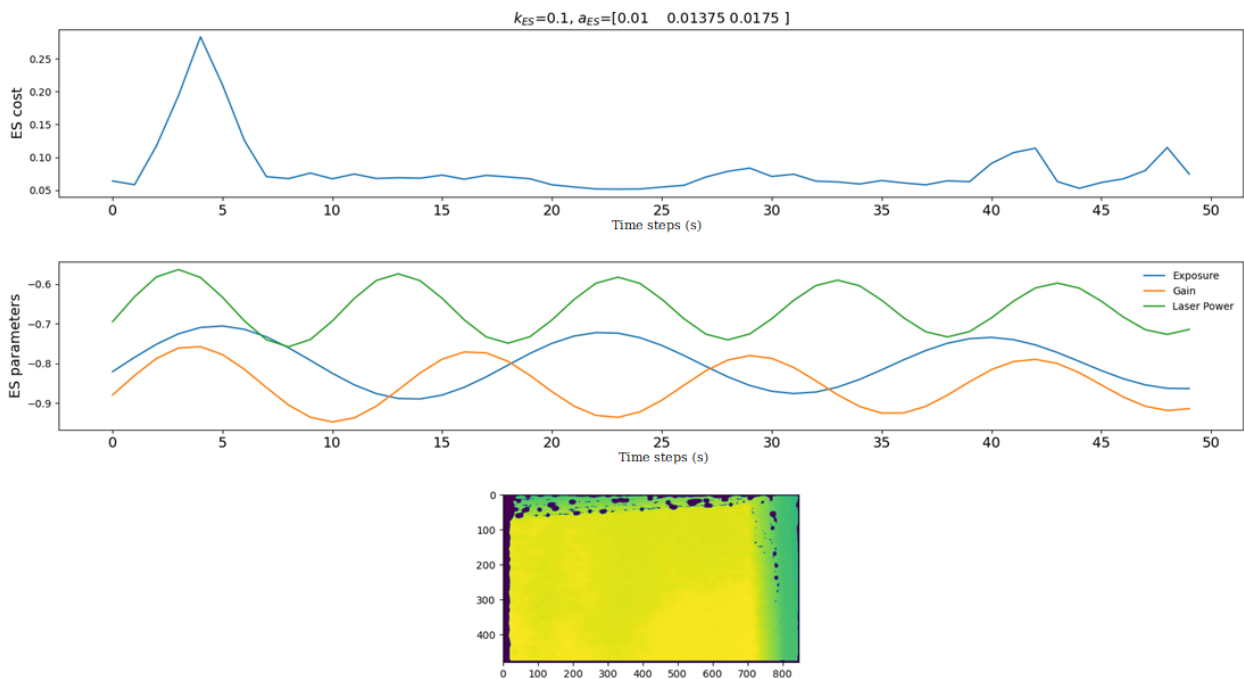


Figure 2.22: Demonstration 1: Baseline test case in a well-lit scene with the factory D435i parameter settings

Next, we darken the environment and start with bad initial conditions of the parameters, with the exposure, gain and IR power all being set to their lowest possible values, leading to a highly

underexposed image pair, and the resulting stereo image had many discontinuities. Finally, however, extremum seeking control converged to a local optimum as shown in Figure 2.23.

The third experiment takes a quickly varying environment by way of the surrounding brightness of the scene. This situation could be like the one that a robot can encounter inside a pipe with sections that may have varying lighting conditions. We first start similar to the second experiment with a dark environment with bad initial conditions, for 5 seconds. We then introduce a light source to the environment using a torch for 5 seconds. Then we remove the light source and measure the change in the cost function. Given the chosen hyperparameters, the cost function signal showed oscillations initially until the light source was introduced. With the introduction of the light source, there were smaller perturbations to the cost function, followed by a stabilization of the cost shown in Figure 2.24.

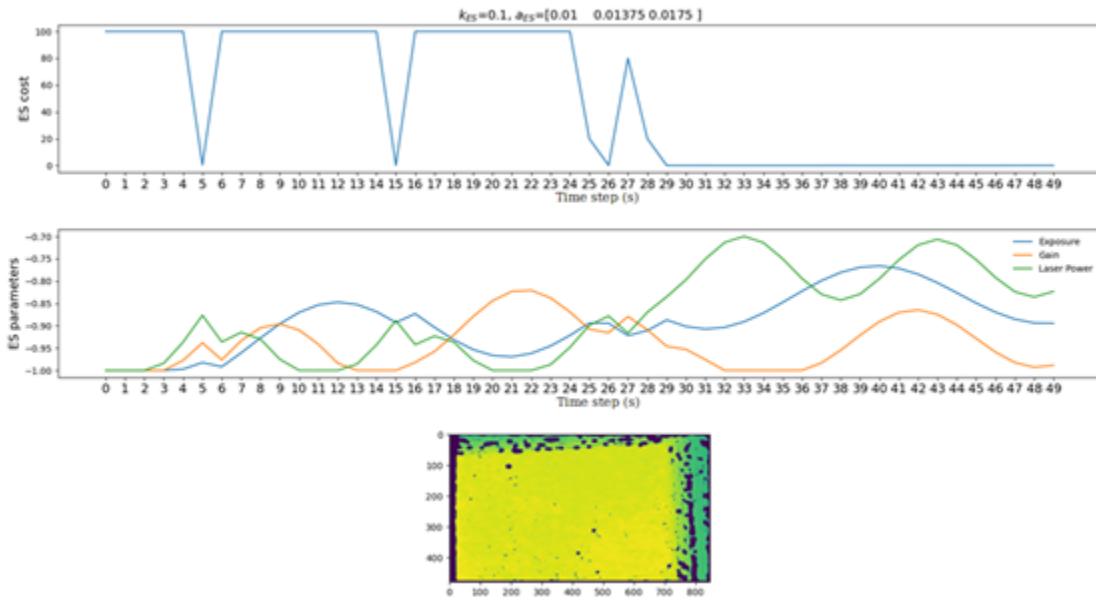


Figure 2.23: Demonstration 2: Poor lighting conditions with bad parameter settings: Low exposure, gain and IR laser power



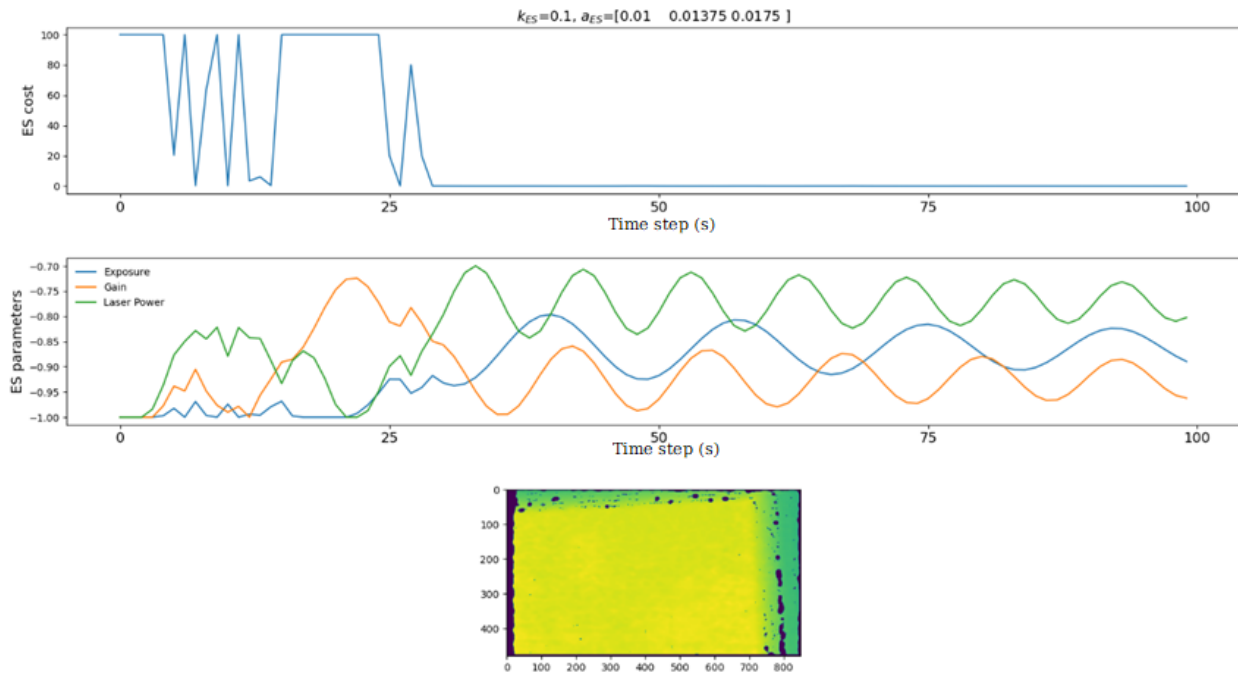


Figure 2.24: Demonstration 3: Test case with varying external illumination conditions starting with a dark environment with poor initial condition setting

### 3 DATA-DRIVEN TECHNIQUES FOR DEFECT DETECTION

#### 3.1 Fully Supervised RGB-D Semantic Segmentation with Uncertainty

##### 3.1.1 Feature Fusion using Fully Convolutional Networks

This section describes a fully supervised machine learning framework for semantic scene segmentation. Convolutional Neural Networks (CNN) yields a hierarchy of learnable features obtained via training on a dataset. The Fully Convolutional Network (FCN) (Shelhamer et al., 2017) is an end-to-end technique to train a model for pixel-wise prediction of categories, segmenting the scene into the various known object categories. Convolutional networks have been used for object detection and image segmentation. Patch-based, region-of-interest based methods contain multiple steps such as region proposal search, as in the Mask R-CNN (K. He et al., 2017), and Faster R-CNN (Ren et al., 2017) techniques, or it contains a fixed set of regions to directly output a set of bounding boxes in an end-to-end fashion with requirements on anchor boxes, as in the You Only Look Once (YOLO) network architecture (J. Li et al., 2018; Y. Li et al., 2019; Redmon & Farhadi, 2018).

FCN takes inspiration from the classification convolutional network architecture and removes the final fully connected layer, and replaces it with an expanding, upsampling architecture. Fully connected layers can be interpreted as convolutions that cover the entire output from the previous final convolution layer. Projecting the condensed representation to a higher feature dimension can be accomplished using a deconvolution operation. This can be done in stages, mirroring the convolution layers. The transposed convolution is not an inversion of the convolution but takes a lower dimensional feature and applies a convolutional operation with padding on the lower dimensional representation to transform it into the desired higher dimensional feature. The usefulness of the transposed convolution is because we have learnable weights on the filter kernels, and the eventual upsampled original dimension representation can be replaced with the label matrix, consisting of pixel-wise labels for the object categories. Fusion at the convolutional layer level is accomplished in FCNs by modifying the network architecture to combine feature representations along the network. In our experiments, our FCN uses the VGG-16 backbone network. The VGG-16 network consists of 5 convolutional blocks, where each block ends with a max-pooling layer. The outputs of the max pooling layer and the corresponding similar-sized

output from the deconvolution layer are combined, as shown in Figure 3.1. Incorporation of information from both the downsampling and upsampling layers enables pixel-wise predictions as the downsampling layers provide cues on the presence of object categories, as seen in regular classification networks, which leverage this information using fully connected layers, and the upsampling layers enable localization of the objects of interest by expanding the condensed representation.

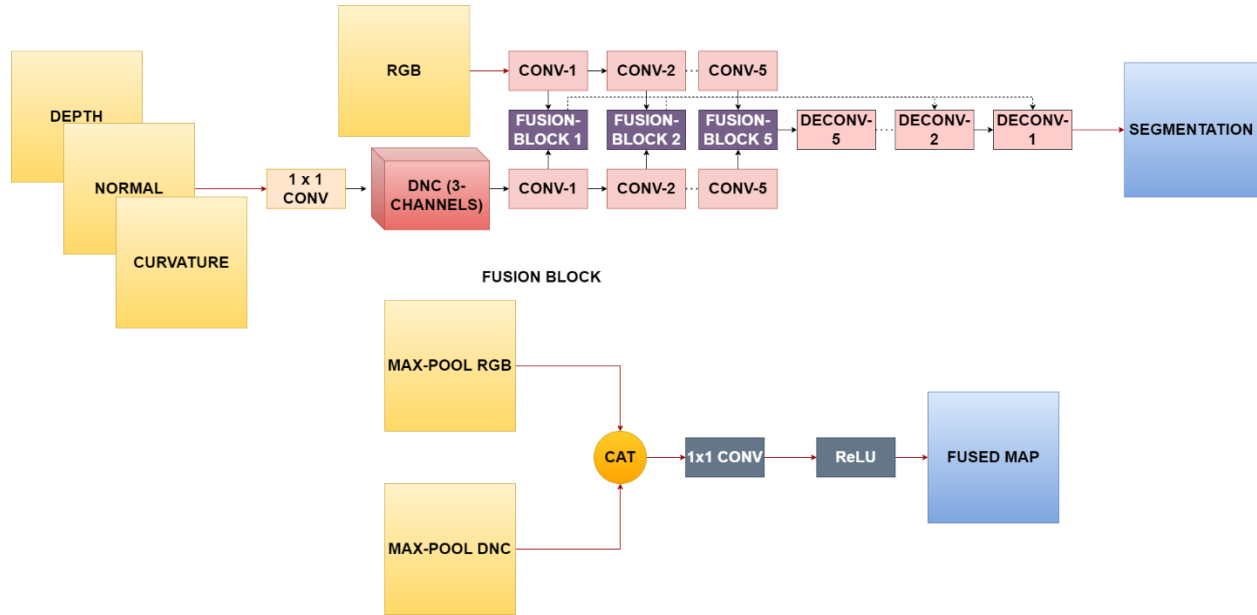


Figure 3.1: Overview of the proposed network architecture

The purpose of using multiple modalities in prediction is to provide complementary information to aid model prediction. The RGB data stream provides texture and plane-projected morphological information without depth perception for industrial inspection. The depth stream provides a more fine-grained morphological representation of the scene. We also propose a method to integrate data streams derived from the depth stream that would be relevant to the pipeline defect detection problem. Data fusion rules have been extensively studied in the field of image processing. The complexity of the fusion methods has increased to produce incremental improvements in quantitative performance metrics when evaluated on large datasets of natural images. However, there are missing concrete explanations as to why these modifications work and whether the improvements are predictably higher across different datasets. The empiricism of deep learning architecture modifications led us to approach this part of the network by using a simpler fusion rule architecture derived from classical decision-theoretic and data fusion principles. We contend

that there are two principled ways of approaching data fusion in the context of deep learning: (i) Neural Architecture Search (ii) Rules-based fusion. The first approach seeks a network architecture that can robustly fuse features from different data sources to satisfy a meta-objective of predicting well across multiple datasets. The second approach, while not directly geared towards improving detection performance metrics, would instead target providing interpretability for how the individual modality information is being utilized for the final prediction and provide a way for the end-user to make operational choices for sensor integration. We contend that industrial inspection in its current state could benefit from a hybrid approach of data-driven feature extraction, engineered data pre-processing, and feature fusion rules.

Fusion can be accomplished in multiple ways:

1. Data Fusion: In this case, the fusion occurs at the level of the raw data.
2. Feature Fusion: Features extracted from the data are fused together in intelligent ways to produce a new, combined feature representation.
3. Decision Fusion: Several classifiers, each operating on one type of data, is used to fuse the decisions made by each classifier to produce a combined evaluation using all the sources available.

We use RGB-D data for performing hierarchical feature fusion and then integrate new data sources derived from the depth map that was hypothesized to be relevant for pipeline data.

### **Fusion block architecture 1:**

The first fusion architecture was built upon the VGG convolutional network backbone with pre-trained weights. We remove the fully connected component and replace it with the new deconvolutional layer that is to be trained with the NYU dataset. The fusion operation is:

$$a_{ES} = \omega_i D^2 \tag{3.1}$$

$$F = M_{RGB} + M_D \tag{3.2}$$

This is a simple element-wise addition of features from the convolutional layers. These outputs are then combined with the corresponding deconvolutional layer

### **Fusion block architecture 2:**

The second fusion architecture involved a change in the fusion block. We use a non-linear weighted combination of the max pooling layer outputs from each convolutional block as follows:

$$F = f(M_{RGB}, M_D) \quad 3.3$$

$$f(M_{RGB}, M_D) = ReLU((M_{RGB}, M_D) * K) \quad 3.4$$

where  $K$  is a  $1 \times 1$  convolutional filter set.

The input features from the color and depth streams are first concatenated to create an initial joint representation. To implement non-linear weighting, we need to multiply the depth and RGB components by a scale factor. Concatenation of the RGB and depth channels doubles the channel dimension length. The convolutional operator acts on the two sections of the concatenated output to reduce the channel dimension back to 64. The non-linear weighted combination fusion layer produces a set of  $C_O = \frac{C_I}{2}$  weighted combinations of input features that have  $C_I$  channels. Let the input features have a dimension of  $(C_I \times M \times N)$ . The  $1 \times 1$  convolution layer has a shape  $(C_O \times C_I \times 1 \times 1)$ . Each kernel  $K_i$  produces a distribution of weights, when convolved with the input features produces a weighted sum  $\sum_j K_{ij} \cdot I_j = F_i$ . The final fused output of the  $1 \times 1$  convolution is a concatenation of each of these individual weighted combinations of the original input. From the point of view of interpretability, however, this is different from a simple weighted combination. Instead, we have an output map that consists of a set of weighted combinations, densely weighted by a combination of the input features:

$$K_1 = [K_{11}, K_{12}, \dots, K_{1C_I}], \quad \dots, \quad K_{C_O} = [K_{C_O1}, K_{C_O2}, \dots, K_{C_O C_I}] \quad 3.5$$

While the weights in this model are not resolved to the positive domain the kernels provide a way to interpret the relative contribution of each input feature for each channel in the output feature.

### **Depth-Normal-Curvature Representation for Semantic Segmentation**

The problem of transforming the depth map of a cylinder to an equivalent flat plate with highlighted defect areas was studied by (Alzuhiri et al., 2021). (Alzuhiri et al., 2021) used a least-squares fit of an ellipsoidal surface to accomplish this task. But this approach is computationally intensive as it requires parametric modeling of pipeline geometry and is sensitive to the location

of the camera. Moreover, it was primarily used for the reconstruction of gas pipeline geometry.

We propose a representational scheme, the DNC (Depth-Normal-Curvature) representation. This is a 6-channel representation compressed into 3 channels using 1x1 convolutions, as shown in Figure 3.2. The surface normal is a 3-channel image, one for each component of the normal vector. The curvature consists of the mean and Gaussian curvatures, which can be computed from the principal curvatures of a point on a surface. The depth information obtained from the stereo camera contains the curvature and normal information implicitly. Therefore, detailed descriptions of surface characteristics can be derived from depth maps.

Most of the ILI tools are used to inspect pipelines and run through the cylindrical sections in the pipelines and the camera in our experiments is placed approximately along the axis of the pipe about which it rotates. Therefore, the depth data is not captured obliquely but approximately normal to the surface of the pipe. This makes it possible to take advantage of the symmetry in the geometry of the view being captured. As the camera rotates about the pipe axis, the curvature of the cylinder remains constant. It is important to note the sources that cause a deviation from this ideal situation. They are as follows:

- the noise inherent in the depth signal,
- the noise induced by the algorithm used to compute curvature,
- the signals from the defect

Similarly, for general detection and scene understanding problems, the surface normal provides additional geometric cues like curvature. The normal serves to provide a "bump map", similar in vain to the ones used in computer graphics, to simulate wrinkles and bumps in textures. The differential geometry of a surface provides a discretized formulation of normals from first derivative information and curvatures from second derivative information if these derivatives exist throughout the domain.

The surface normals and curvatures are derived from the depth map  $Z = h(X, Y)$  which is a Monge patch  $\zeta : U \rightarrow R^3$  with the form:

$$\zeta(X, Y) = (X, Y, h(X, Y)), (X, Y) \in U \quad 3.6$$

The normals  $\eta$  can be calculated using the first derivative information, and the curvature can be

calculated using the second derivative information. The Gaussian and mean curvatures  $K$  and  $H$  of the surface are then computed using the standard equations describing the fundamental coefficients of a surface.

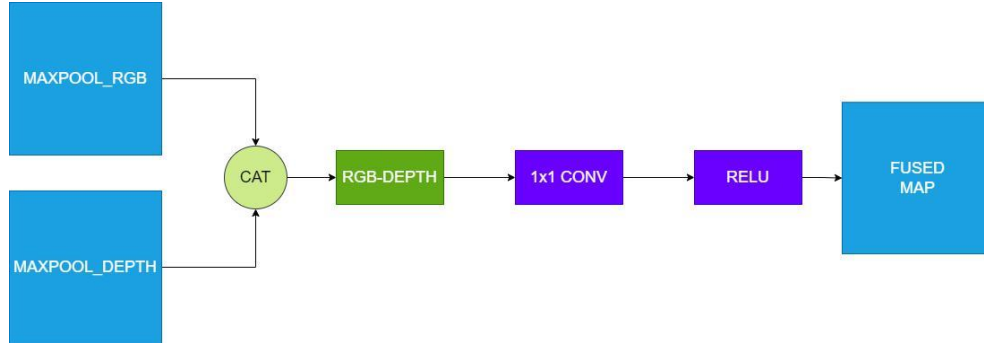


Figure 3.2: RGB-D Fusion Module

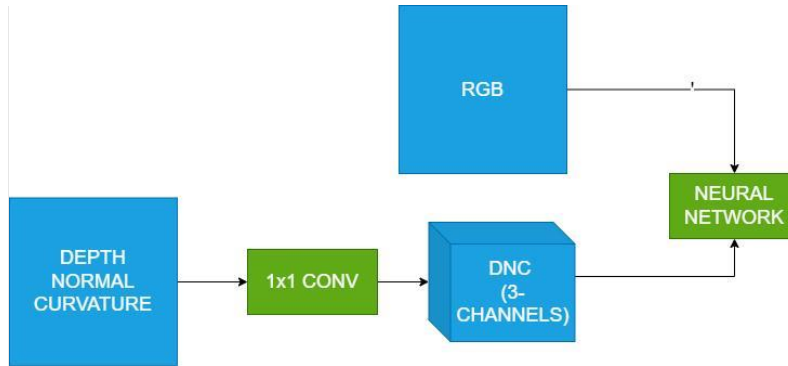


Figure 3.3: RGB-DNC Data Pre-processing Module.

The derivatives are computed along the  $x_1$  and  $x_2$  directions as follows:

$$D_i \zeta = \begin{bmatrix} \frac{\partial X_j}{\partial x_i} & \frac{\partial Y_j}{\partial x_i} & \frac{\partial Z_j}{\partial x_i} \end{bmatrix} \quad 3.7$$

$$D_{ii} \zeta = \begin{bmatrix} \frac{\partial^2 X_j}{\partial x_i^2} & \frac{\partial^2 Y_j}{\partial x_i^2} & \frac{\partial^2 Z_j}{\partial x_i^2} \end{bmatrix} \quad 3.8$$

$$D_{ij} \zeta = \begin{bmatrix} \frac{\partial^2 X_j}{\partial x_i x_j} & \frac{\partial^2 Y_j}{\partial x_i x_j} & \frac{\partial^2 Z_j}{\partial x_i x_j} \end{bmatrix} \quad 3.9$$

$$\eta(X, Y) = \frac{D_x \zeta X D_y \zeta}{|D_x \zeta X D_y \zeta|} \quad 3.10$$

The curvature information is derived from the fundamental coefficients of the surface by first computing the inner product between the first derivatives for the first 3 components E, F, and G. We use the inner product between the second derivative components of the surface and the normal map for computing L, M and N:

$$E = D_{x_1} \zeta \cdot D_{x_1} \zeta \quad 3.11$$

$$F = D_{x_1} \zeta \cdot D_{x_2} \zeta \quad 3.12$$

$$G = D_{x_2} \zeta \cdot D_{x_2} \zeta \quad 3.13$$

$$L = D_{x_1 x_1} \zeta \cdot \eta(X, Y) \quad 3.14$$

$$M = D_{x_1 y_2} \zeta \cdot \eta(X, Y) \quad 3.15$$

$$N = D_{x_2 x_2} \zeta \cdot \eta(X, Y) \quad 3.16$$

$$K = \frac{LN - M^2}{EG - F^2} \quad 3.17$$

$$H = \frac{LG + NE - 2FM}{2(EG - F^2)} \quad 3.18$$

We utilize the equations outlined above to obtain representations of simple geometric shapes. This curvature characterization is "visible-invariant", a feature that is invariant to viewpoint transformations that preserve the visibility of the region in question. A discrete smooth flat plate was created to demonstrate this method. We use a (1000x1000) grid with  $z = 1$  as shown in Figure 3.4. The normals were calculated to be (0,0,1) across the entire grid, which is accurate. Both mean and Gaussian curvatures are zero, which tells us that the computation is correct for the flat plate. Next, a cylinder section was created using a (1000x1000) grid with a 180-degree field of view as shown in Figure 3.4. The mean curvature was calculated to be -1, and the Gaussian curvature was calculated to be 0. The cylinder created now can be considered as bending the flat plate about the



longitudinal axis. While the surface shape has changed, the length of the arc connecting two points in that surface has not. The Gaussian curvature formulation outlined above is invariant to such transformations and remains unchanged, but the mean curvature is not invariant to the change in the embedding of the surface within the 3D space. The next case that is considered is the rotated cylinder. This case demonstrates a scenario that can occur during data collection from the pipe sample. Due to the misalignment of the camera, it is possible that the depth map can deviate from being approximately normal to the surface. The curvatures demonstrate that they are approximately invariant to such rotations.

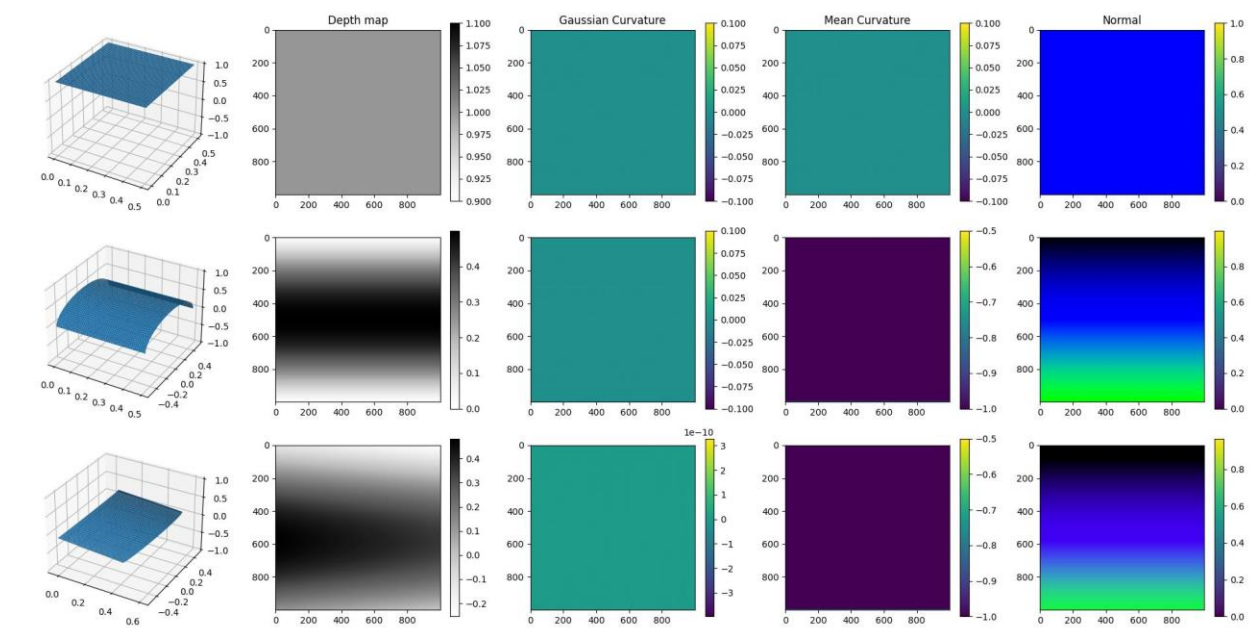


Figure 3.4: Demonstration of the curvature and normal maps for idealized surface formulations

Figure 3.5 shows a demonstrative example comparing the depth data with the data derived from it. This data is passed into the neural network using the RGB-DNC data processing module, as shown in Figure 3.3. The Gaussian curvature is the only curvature used in our experiments as it serves to indicate changes in surface shape at each discrete point of the surface. The curvatures shown in the pitted surface indicate higher positive values than the surrounding area. Mean curvature changes in sign depending on whether the surface is convex or concave from the camera viewpoint, but Gaussian curvature is invariant to this change in direction. From the above demonstrations, we observe that a Gaussian curvature feature descriptor without issues such as signal noise is sufficient for detecting pitting defects embedded in a smooth cylinder.

Improvements to this formulation are directions for future research.

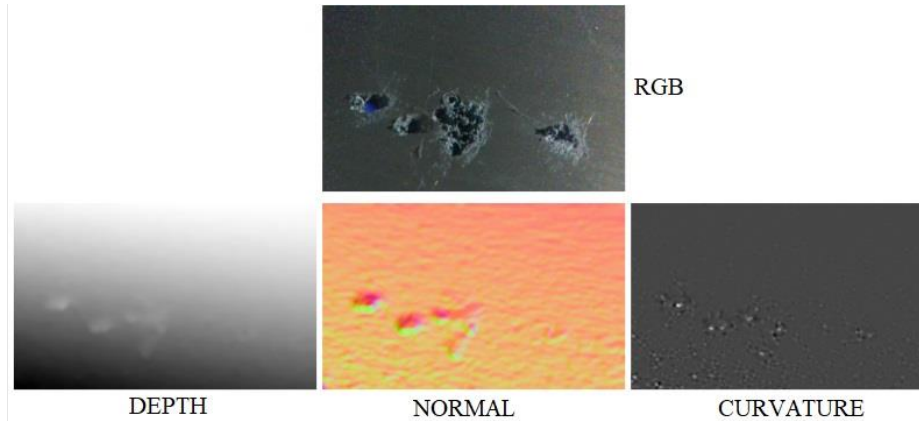


Figure 3.5: DNC data derived from the point cloud representation

### 3.1.2 *Epistemic and Aleatoric Uncertainty Quantification using MC-Dropout*

Structural prognostics benefit immensely from modeling predictive uncertainty at multiple stages of analysis. Unfortunately, point estimates provided by deep neural networks may lead to overconfident predictions that can be misleading. Ideally, we seek expressive models, have a unifying formalism to help extend the model's capabilities for downstream tasks and have a way to explain its predictions to the end user. This is achieved by taking a Bayesian view of the problem. The Bayesian framework lends itself very well to modeling a broad range of problems, and it is easier to work with at the level of prognostics and risk assessment. In this section, we describe the Bayesian view of deep learning to evaluate and contextualize the uncertainties for defect detection. We expand upon some of the challenges in the disambiguation of uncertainties in the defect detection problem. We analyze a benchmark problem to verify the capabilities of this approach to uncertainties and then justify the use of this approach to modeling uncertainties.

Consider the general setup for Bayesian inference: A prior distribution is defined over the space of functions  $p(f)$  where  $f$  is a sample from the distribution of functions that could have generated the data  $D = (X, Y)$ . The likelihood function is  $p(Y | f, X)$ . The posterior can be written using the Bayes rule from the prior  $p(f)$  and the likelihood function. For an unseen data point  $x^*$ , the predictive distribution can be computed by integrating over all possible functions  $f^*$  to yield the conditional probability for prediction  $y^*$ :

$$p(y^* | x^*, X, Y) = \int p(y^* | f^*)p(f^* | x^*, X, Y)df^* \quad 3.19$$

The problem with the above equation arises from the inability of neural networks to properly estimate uncertainty as part of the prediction. Calculating a measure of uncertainty is more challenging for deep networks than approaches such as Bayesian linear models and Gaussian processes. This is because larger parameter spaces in deep networks produce posteriors that are computationally intractable to integrate and do not have pre-determined conjugate forms that make direct evaluations of updated posterior distributions possible. Due to these reasons, deep learning does not lend itself to analytical Bayesian inference. Instead, it requires approximate inference techniques, which have seen a lot of development in the computational statistics literature. This paper aims to use a computationally cheap approximation to Bayesian inference in neural networks, the dropout (Srivastava et al., 2014; Gal and Ghahramani, 2016). This approach relies on producing Monte-Carlo draws from a neural network regularized by dropout to calculate a measure of dispersion in the prediction. Dropout is classically used to modulate model complexity by deactivating some parameters at random. In a Bayesian Deep Learning context, Gal et.al [16] make use of dropout as a model averaging tool during inference by sampling prediction outputs from various instantiations of the neural network. Their analysis shows that the dropout-regularized model is learning parameters like that learned by the approximate posterior distribution of a Deep Gaussian Process.

Evaluating the uncertainty is done by averaging  $N$  forward passes during inference. This is equivalent to drawing  $N$  samples from the set of parameters defined in the model posterior and evaluating a function using each of those samples. The variance of these samples computes a form of uncertainty. Given that the desired output variable is drawn from an arbitrary distribution, the sample mean approaches the population means at large sample sizes, with a spread that is dictated by the form of the function resulting from each dropout sampling iteration. The question here is whether the uncertainty is purely derived from the uncertainty about the weights. The analysis of the variance in (Brach et al., 2020) provides some insight into this problem. If  $E_i$  and  $V_i$  are the expected value and variance, respectively of the  $i^{th}$  node at the input and dropout is applied, the expected value and variance at the output after dropout is given by:

$$E_i^D = E_i(1 - p) \quad 3.20$$

$$V_i^D = V_i p(1 - p) + V_i (1 - p)^2 + E_i^2 p(1 - p) \quad 3.21$$

The variance equation shows that there is a propagation of uncertainty across layers. Since the input sample during inference has no uncertainty, the only contribution to the variance in MC Dropout is from the dropout layer. Therefore, this component of uncertainty is driven by the weight uncertainty.

The uncertainty in the model needs to be distinguished from the uncertainty in the input data. It is possible to illustrate this distinction with the Bayesian Linear Regression model, where we decompose the uncertainty into multiple parts, as shown in Figure 3.6. As always, there is an uncertainty associated with the weights, which is expressed as part of the prior distribution. There is also inherent uncertainty in the training data distribution. This is expressed using the likelihood function of the output given the data and the parameters. The component of uncertainty that depends on the noise in the training data distribution is called the aleatoric uncertainty. Classical Bayesian linear regression formulations as shown by (Bishop, 2014) [18] assume a constant, known value for this term. However, this assumption does not hold true for real-world data, as some training samples can have higher variance than others - an example of this in the pipeline problem is the greater predictive uncertainty at places where some occlusions and reflections cause a degradation in the quality of the depth image. Occlusions and reflections can cause invalid depth values, which in turn affects detection performance.

The proposed model learns a component of the variance as a function of the data, referred to as the heteroscedastic aleatoric uncertainty - the component of uncertainty that cannot be explained away with more data. The epistemic component of the uncertainty is calculated using Monte Carlo (MC) dropout during inference. The robotic system that has been developed can produce data that shows low variability in artifacts such as reflections and has high consistency in producing image maps without occlusions, view changes, etc. These sources of variability can be considered “input-dependent” and are captured by the heteroscedastic aleatoric uncertainty term. On the other hand, the industrial imaging domain does not always have many samples to train a model on, which requires epistemic uncertainty to reflect this. The approach used by Kendall and Gal (Kendall & Gal, 2017) distinguishes epistemic and aleatoric uncertainty by deriving the loss function from the

likelihood formulation. We demonstrate this approach first on a toy classification problem.

### **Modeling – Regression uncertainty quantification:**

We verify the required capabilities of the uncertainty modeling approach using a standard Bayesian Linear Regression model as a benchmark with homoscedastic noise, as seen in Figure 3.7. The network outputs 2 values – the prediction and its associated log-variance. The epistemic uncertainty is captured by computing the variance of the MC Dropout draws. The aleatoric uncertainty is captured by exponentiation and transforming the log variance back into the original space. The model training assumes that the data generating distribution prior is a Gaussian. The negative log-likelihood of this distribution produces the required regression loss, modified for uncertainty quantification:

$$L_{regression} = \frac{1}{N} \sum \frac{1}{2\sigma(x_i^2)} \|y_i - x_i\|^2 + 0.5 \ln \sigma(x_i)^2 \quad 3.22$$

This is a modification of the MSE (Mean Squared Error) loss with an additional variance term. For numerical stability, the  $\log \sigma(x_i)^2$  term is modeled using the aleatoric output directly, and the variance in the first term is exponentiated to produce:

$$L = \frac{1}{N} \sum \frac{1}{2} \|y_i - f(x_i)\|^2 + \frac{1}{2} s_i \quad 3.23$$

The exponential in the first term ensures a positive value for the variance, and the second term serves to balance the first as this term gives a larger contribution to the loss when the variance is high. Kendall and Gal (Kendall & Gal, 2017) has derived this model from the likelihood loss, so it is a “consequence of the probabilistic interpretation of the model.”

We analyze how well the model separates the sources of uncertainty. The baseline is a Bayesian linear model to fit a line, with data generated from a Gaussian. In this case, we assume a known, constant variance across the sampling domain to simplify the analysis. The objective of this evaluation is to compare the epistemic uncertainties and the ability of the model to capture the inherent noise in the data using the predictive aleatoric variance and analyze the reduction in uncertainty for samples within the training domain compared to those far away from it. Performance of the model outside the training data range [-1,-1] increases the epistemic

uncertainty, which is a desired characteristic for determining the distance of the samples from the training data.

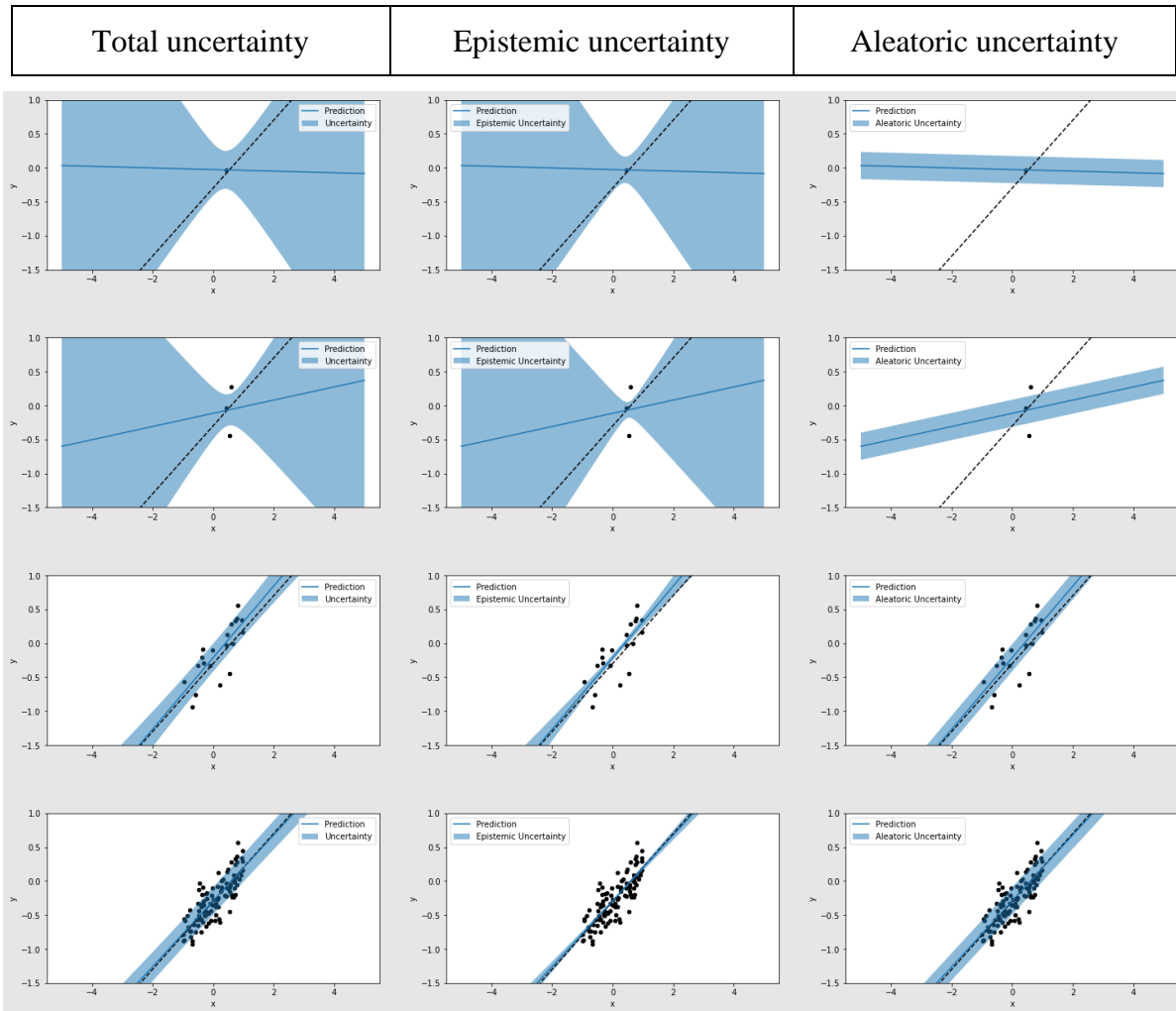


Figure 3.6: Bayesian Linear Model baseline uncertainties with Gaussian prior for (Top) 1 Training sample (Middle-1) 3 Training samples (Middle-2) 20 Training samples (Bottom) 100 Training samples: Left column – Total uncertainty; Middle column – Epistemic uncertainty; Right column – Aleatoric uncertainty

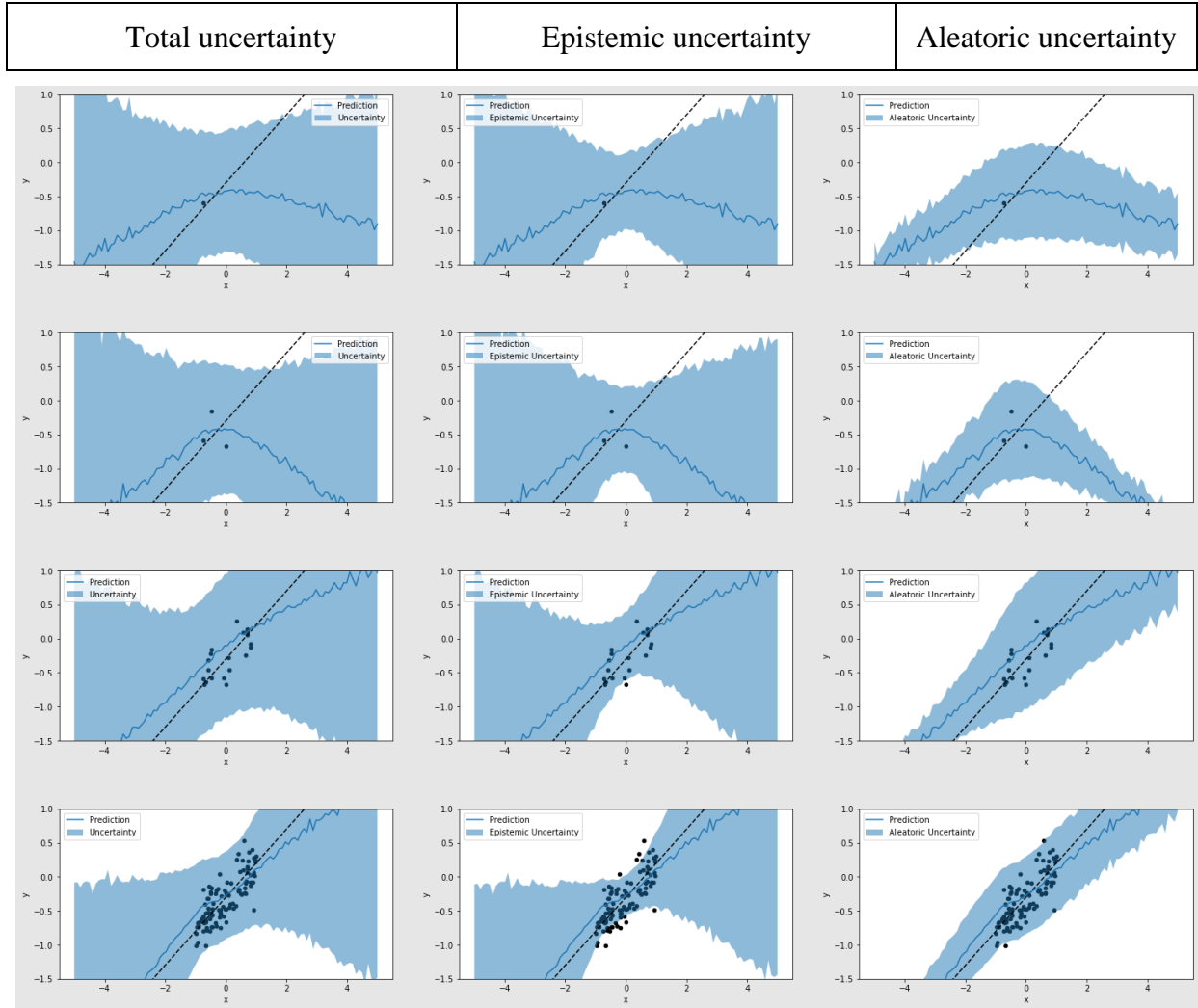


Figure 3.7: MC-Dropout uncertainties for (Top) 1 Training sample (Middle-1) 3 Training samples (Middle-2) 20 Training samples (Bottom) 100 Training samples: Left column – Total uncertainty; Middle column – Epistemic uncertainty; Right column – Aleatoric uncertainty

### Modeling – Classification uncertainty quantification:

Epistemic and aleatoric uncertainty is related to the decision boundary in classification. The neural network used outputs 2 quantities – the prediction, and its associated aleatoric variance, which is estimated implicitly without the need for ground truth variances. On the other hand, the epistemic uncertainty is captured by computing the variance of the MC Dropout draws. We verify the approach on the two moons dataset, visualized in Figure 3.8. The two moons dataset is a non-linearly separable dataset with 2 inputs. Since the model has 2 categories, the network outputs 4 values – 2 for the predictions and 2 for each of the variances. The variance terms are the input-

dependent uncertainties. For a classifier:

$$y_i | x_i \sim \text{Bern}(\psi(w^T x_i)) \quad 3.24$$

Assuming softmax activation  $\psi$ , the optimal weights can be derived using Maximum Likelihood Estimation (MLE):

$$\begin{aligned} w &= \text{argmax}_w p(y_i, x_i | w) \\ w &= \text{argmax}_w p(y_i | x_i, w) p(x_i | w) \\ w &= \text{argmax}_w p(y_i | x_i, w) \end{aligned} \quad 3.25$$

We model the prediction as a Gaussian distribution. The mean of this distribution is taken as the prediction  $f(x_i)^w$ . The variance term comes from the learned aleatoric variance output from the network. We then sample  $t$  times from this distribution and then average its log-softmax over the sampling dimension. To avoid underflow and overflow issues, this is implemented by first transforming the  $y_{i,t}$  Using the log sum exp trick and then averaging over the sampling dimension. The final form of the loss is implemented directly using the Negative Log-Likelihood (NLL) loss after obtaining the samples as described and comparing the prediction against the ground truth. The loss is stochastic because it depends on Monte Carlo drawing from the variance term.

$$\begin{aligned} y_i | w &\sim N(f(x_i)^w, \sigma_i^{w^2}) \\ y_{i,t} &= f(x_i)^w + \epsilon_t, \epsilon_t \sim N(0, \sigma_i^{w^2}) \\ L_x &= \text{NLL}(y_i, y_{gt}) \end{aligned} \quad 3.26$$

During inference, the procedure to compute uncertainty is by doing Monte Carlo sampling using dropout to obtain predictions. The epistemic uncertainty is computed as the variance of these predictions. The aleatoric uncertainty is the learned component of the prediction and is directly obtained from the formulation. The results in Figure 3.8 show that the epistemic uncertainty reduces with more training data while the aleatoric uncertainty remains oscillatory and independent of this trend. It also shows a demonstrative example of what the decision boundary looks like when using MC-Dropout. The areas with more data have a lower uncertainty in the



decision boundary, indicated by the lower amount of variance in classification. At the extreme ends of the decision boundary, we notice a widening of the noise profile, indicating a higher uncertainty for the discriminative model to assign a fixed label to a sample near the boundary.

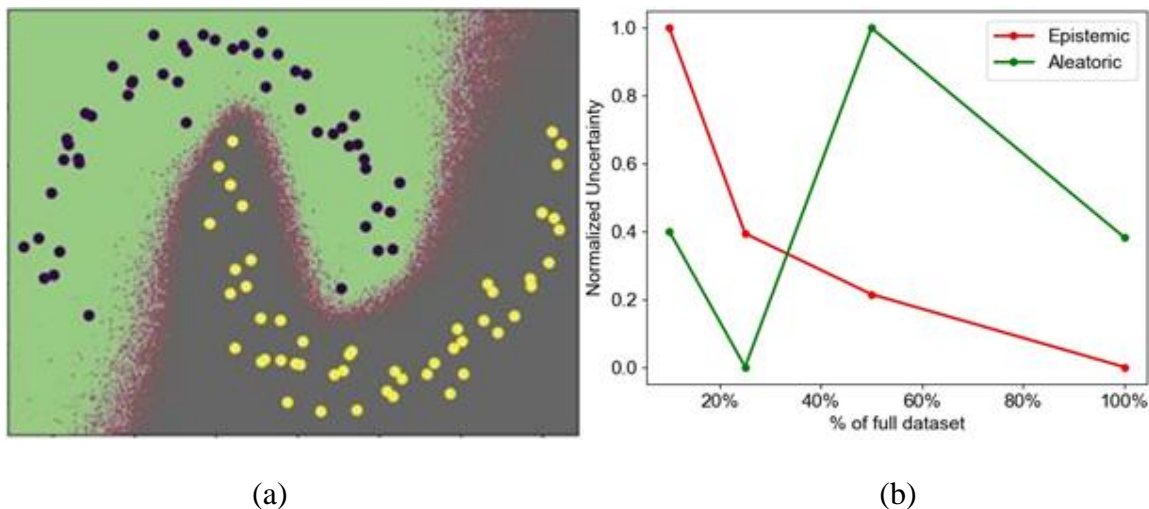


Figure 3.8: (a) Visualization of the two moons dataset with a visualization of the decision boundary when using MC-Dropout (b) Classification with uncertainty in the two moons dataset - Variation of uncertainties with sample size

### 3.1.3 Results and Discussion

The technique used for uncertainty modeling was verified in the earlier section to capture the desired behaviors for out-of-sample uncertainty, detecting the higher level of epistemic uncertainty when the test samples are further away from the training samples and an invariant aleatoric component with respect to the number of training samples. Therefore, the developed model was applied to modeling the uncertainty in the ASU Pipe defects dataset and the CrackForest dataset. The CrackForest dataset consists of 151 images of crack with asphalt backgrounds. The Concrete Crack dataset is a larger dataset but is only used for validation, after training with the CrackForest dataset. The results for the uncertainty quantification for each dataset is summarized in Figure 3.9. We also apply the trained model to the Concrete Crack dataset, and the uncertainty quantification results are compared against the CrackForest dataset in Figure 3.10. The use of these benchmark datasets helps evaluate the model on a larger defect detection dataset. It also helps evaluate the model performance when the training data distribution shifts away from the validation set, as in Figure 3.10. The results show that the epistemic uncertainty remains consistent across training

samples, surprisingly showing lower average uncertainty values than the original dataset for the lower sample regime. This observation was surprising, as we expected a gradual decrease in epistemic uncertainty, with a higher uncertainty value for the cross-dataset examples.

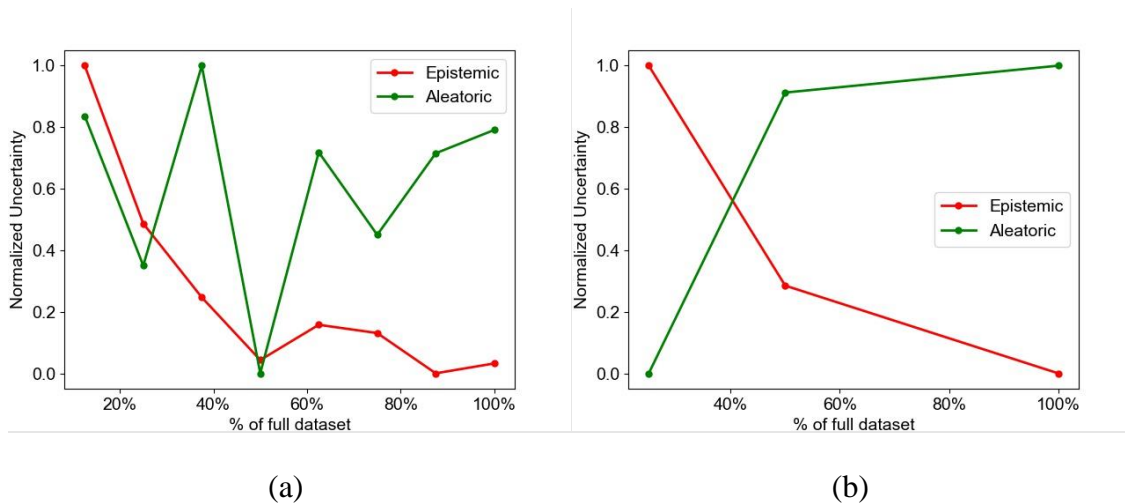


Figure 3.9: (a) CrackForest dataset and (b) ASU Pipe Dataset

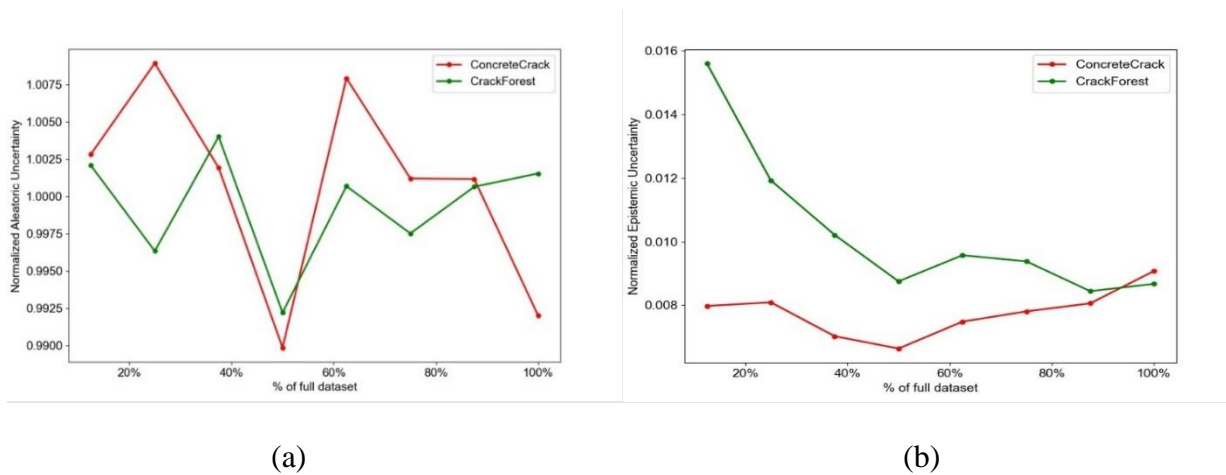


Figure 3.10: Uncertainty Evaluation for images from the ConcreteCrack dataset with a model trained on the CrackForest dataset: (a) Aleatoric Uncertainty and (b) Epistemic Uncertainty

A goal of predictive models under uncertainty is to ensure that a higher level of uncertainty is assigned to labels that deviate from the ground truth. This can be verified in the validation set, as the ground truth is available during evaluation. Figure 3.12 demonstrates how the epistemic uncertainty score tracks with the prediction quality, which is measured by the class-wise F1 score. Ideally, the points need to fall close to the red line, indicating an increasing uncertainty for samples

with poor detection scores in the test set. The top row of Figure 3.12 is calibrated for the background class and not for the defect classes, owing to a lower number of training samples overall in the dataset. To demonstrate that more data does help in improving uncertainty calibration, we overfit the model to a training set by splitting the training and validation sets after performing manual data augmentation, which includes horizontal and vertical flips, and image scaling to reduce the number of background pixels. While this approach is not recommended for testing generalization, the objective of this test was to verify calibration, assuming only small perturbations around the training data and with a greater number of training samples overall. We also use this dataset and model to demonstrate the defect measurement results, as shown in Figure 3.11. The objective was to evaluate the D435i performance for quantifying defects and not the model's generalization under a small training dataset. With more training data and a validation set close to the training set, the uncertainty correlates well with the background and pitting defect classes with a higher number of training samples compared to the crack defect. For the CrackForest dataset, the background class is again well calibrated with respect to the prediction F1 score. The concrete crack dataset, however, displayed an interesting trend, with the model producing nearly all its variance in confidence within a narrow band of F-1 scores. To further investigate this, we then added random brightness, saturation, contrast, and hue jitter along with standard Gaussian noise to the input images to further perturb them from the training sample space, as seen in Figure 3.14. The reduction in image quality significantly reduced the overall F1 score for the crack defect and altered the epistemic uncertainty for both the crack and the background. This implies that the original observation was just the model predicting a relatively higher uncertainty for these new samples, even with a good F1 score.

A demonstration of defect measurements and depth estimates is shown in Figure 3.11 with errors indicating that the segmentation method can produce accurate defect areas with the variance in predicting the defect boundary. The sensor can capture pitting depth on the millimeter scale but has larger errors owing to resolution limitations. The spatial resolution limits and the segmentation errors also affect the results for the width of thin cracks. The depth sensor was found to be unreliable below the millimeter scale. It has reliable detections of defect depths on the millimeter scale and high precision when the depth of the defect is at or above the order of 1/10th of a centimeter. Signals of interest captured at sub-millimeter scales can be used for anomaly detection, but the corresponding depth measurements will not be reliable. In pipe damage modeling, this

situation corresponds to early detection for corrosion and oxidization of metallic surfaces (B. Zhang & Ma, 2019). At the early stages of surface degradation, the dominant anomaly signal is increased metallic surface roughness, which can be detected but not precisely characterized by our stereo setup. Further improvements to the depth resolution for optical metrology at a sub-millimeter scale using commercial sensors is a promising research direction. This implies that commercial sensors must be optimized for sub-millimeter scale defects for ease of use in industrial metrology applications requiring precise defect depth estimates.

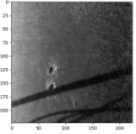
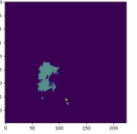
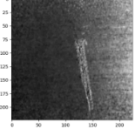
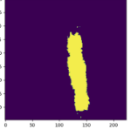
Image	Prediction	Defect	Measurement	From Prediction - Mean (SD)	From Ground Truth	
		Pitting	Area ( $mm^2$ )	1577.47(74.56)	1028.01	
			Depth( $mm$ )	4.93(0.09)	4.79	
			Length( $mm$ )	44.98(0.84)	37.74	
		Crack	Area( $mm^2$ )	-	732.79	
			Depth( $mm$ )	-	2.4	
			Length( $mm$ )	-	57.24	
		Crack	Area( $mm^2$ )	3660.01(29.20)	2353.41	
			Depth( $mm$ )	2.11(4.23)	0	
			Length( $mm$ )	86.73(1.54)	79.92	
			Width( $mm$ )	23.02(0.28)	17.05	

Figure 3.11: Defect measurement demonstration on the pipeline sample

Figure 3.15 shows some demonstrative examples of detections along with the corresponding uncertainties for the CrackForest and ConcreteCrack datasets. The detections for the ASU pipe dataset are shown in Figure 3.13. The uncertainty maps show that the source of uncertainty is largely at the boundaries of the detected defect. The variance in defect area profiles is useful for downstream prognostics, and obtaining uncertainty estimates along with defect morphology is essential for detailed risk evaluations. The reason why we have demonstrated this method on 3 types of datasets is to demonstrate the effectiveness of the method on various types of sensors. We can also use higher resolution depth cameras, time of flight and structured light cameras for collecting depth data. This gives users the ability to use transfer learning to adapt their model to new types of sensors.

Table 3.1: Comparing the performance of various types of input data at a constant dropout value

Data	% Of full dataset	Epistemic Uncertainty	Aleatoric Uncertainty	mF1
RGB	25%	0.0123	1.0132	0.7028
	50%	0.0130	1.0113	0.7566
	100%	0.0106	1.0135	0.7632
RGB-D	25%	0.015	1.0176	0.6573
	50%	0.0104	1.0177	0.7530
	100%	0.0101	1.0201	0.7555
RGB-DC	25%	0.0108	1.0500	0.6560
	50%	0.0098	1.0481	0.7455
	100%	0.0090	1.0456	0.7644
RGB-DNC	25%	0.0142	1.0284	0.6585
	50%	0.0113	1.0295	0.7527
	100%	0.0102	1.0296	0.7584

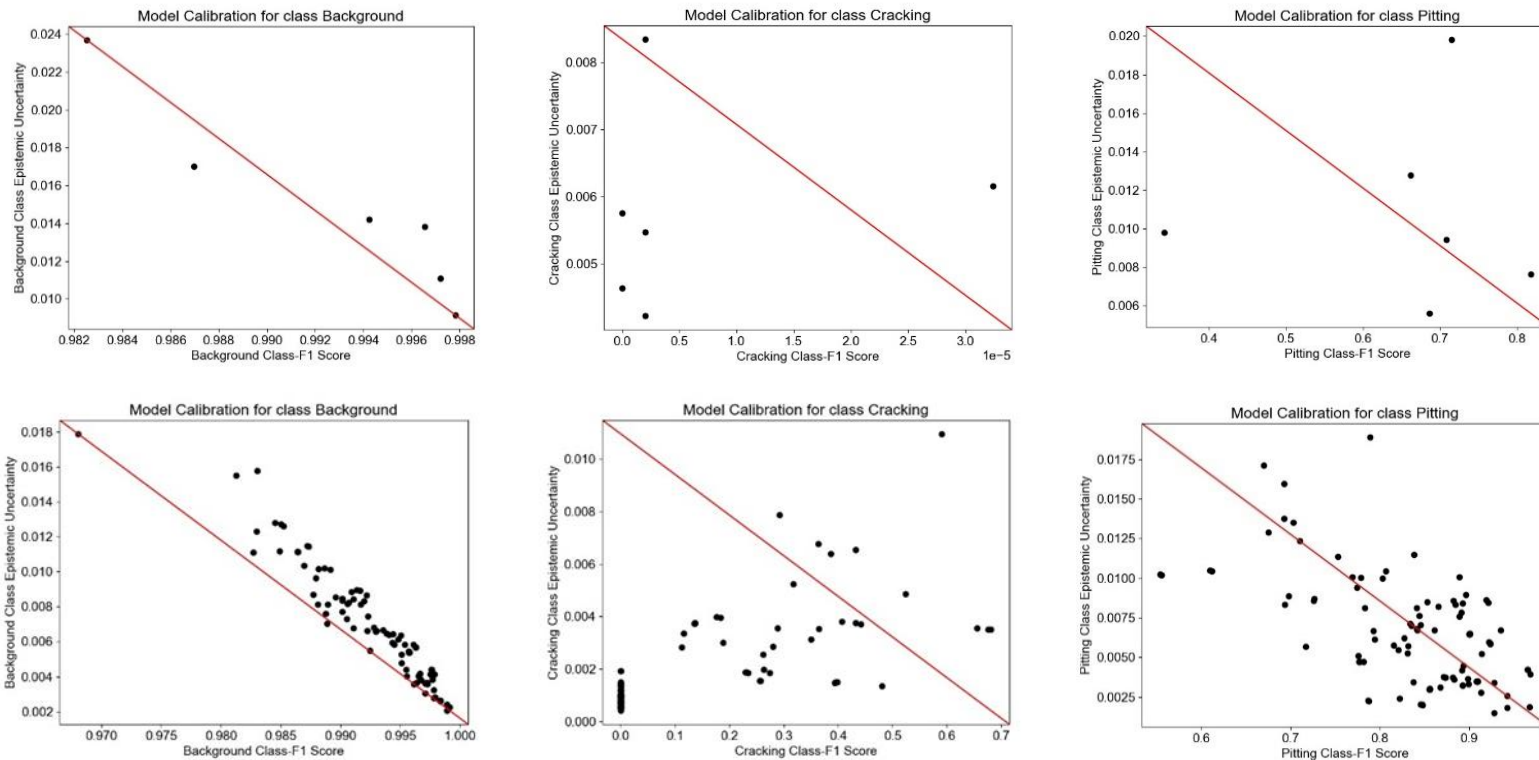


Figure 3.12 : Classwise Epistemic Uncertainty - F1 score calibration performance for the ASU Pipe RGB-D validation set across classes: Top row - Small ASU Pipe Dataset with a test set held-out (Left) Background (Middle) Cracks (Right) Pits, Bottom Row - Augmented ASU Pipe Dataset with test samples augmented from the training samples for verifying model calibration assuming the test set distribution is very close to the training data distribution (Left) Background (Middle) Cracks (Right) Pits

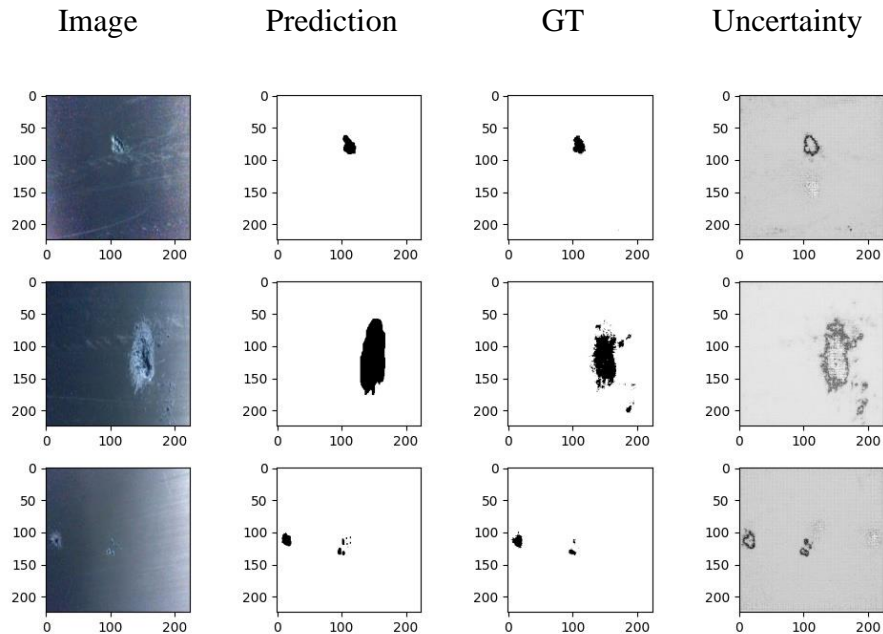


Figure 3.13: Demonstrative detections from the ASU pipe dataset: (Left to Right) Image, Ground Truth, Prediction and Total Uncertainty

Next, we analyze the various fusion layers' performance and input data types. Table 3.1 shows the comparative performance of the RGB, RGB-D, RGB-DC, and RGB-DNC input data at various training sample sizes at a dropout of 0.5. We also evaluate the model across various dropout ratios, as shown in Figure 3.16. From the results, the data types themselves do not cause substantial absolute improvements to raw detection performance. However, the ablation studies show that the epistemic uncertainty is lower when using the fused representations than only using RGB data, with the RGB-DC performing the best across all dropout ratios. The small dataset has some limitations in the number of claims that can be made on generalization, but results have shown a greater propensity for pitting defect accuracy improvements when using additional data apart from just RGB information, with slight improvements to the mean F1 (mF1) score by the incorporation of normal and curvature information instead of just the depth map. The F1 score is a harmonic mean of precision and recall. Precision computes the proportion of true positives (TP) to the total number of positive detections, including false positives (FP). Recall, on the other hand, computes the proportion of true positives to the sum of true positives and missed detections (false negatives).

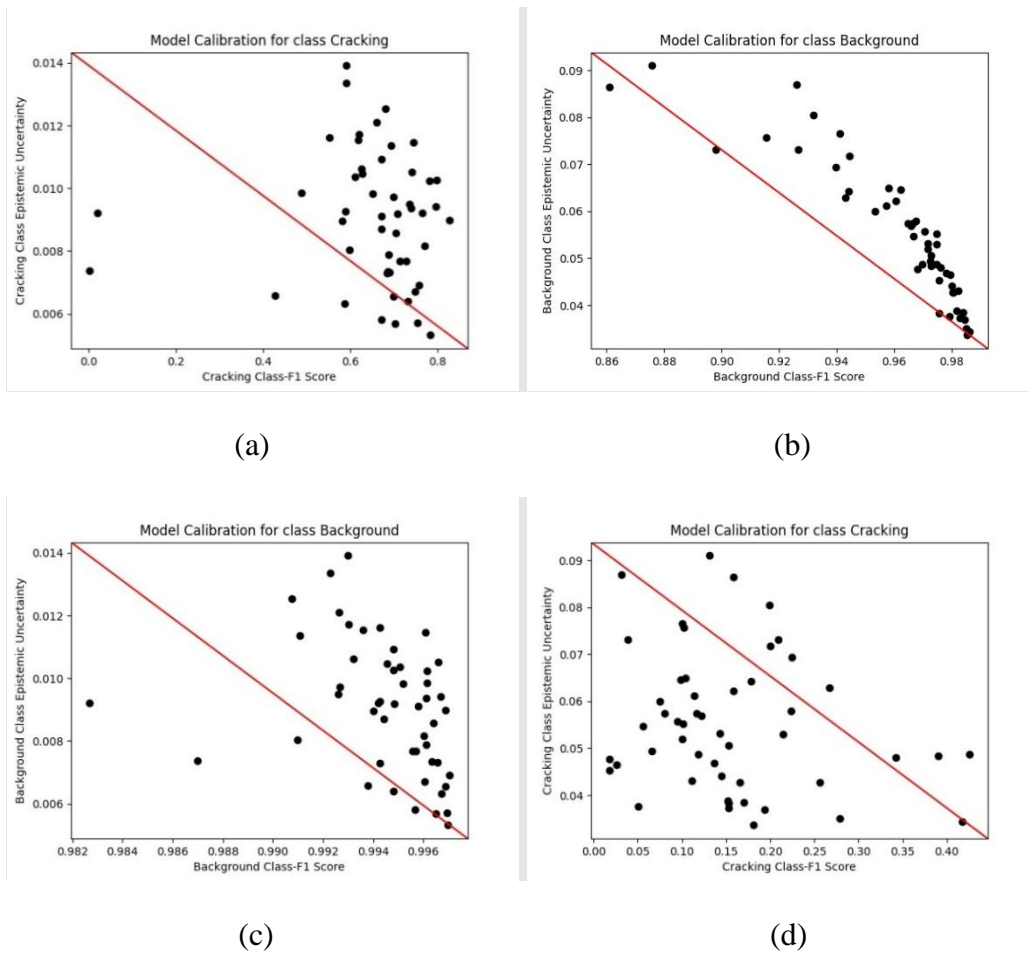


Figure 3.14: Classwise Epistemic Uncertainty - F1 score calibration performance for the ConcreteCrack dataset by class: Top-Row: Original unaltered data (a) Background (b) Crack, Bottom-Row: Data with added Gaussian noise and random brightness, contrast, hue and saturation jitter for (c) Background (d) Crack



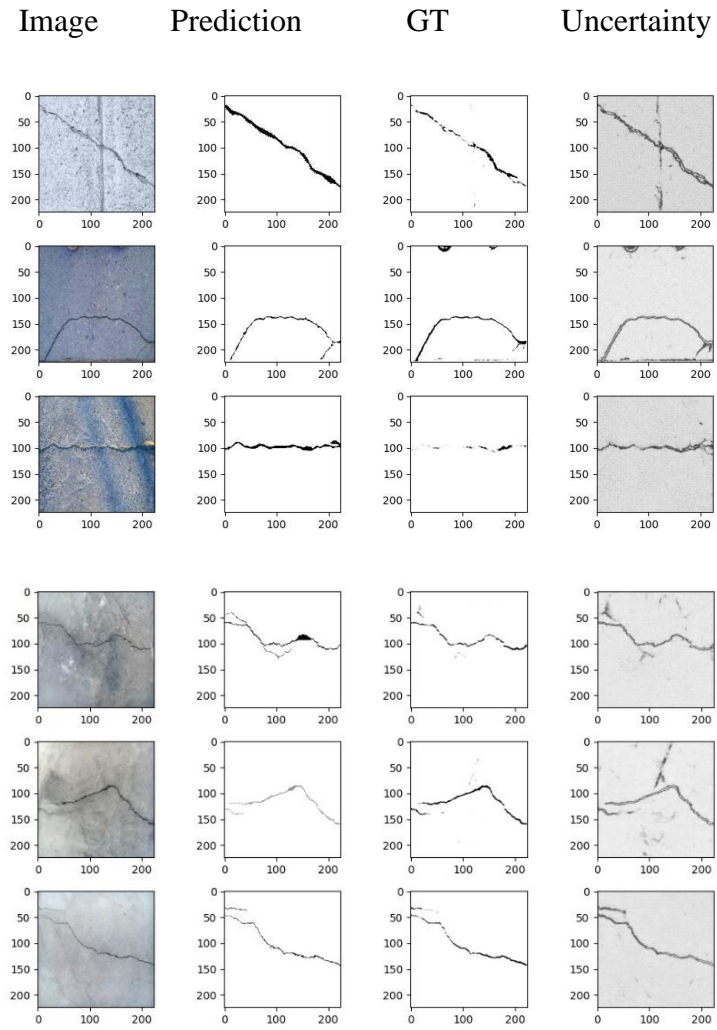
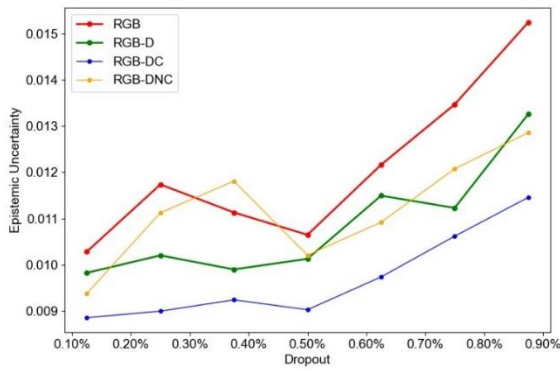
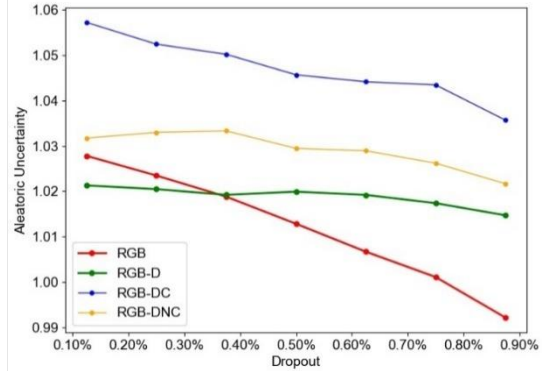


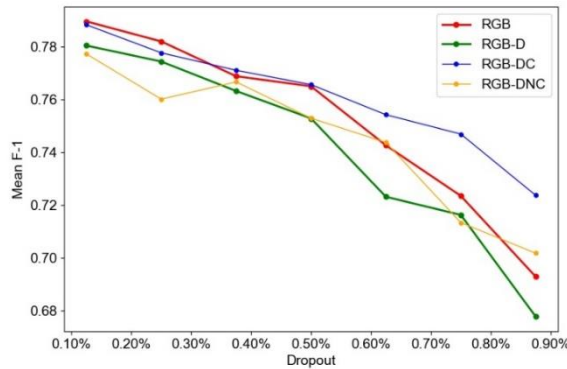
Figure 3.15: Demonstrative examples of detections in the CrackForest (Top-3 rows) and the ConcreteCrack Dataset (Bottom-3 rows). Images from Left to Right: Input, Prediction, Ground Truth (GT), Total Uncertainty



(a)



(b)



(c)

Figure 3.16: (a) Epistemic Uncertainty across various dropouts and input data types (b) Aleatoric Uncertainty across various dropouts and input data types (c) Mean-F1 across various dropouts and input data types

## 3.2 Semi-Supervised Semantic Segmentation using Activation Map Interpolation

### 3.2.1 Background

In the past decade, remarkable progress has been made in image segmentation due to the deep learning revolution. The advent of powerful GPUs has enabled the industry to utilize large, labeled training sets to fit deep networks with huge parameters. However, the pace of progress in computation power has not remedied the lack of quality-labeled training data, which remains a challenge. Furthermore, labeling data for semantic segmentation is also time-consuming and

expensive. Therefore, there is a need for a model that can learn from a small set of labeled examples and a larger set of unlabeled samples. This section proposes a segmentation method for surface defect detection, which will help condition monitoring and quality assessment framework for manufacturing and civil infrastructure applications. While the defect image characteristics vary between these applications, the data-driven approach used in our models mean that they can be easily adapted.

Defect detection databases are currently limited compared to natural image database. This leads to the lack of large datasets, which can hinder deep learning-based methods for defect detection. In addition, manual labeling of datasets is also expensive and involves tedious work for experienced engineers. These databases also present various challenges compared to natural images: They have fewer distinctive features and often require algorithms that can work in real-time. They also have high intra-class similarities and minuscule defects that can be hard to detect, such as hairline cracks. Defect detection has been a focus of multiple studies for classification and coarse-grained object detection. However, fewer studies focus on fine-grained detection and segmentation of defects. Pixel-level segmentation offers significant advantages over detecting object categories (classification) or coarse-grained bounding boxes, such as a better understanding of defect morphology, which would help improve downstream prognostics, risk assessment, and quality management. Given the issues surrounding data labeling requirements and the lack of quality samples, this study addresses them by focusing on fine-grained defect detection using a limited number of labeled samples and a larger corpus of unlabeled samples.

The proposed method will lie on the following cornerstones: The smoothness assumption, the cluster assumption, and consistency regularization. The cluster assumption says that the decision boundaries between classes should not overlap but instead be well-separated. The manifold hypothesis states that the dissimilar data that are generated from the same process occupy the same space in a low-dimensional manifold. Manifold learning, for example, finds the space that best separates the classes in a lower dimension. Therefore, changing the space in which the data lies can change a potentially non-linear decision boundary with intersections into well-separated clusters that obey the cluster assumption. If the cluster assumption holds, consistency regularization forces a network to produce consistent outputs for perturbed inputs. This is accomplished by penalizing predictions on the perturbed input that deviate from the expected

output. Doing so increases the prediction accuracy at the original data point and in its neighborhood. If the cluster assumption is violated, the consistency regularization may end up offering no benefit, as some of the samples may get misclassified.

The consistency regularization assumption in image classification settings is easy to justify. Given a labeled sample  $(x_L, y_L)$ , any corruption or perturbation of the input  $\tilde{x} = x_L + \epsilon$  should not change the classification target. However, in the context of segmentation, the task is to produce a dense pixel-wise prediction of classes. Some augmentations of the input, such as rotation or translation, can distort the inter-pixel relations and leads to a completely different output. Moreover, the cluster assumption is also shown to be violated at the image level, as low-density regions do not line up with the boundaries of the objects of interest, as demonstrated by French et al. Therefore, a method is needed to make the prediction robust to images in the neighborhood of the original sample. Consistency regularization constrains the predictions in the neighborhood of a sample to produce the exact prediction as the unperturbed samples. This approach is based on the validity of the cluster assumption. The simplest way to enforce consistency regularization is to perform random perturbations to the input. However, the space of possible perturbations in high-dimensional data is large, and randomly choosing perturbations can be ineffective.

The proposed method takes in a corpus of a small set of labeled images and a larger set of unlabeled images. The samples are passed into a multi-branch network, where a multi-component loss is learned jointly. Labeled samples are learned using a supervised loss, and unlabeled samples are learned using a consistency loss.

### **3.2.2 Related Work**

Defect detection using optical methods has been studied extensively. Before the deep learning revolution, a lot of the methods focused on hand-engineered feature classification and defect classification using classical machine learning and statistical techniques. (Y. Liu et al., 2019) used a multi-block local binary pattern (LBP) to filter features across multiple scales. (Y. Shi et al., 2016) performs road crack detection using engineered features and random forests. (Suvdaa et al., 2012) uses SIFT features followed by an SVM for defect classification. (M. Liu et al., 2016) uses image binarization optimization using genetic algorithms to segment defects in an unsupervised fashion, by using a thresholding approach based on inter and intra-class variance statistics. In most

of the classical image processing-based approaches, the process of detection is multi-stage, requires lower amounts of training data and have simpler mathematical models, and complicated algorithmic models. The advent of deep learning-based approaches saw rapid progress in test-set performance for defect detection algorithms, and the focus shifted to highly data-driven methods for detection. Classification based on deep learning has been explored by several works. (Y. Liu et al., 2020) uses a CNN architecture with fixed scales at two image resolutions, which compute features at the corresponding scales and are then concatenated for defect classification. (Chen et al., 2018) uses an ensemble of three networks to classify defect images, and F.A. Saiz et al. (Saiz et al., 2018) combines pre-processing steps with a CNN to increase robustness of the CNN to inter-class defects. Semi-supervised classification algorithms such as (Di et al., 2019; Wang et al., 2021) use varied methods to distinguish between textural samples with limited labeled data. (Di et al., 2019) uses a large unlabeled corpus to train a convolutional auto-encoder and a GAN is then used to increase the sample size. Finally, the encoder of the trained auto-encoder model is used to extract features from the data and is fed into a softmax layer for classification. (Wang et al., 2021) uses a graph-based deep network to perform classification. However, classification does not give us any indication of object location, without further post-processing. This gap was filled by various object detection (J. Li et al., 2018; Y. Li et al., 2019) and semantic segmentation methods. It has been shown that deep networks-semantic segmentation approaches achieve particularly good performance, with various architectural modifications to boost the performance in the validation and test sets, as in (Cheng & Yu, 2021; Dong et al., 2020; Z. He & Liu, 2020; Tabernik et al., 2020; Yang et al., 2020). However, a gap remains in addressing semi-supervised semantic segmentation in surface defect detection applications, which this paper will address.

The task of semi-supervised learning has been explored using the following approaches: Pseudo-Labeling and self-training (Lee, 2013), consistency regularization, interpolation consistency training (ICT) (Verma et al., 2019), and virtual adversarial training (VAT) (Miyato et al., 2019).

The pseudo-labeling approach takes in a corpus of labeled and unlabeled samples and generates fake targets for the unlabeled samples. For the segmentation problem, pseudo-labels can be generated using Class Activation Maps (CAM) (Zhou et al., 2016). However, the problem with pseudo-labels is that the errors made in target generation has no mechanism of correction. As a result, using incorrect pseudo-labels will tend to propagate these errors through the model during

backpropagation, and cause a positive feedback loop of erroneous predictions leading to incorrect weight updates. Self-training improves upon the pseudo-labeling approach by successively adding in new samples into the corpus and by selecting the samples that it is most confident about.

Consistency regularization constrains the predictions in the neighborhood of a sample to produce the same prediction as the unperturbed sample and is based on the validity of the cluster assumption. The simplest way to enforce consistency regularization is to perform random perturbations to the input. However, the space of possible perturbations in high dimensional data is large and randomly choosing perturbations can be ineffective. Among the more sophisticated consistency regularization models, the VAT model uses an adversarial perturbation to perturb the input in the direction of the greatest predicted output change. The ICT approach uses an interpolation between a pair of unlabeled samples such that the new sample lies in between the two, and consistency is enforced by constraining the transition curve between the two points is linear. CutMix (Yun et al., 2019) trains classifiers by introducing a random rectangular mask in the image to inpaint another image, and effectively combine the two images to produce a perturbation. (French et al., 2019) combines CutMix with the Mean Teacher model (Tarvainen & Valpola, 2017), and predict pixel-wise semantic segmentation in a semi-supervised fashion. This paper takes inspiration from the ICT approach to work for semantic segmentation by operating on the probability space defined by the activation map of the features, instead of using the input space.

### 3.2.3 Methodology

Let  $(x_i, x_j)$  be a pair of unlabeled samples extracted from the data and let  $(x_i^{CAM}, x_j^{CAM})$  be the activation maps of the representation. The class activation maps are obtained using a global average pooling (GAP) layer (Zhou et al., 2016). The GAP performs a channel-wise spatial averaging of data, and this is multiplied by the weights  $w_k^c$  to yield the activation map  $M_c$  as follows:

$$F^k = \sum_{\{x,y\}} f_k(x, y) \tag{3.27}$$

$$M_c(x, y) = \sum_k w_k^c f_k(x, y) \tag{3.28}$$

Images at the input level are high-dimensional representations that do not strictly obey the cluster assumption French, et.al, 2019. On the other hand, the hidden representations are at a lower dimension and were expected to have better clustering properties. This is identified by plotting the hidden representations' patch-wise Euclidean distances by first obtaining a class activation map and then interpolating it to the original image size, as shown in Figure 3.17.

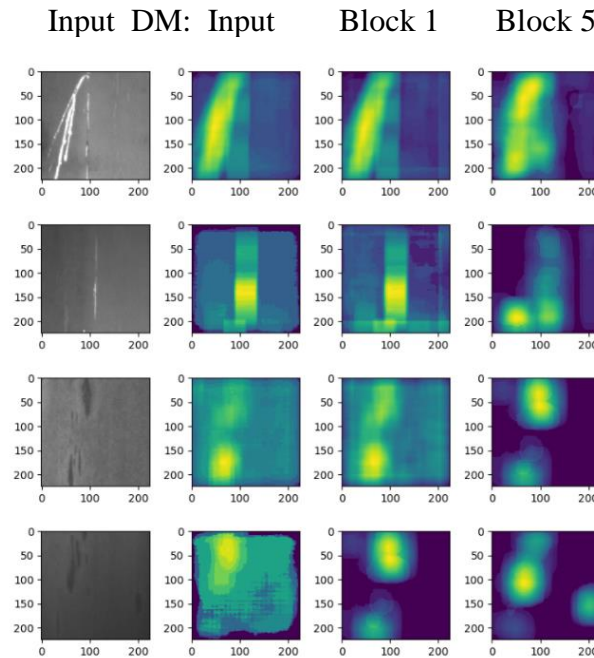


Figure 3.17: Cluster assumption demonstration by computing the patch wise Euclidean distance between a patch and its neighbors. Higher values are indicated with yellow: The left-most image is the input, followed by the distance maps (DM) of the following: the input image, the activation map from the end of convolutional Block 1, and the activation map from the end of Block 5.

The patch-wise distances better conform to object boundaries at the hidden representation level than the input level. The calculated class activation map produces pixel-wise probability distributions that can be interpolated. The element-wise sum along the channel dimension is 1. Therefore, the class activation map is a  $W \times H$ -dimensional distribution. Assuming that each pixel in the activation map is i.i.d (independent and identically distributed), they can be interpreted as containing a probability distribution per pixel. By extension, it can also be argued that interpolating the two activation maps can produce another probability distribution, as the operation performed is a convex combination with every resulting value being positive. In interpolation consistency

training, the interpolation is performed using the following formulation directly on the input data sample:

$$x_{mix} = \lambda x_i + (1 - \lambda)x_j \quad 3.29$$

$$\lambda \sim \text{Beta}(\alpha, \alpha)$$

Following the ICT approach, a non-negative mixing value is sampled from the beta distribution. The difference between this approach and the one described in ICT is that the interpolation happens for segmentation instead of classification, and the level at which the mixing happens is not at the image-level but at the level of the hidden layers. The resulting representation is then passed through decoder layers to obtain a final set of features with the same dimension as the image. The inputs  $(x_i, x_j)$  are also passed through the decoder separately and mixed using the mixing function again at the higher dimension. To make this similar to the density from the input side, the features are operated on by the softmax and the KL divergence loss between the mixed output  $\tilde{y}_{mix}$  and the output obtained by combining the predictions from  $f_{\theta'}(x_i), f_{\theta'}(x_j)$  is computed:

$$L_{cons} = H\left(\tilde{y}_{mix}, \text{mix}(f_{\theta'}(x_i), f_{\theta'}(x_j))\right) - H(\tilde{y}_{mix}) \quad 3.30$$

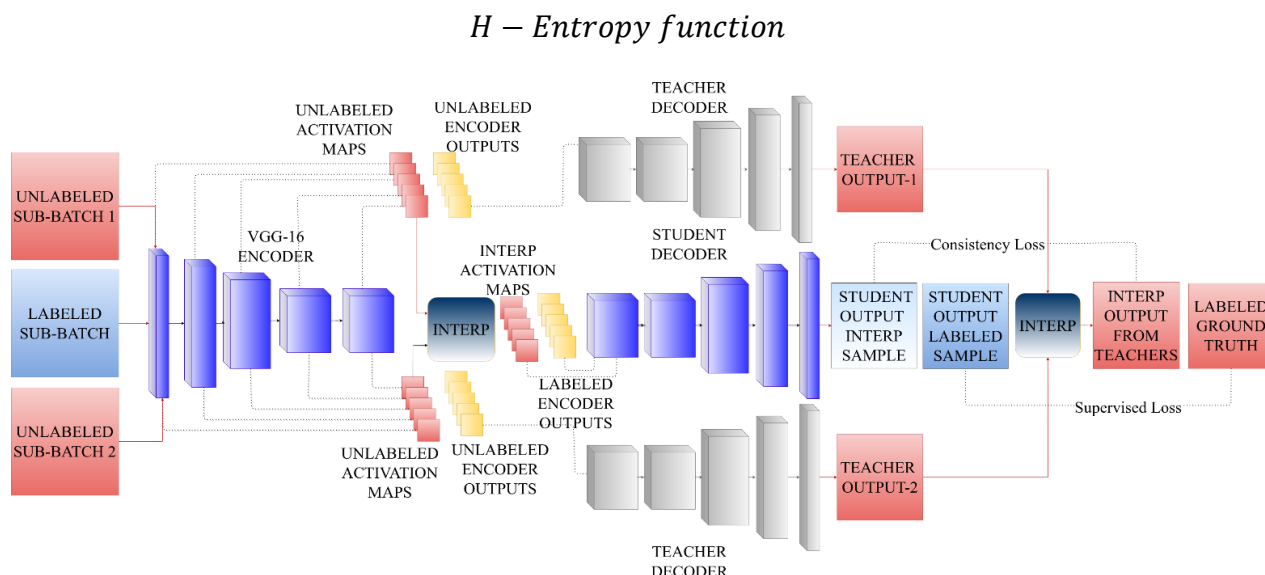


Figure 3.18: Overall methodology for semi-supervised segmentation using interpolation consistency in the probability space: Blocks with the same colors denote related elements.



Figure 3.18 shows the overall methodology, where the network designated  $f(\theta')$  is obtained as the exponential moving average of the weights of the central network  $f(\theta)$ , as in the mean-teacher algorithm (Tarvainen & Valpola, 2017). The student decoder is the only decoder branch to have its weights updated by backpropagation, and the knowledge from the student is distilled into the teacher branch using the mean teacher algorithm. The mean teacher algorithm ensures that the teacher model weights are updated according to the student decoder's exponential moving average (EMA) at every step of the model.

$$g(\theta_t) = \alpha g(\theta_{t-1}) + (1 - \alpha)f(\theta_t) \quad 3.31$$

Choosing the mean teacher model in classification settings improved performance when the number of labeled samples was scarce, even when compared to the Temporal-Ensembling model (Laine & Aila, 2017).

The supervised loss function is computed on the outputs from the student decoder, and the loss is the standard cross-entropy loss:

$$L_{sup}(\tilde{y}_l, y_l) = y_l \log(\tilde{y}_l) + (1 - y_l) \log(1 - \tilde{y}_l) \quad 3.32$$

The consistency loss function is computed by using the outputs from the teacher decoder and the student decoder outputs. The loss is the KL-divergence loss, which is based on the relative entropy function between two distributions:

$$L_{cons}(\tilde{y}_l, y_l) = H\left(\tilde{y}_{mix}, \text{mix}\left(f_{\theta}(x_i), f_{\theta'}(x_j)\right)\right) - H(\tilde{y}_{mix}) \quad 3.33$$

$$H_{P-Q} = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

The total loss is computed as the combination of the supervised and consistency losses, with a ramp-up weight function  $w(t)$  for the consistency loss such that it becomes more dominant in later epochs:

$$L = L_{sup} + w(t)L_{cons} \quad 3.34$$

The overall network architecture consists of an encoder, and two decoders, the teacher decoder,

and the student decoder. This is in keeping with the mean teacher model and the interpolation consistency training approach mentioned in the preceding sections. The encoder is a VGG-16 network pretrained on the ImageNet dataset. The final fully connected layer and average pooling layer is removed and replaced with an adaptive average pooling layer. The fully connected layer is replaced with a 1x1 convolutional layer at the output of each of the 5 sub-blocks of the VGG-16 network, used to compute class activation maps at each stage. The decoder consists of a series of transposed convolutional layers, to upsample the image back to the original size.

The dataset is loaded into the network using a batching method that provides 3 sub-batches for every iteration: 1 sub-batch for labeled images, and the other two consist of unlabeled images. The labeled images are iteratively sampled with replacement until all the unlabeled images are accounted for. The labeled image passes through the common VGG-16 encoder and the student decoder. The unlabeled image first passes through the common encoder, and the corresponding class activation maps are extracted and normalized to sum to 1 in the class dimension. Each of the 5 activation maps from the unlabeled data pair is interpolated, and the interpolated activation maps are passed into the student decoder and resized back into the right channel dimensions using a 1x1 convolution.

The remaining unlabeled samples are passed through the teacher decoder and interpolated after the outputs are obtained. The consistency loss is computed between the output obtained from the interpolated inputs through the student decoder and the output interpolated after they pass through the teacher decoder. This loss function is defined to be the KL-divergence loss, which measures the similarity between probability distributions. In this case, the KL-divergence is computed by summing over the class dimension, and then averaging over the batch and spatial dimensions.

### ***3.2.4 Results and Discussion***

We obtain results for this approach on benchmark datasets shown in Table 3.2 and compare it against related works and show competitive performance. We first present the results obtained on the NEU surface defect dataset and compared against the baseline supervised training method for various parameter settings and labeled dataset sizes. These results are for a training set that consists of 600 images, and a validation set that consists of 300 images from the surface defect dataset. The training and validation sets are kept separate during the training of the model, so the model does

not use any of the validation samples for training. Loss function weighting is a training hyperparameter. For the NEU dataset, the model performed best when the loss components were both unweighted. The results show that the mIU obtained in the validation set using the semi-supervised approach exceeds the fully supervised training baseline by 25% when there are only 30 labeled samples, and by 23% when there are 60 labeled samples, as shown in Table 3.3.

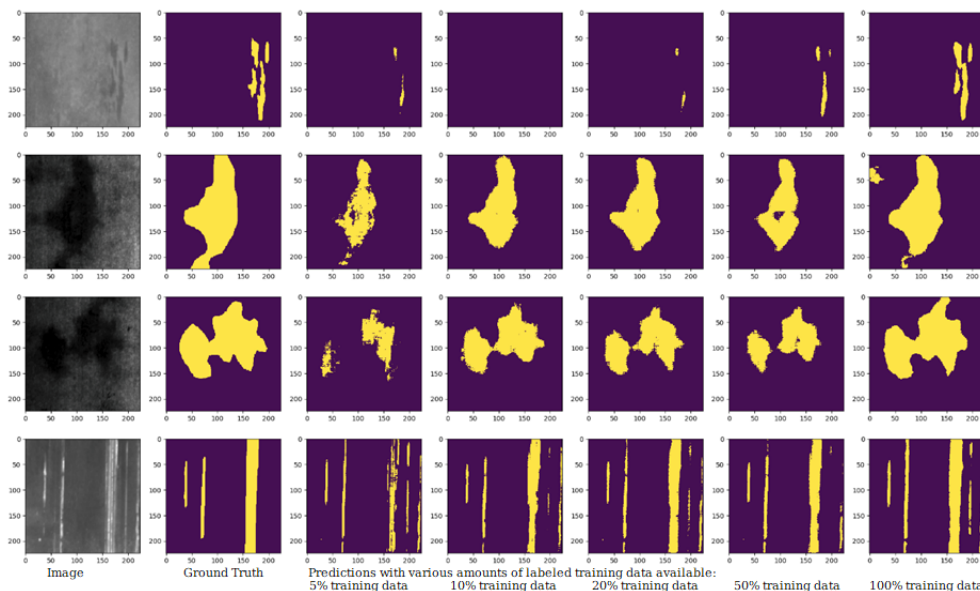


Figure 3.19: Demonstrative detections on the NEU validation set: (Left to Right) Image, Ground Truth, Predictions: 5% labeled training data, 10% labeled training data, 20% labeled training data, 50% labeled training data, 100% labeled training data.

Figure 3.20 shows some sample detections while using only 30 labeled images. The demonstrations indicate that the semi-supervised learning model manages to obtain correct detections in places where there are none in the supervised case and manages to eliminate spurious noise produced in the detections of the supervised model. Figure 3.19 summarizes a compilation of detections in the validation set with various amounts of labeled data.

Table 3.2: Dataset sizes

Dataset	Training	Validation
NEU	600	300
Crack Detection	135	16

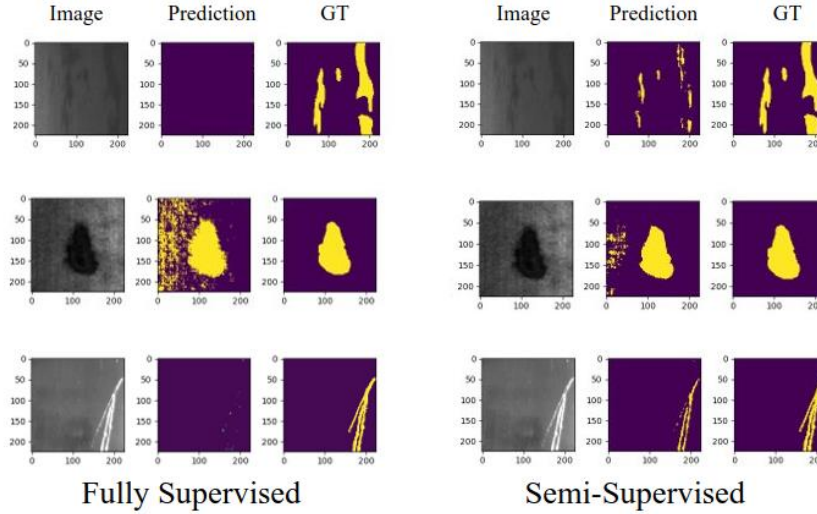


Figure 3.20: Demonstrative detections illustrating the effect of adding additional unlabeled data to the set and using the consistency loss. In this case, the comparison is made between the fully supervised case with 30 labeled samples, and the semi-supervised case with 30 labeled samples and 570 unlabeled samples.

The results in Table 3.3 show that the addition of these unlabeled samples, along with the consistency loss, provided a boost to the performance of the semi-supervised model, as compared to the purely supervised model. The table also shows, however, that the performance benefit provided by the unlabeled samples starts to drop off as more data is added to the labeled list, with a 50% labeled set providing only 1% less performance as compared to the semi-supervised model. Having a 30-sample labeled set provided an IU that was 20% lower than training with the fully supervised set, with all 600 samples being labeled.

A comparison was made between equally weighted losses and a ramp-up function applied to the consistency loss over several epochs. In this case, the number of epochs was chosen to be 80 empirically, using the evolution of the supervised loss function in the unweighted case. This made the supervised loss component dominant in the initial stages of the training. However, the performance produced by this approach was consistently lower on the NEU dataset than that produced when both the losses were weighted equally, as shown in Table 3.4.

To evaluate how much better the addition of 30 labeled samples is to the model, the model was trained with only unlabeled samples. This meant that the network did not have any supervised loss component but only an unsupervised loss. This resulted in a poor mIU of 8%, which was improved

more than 7x when a limited supervised set of 30 samples was added.

Finally, the results obtained from the proposed method were compared against benchmark results from recent works in Table 3.5, and is shown to have comparable results with state of the art in fully supervised learning, and the semi-supervised benchmarks perform at approximately 20% lower than the fully labeled, fully supervised training model for the 5% labeled case, and 15% points lower in the 50% labeled case.

Table 3.3: Metrics computed for the validation set at various proportions of labeled samples, with a performance comparison between the loss ramp-up approach and the equally weighted losses approach

Labeled Samples	Supervision	mIU
5%	Ramp	0.46
	No Ramp	0.64
10%	Ramp	0.53
	No Ramp	0.66
20%	Ramp	0.57
	No Ramp	0.64
50%	Ramp	0.63
	No Ramp	0.69
100%	Supervised	0.84

The second dataset to be considered for evaluation is the CrackForest Dataset (CFD). This dataset consists of a set of 151 labeled images and comprises a binary segmentation problem along with their segmentations and ground truths. The challenge posed by this dataset is in the ability to capture the crack morphology accurately. The experiments were conducted using 135 labeled samples, along with data augmentation for the supervised baseline, and semi-supervised benchmarks were computed as well. A comparison between the supervised baseline, other supervised deep learning architectures, and traditional image processing methods is made. The results indicate that the supervised baseline exceeds the performance of most of the preceding

works, as shown in Table 3.6, with a pixel tolerance of 0 for measuring true positives. Pixel tolerances were introduced for this dataset to measure performance in other benchmarks, as the ground truth measurements can have uncertainties, and the morphology of the detected cracks need not align exactly with the pixels marked out in the ground truth. However, in this study, the network used had enough parameters to be able to capture fine-grained localization information and therefore did not have the issue of uncertainties in learning pixel-wise locations, unlike in traditional image processing methods. The training set consisted of 135 labeled images, leaving 16 images for testing. The number of training samples was chosen to ensure ease of batching of the semi-supervised data sampler. The performance gap between the supervised and semi-supervised benchmark was narrower in this case due to the relative simplicity of this dataset. Some demonstrative examples are shown in Figure 3.21.

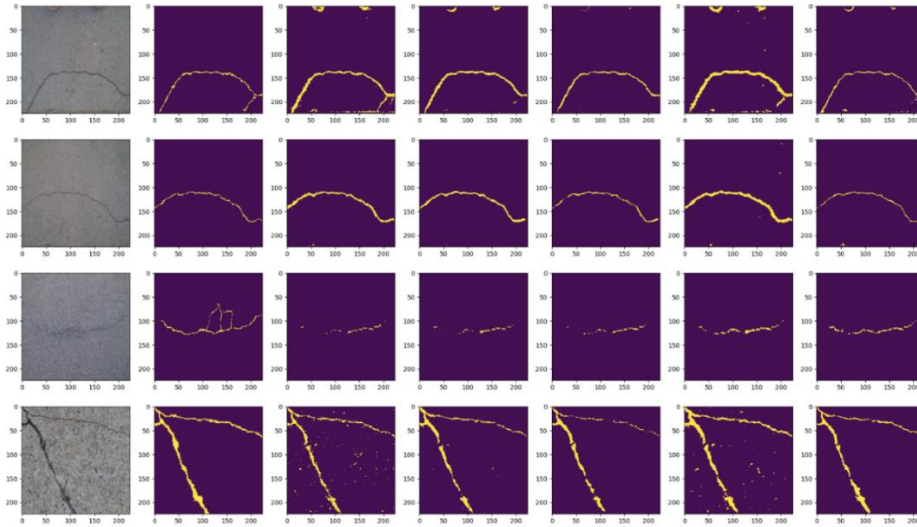


Figure 3.21: Demonstrative detections on the CrackForest validation set: (Left to Right) Image, Ground Truth, 5% Labeled, 10% Labeled, 20% Labeled, 50% Labeled, Fully Labeled

Table 3.4: Metrics computed for the validation set at various proportions of labeled samples, with a performance comparison between the semi-supervised model and the fully supervised model

Labeled Samples	Supervision	mIU
5%	Supervised	0.39
	Semi-Supervised	0.64
10%	Supervised	0.43
	Semi-Supervised	0.66
20%	Supervised	0.51
	Semi-Supervised	0.64
50%	Supervised	0.68
	Semi-Supervised	0.69
100%	Supervised	0.84

Table 3.5: Comparison of the results obtained in the test set for the NEU dataset

Method	Type	Labeled	mIU
SegNet	Fully Supervised	100%	0.5657
PSPNet	Fully Supervised	100%	0.7225
PGA-Net	Fully Supervised	100%	0.8215
Ours	Fully Supervised	100%	0.8309
Ours	Semi Supervised	5%	0.6429
Ours	Semi Supervised	10%	0.6674
Ours	Semi Supervised	20%	0.6454
Ours	Semi Supervised	50%	0.6992

Table 3.6: Comparison of results obtained in the test set for the CrackForest dataset

Method	Type	Labeled	Tolerance (px)	mIU	F-1
Canny	Image Processing	-	5	-	0.1576
CrackForest	Image-level labels + Hand engineered features	-	5	-	0.8571
U-Net	Fully Supervised	100%	0	0.55	0.7015
Res U-Net + ASPP	Fully Supervised	100%	0	0.56	0.7121
Ours	Fully Supervised	100%	0	0.7312	0.8443
Ours	Semi Supervised	5%	0	0.6944	0.8194
Ours	Semi Supervised	10%	0	0.6881	0.8151
Ours	Semi Supervised	20%	0	0.7109	0.8310
Ours	Semi Supervised	50%	0	0.6733	0.8047



## 4 DAMAGE PROGNOSTICS AND PHYSICS-BASED MODELS FOR INTERACTING THREATS

### 4.1 Semi-Empirical models for failure pressure and remaining useful life estimates

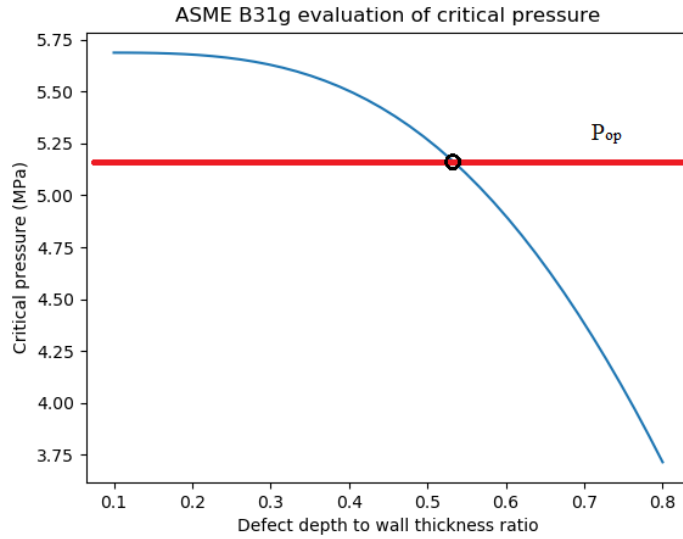


Figure 4.1: Prediction of failure pressure with depth to wall thickness ratio for a carbon steel pipe

This section summarizes a prognostic failure analysis for a single pitting defect using the pipe failure formulation from ASME B31G (The American Society of Mechanical Engineers, 2009). For a cylindrical pipe with yield strength  $S_y$ , wall thickness  $t$ , outer diameter  $D$ , and effective length of the defect  $L$ , the allowed normal operating pressure is given by Barlow's formula as:

$$P_{op} = 2 * t * \frac{S_y}{D} \quad 4.1$$

For a single corrosion defect that has a time-dependent depth and length  $d(t), l(t)$ , the failure pressure equations are given by:

$$P_f = \frac{1.1S_y 2t}{D} \left( \frac{1 - 0.85 \left(\frac{d}{t}\right)}{1 - \frac{0.85 \left(\frac{d}{t}\right)}{M1}} \right) \quad 4.2$$

where  $M1$  is the Folias factor computed as:

$$M1 = \sqrt{1 + 0.6275z - 0.003375z^2}, \quad z = \frac{L^2}{Dt} \leq 50 \quad 4.3$$

$$M1 = 0.032z + 3.3, \quad z = \frac{L^2}{Dt} > 50 \quad 4.4$$

Figure 4.1 shows a simple deterministic demonstration of the point of failure of the pipe with a single pitting corrosion defect of a particular size and a deterministic growth rate. However, growth rates and measurements of defect sizes have uncertainties. Therefore, to obtain the failure probability, we perform a Monte-Carlo simulation with the defect size being a random variable due to measurement uncertainty and the growth rate being a random variable due to model uncertainty. The growth rate is assumed to be a linear function (Vanaei et al., 2017), but this model does not consider the correlations between the defect's length and depth growth rate. The variables are drawn from a standard normal distribution  $N(\mu, \sigma)$  as:

$$N_{depthGrowth}(\mu_d = 0.1, \sigma_d = 0.01) \quad 4.5$$

$$N_{lengthGrowth}(\mu_l = 0.1, \sigma_d = 0.01) \quad 4.6$$

$$N_{initialdepth}(\mu_{id} = 0.5, \sigma_{id} = 0.01) \quad 4.7$$

$$N_{initialLength}(\mu_{il} = 0.5, \sigma_{il} = 0.01) \quad 4.8$$

We can define a limit state function  $z$  such that  $z = P_f - P_{op}$  and the probability of failure is computed by considering the simulation outcomes such that  $z \leq 0$ . The measured depth and the length were first sampled from a normal distribution accounting for measurement uncertainty. Then, the growth rate was also sampled from a normal distribution to account for model uncertainty. The range of times for prognostics was from 10 years to 100 years, and the failure pressure was calculated for upto 100 years. The pipe is considered to fail if the failure pressure goes below the operating pressure, and if the predicted depth of the defect exceeds 90% of the pipe wall thickness. For the next iteration, we sample once more from the normal distributions for the depth and length of the defect along with their growth rates, and the process is repeated. The

instances of failure are counted, and if the probability of failure exceeds 20%, the pipe is said to be unreliable.

## 4.2 Kriging model trained on FEM data for failure analysis

The active learning Kriging model (Echard et al., 2011) is an interpolation method used to estimate a response at a certain point based on covariances. It is also used to determine the local variance at a point in addition to predicting the value, unlike the Response surface model, which is only useful in predicting the performance function's sign. The role of limit state function  $G(x)$  is to find the local variance as well as the value at the point. The active Learning Kriging model is an approach where the mathematical model is updated after every prediction making the next prediction more accurate.

The first step of Kriging consists of defining this stochastic field with its parameters according to a design of experiments. The Best Linear Unbiased Predictor (BLUP) is used to estimate the value at a given point.

$$G(x) = F(x, \beta) + z(x) \quad 4.9$$

$F(x, \beta)$  is the deterministic part which gives an approximation of the response in mean.

$$F(x, \beta) = f(x)^t \beta \quad 4.10$$

where  $f(x)^t = \{f_1(x), \dots, f_k(x)\}$

$z(x)$  is a stationary Gaussian process with zero mean and covariance between two points of space  $x$  and  $w$  defined by:

$$cov(z(x); z(w)) = \sigma_z^2 R_\theta(x; w) \quad 4.11$$

where  $\sigma_z^2$  is the process variance and  $R_\theta$  the correlation function defined by its set of parameters  $\theta$ . The Kriging variance  $\sigma_G^2(x)$  is defined as the minimum of the mean squared error between  $\hat{G}(x)$  and  $G(x)$ . It can be expressed as the following analytical function:

$$\sigma_G^2(x) = \sigma_z^2 \left( 1 + u(x)^T (1^T R_\theta^{-1} 1)^{-1} u(x) - r(x)^T R_\theta^{-1} r(x) \right) \quad 4.12$$

where,  $u(x) = 1^T R_{\theta}^{-1} r(x) - 1$

#### 4.2.1 Methodology

The methodology uses an iterative scheme to calculate the cumulative failure probability over time using MC simulation. Initial defect sizes are sampled from a distribution over the surrogate model input parameters and are updated using a linear growth rate function. Then the responses are plotted for the parameters across time. For example, for the four responses observed in 15 pipes across 10 years, the values will be stored in a 15x10x4 matrix. The limit state function to assess probability of failure is as follows:

- Max 1st principal stress at the corrosion patch surface while Pressure=100% SMYS > 65 ksi
- Avg 1st principal stress along the tip of the crack > 94.3 ksi
- Max 1st principal stress along the crack tip > 114 ksi
- Max 1st principal stress within the volume > 133.6 ksi

If any of the criteria is met, then the failure case is recorded. The number of pipes failed is stored in an array with  $N_f$  being calculated across each time step.

#### 4.2.2 Results and Discussion

This section outlines the work done in this quarter with the surrogate model. We first get baseline results for the surrogate model and compare that against the original implementation with parameters taken within the training data range. Then, we compare the results of cumulative failure probability against the ASME B31G model. Finally, we attempt to remove the effect of the crack from the surrogate model, thereby going away from the training data distribution of the surrogate model and show that this does not capture the expected behavior from the model.

The baseline input parameters for the kriging model along with the ranges selected for this demonstration is shown in [Error! Not a valid bookmark self-reference.](#). The growth rate for the defects is taken at a conservative 4 mpy. While this growth rate produces only approximate estimates, for the purposes of benchmarking and preliminary analysis, we have assumed that this is warranted. Figure 4.2 predicts earlier failure instances when given similar initial conditions for both models.

Table 4.1: Input parameters for the surrogate model for the baseline experiment

Parameter	Distribution
Initial crack depth	$N\sim(0.0869e^{-3}, 0.0454)$
Initial crack length	$N\sim(0.5772e^{-3}, 0.2409)$
Initial corrosion depth	$N\sim(0.0311e^{-3}, 0.0134)$
Distance from the corrosion patch	$N\sim(-0.0427, 3.66)$
Distance from the center of crack	$N\sim(0.0537, 3.66)$
Growth rates	4 mpy - Constant

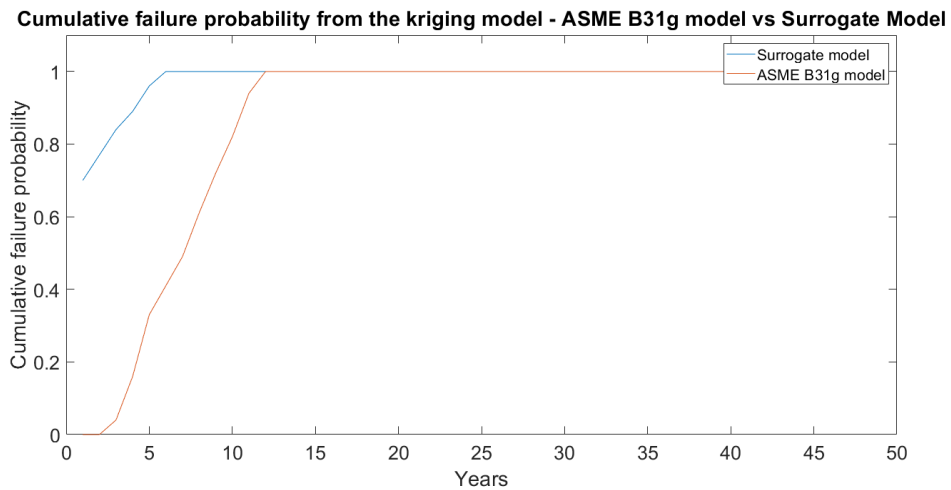


Figure 4.2: Comparing the cumulative failure probability of the surrogate model and the ASME B31G model

### 4.3 Assessment of X65 Gas Pipeline using FEA for Interacting Corrosion Pits and Comparison with ASME B31G

Finite element analysis has been a critical tool in the engineering industries in past decades. With improvements in the computational hardware, the FEA analysis is much cheaper than in the last decades. The X65 gas pipeline with corrosion defects can be analyzed using the Finite element analysis to predict the burst pressure.

#### 4.3.1 Verification and Validation of FEA analysis

The Finite element analysis has been verified and validated by comparing the FEA result against the experimental results obtained by Choi et al., in (Choi et al., 2003). Therefore, the experimental data for various defects sizes obtained by Choi et al. are tabulated in Table 4.2.

Table 4.2: Experimental details for the defect size configuration

Pipe No	Length of the defect (l) (mm)	Width of the defect (c) (mm)	Defect depth (t) (mm)	Burst pressure ( $P_b$ )(MPa)
DA	200	50	4.4	24.11
DB	200	50	8.8	21.76
DC	200	50	13.1	17.15
LA	100	50	8.8	24.30
LC	300	50	8.8	19.80
CB	200	100	8.8	23.42
CC	200	200	8.8	22.64

To verify the FEA, the quarter pipe 3D model with created with the rectangular pit with the dimensions of DB given in Table 4.2. The pipe's outer diameter(D) is 762mm, and the thickness(t) is 17.5mm. The hexagonal mesh was generated with a maximum element size of 5e-2 m. The mesh near the defects was refined to achieve mesh convergence. The symmetry boundary condition and internal pressure was applied as boundary conditions. When von mises stresses across the defect thickness reached 90 % of the ultimate strength was considered a failure, as shown in Figure 4.5.

The internal pressure increases until it comes to the failure criterion. The True stress-strain curve from (Choi et al., 2003) was used to feed plastic strain vs. true stress values in ANSYS, as shown in Figure 4.4.

Finally, the elastic-plastic finite element analysis is performed to predict the burst pressure. The failure criterion was reached when the internal pressure was 21 MPa in FEA, and there was a 3.5% error against the experimental failure pressure. Another case that was observed in (Choi et al., 2003) was when the modified ASME B31G solution was less conservative than the one produced by the FEA analysis. It is seen to occur for longer defects at higher depth-to-thickness ratios. We show a similar case in Figure 4.3, for a defect of length 12.55mm, width 25mm and depth 14mm. The depth of this defect is 80% of the pipe's thickness and has an interaction distance of 2.5mm. The FEM result shows a failure of the pipe at 22.33 MPa. The ASME result shows a resulting failure pressure at 23.16 MPa, which is marginally higher.

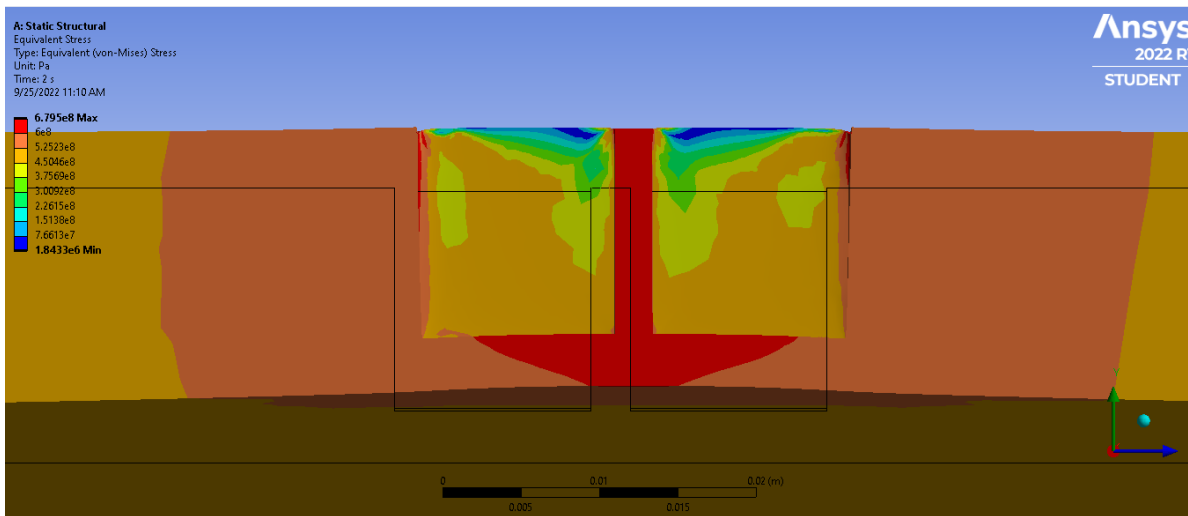


Figure 4.3: FEA simulation result where the limit load predicted by it is more conservative than the modified ASME B31G formula.

### 4.3.2 Parametric Study

After the evaluation of FEA, the parametric study was performed to compare the FEA burst pressure and ASME B31G-2009 semi-empirical formula (of Mechanical Engineers, 2012). The dimensions of the corrosion defects used for the parametric study were tabulated in Table 4.3. Instead of using a single corrosion defect, this paper focuses on the parametric study of the

interaction pits with various defect depths ( $d$ ) and interaction distances. But this interaction distance greater than three times the pipe thickness ( $3t$ ) is out of scope for this study. If the interaction distance between the two pits is less than  $3t$ , it will be considered a single pit, according to the ASME (of Mechanical Engineers, 2012).

Figure 4.6 shows that the ASME burst pressure decreases with increasing interaction distance between the pits, whereas the FEA burst pressure rises with increasing interaction distance. Because the closer the interaction distance, the higher the stress concentration. Hence, the pipe with closer interaction will fail earlier than the pipe with the farthest interaction pits. But ASME B31G averages out the interaction distance as overall defect length; it shows the opposite trend. In all the cases, the ASME B31G is more conservative than the FEA, with the disadvantage of pipe being underused than its full potential.

The tabulated dimensions of the pits were modeled and analyzed using the same procedure elaborated in the verification and validation section to predict the burst pressure of the pipe using Finite element analysis. The results produced by ASME B31G semi-empirical formula (1) and FEA are listed in Table 4.4.

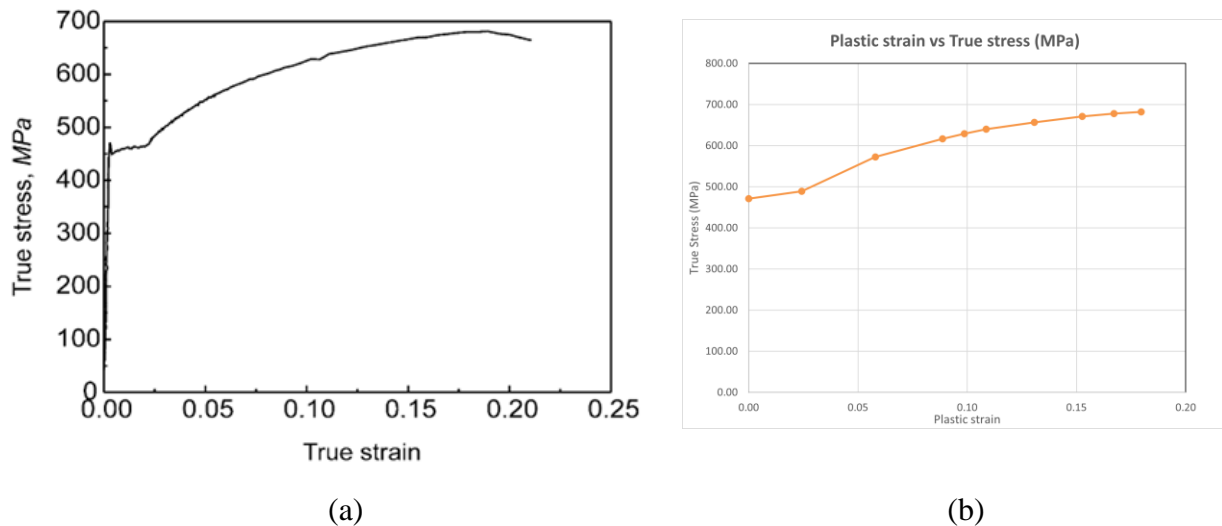


Figure 4.4: (a) True stress-strain curve for X65 (b) Plastic strain vs true stress in ANSYS



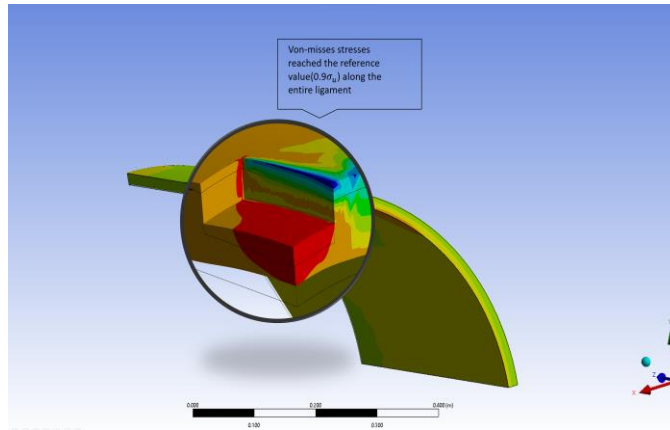


Figure 4.5: Von Mises Failure Criterion Across the ligament of the defect

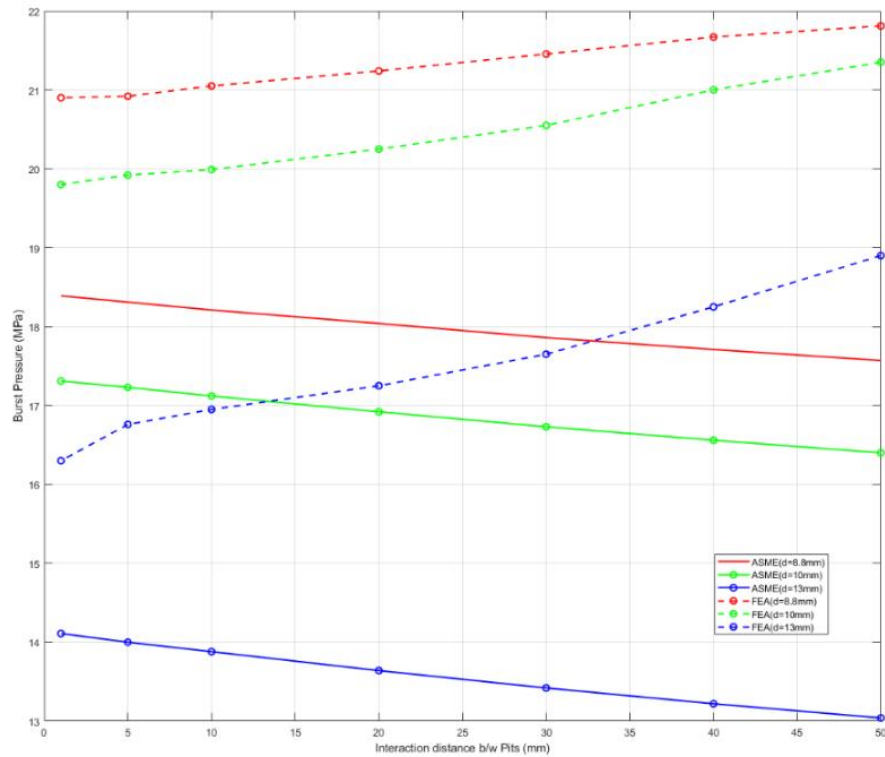


Figure 4.6: Interaction distance between pits vs burst pressure

Table 4.3: Summary of datapoints used for the parametric study

Case Name	Defect Length-1 (mm)	Defect Length-2 (mm)	Defect Width-1 (mm)	Defect Width-2 (mm)	Defect depth-1 (mm)	Defect depth-2 (mm)	Defect distance (mm)
$L_e250 d 8.8$	100	100	50	50	8.8	8.8	50
$L_e240 d 8.8$	100	100	50	50	8.8	8.8	40
$L_e230 d 8.8$	100	100	50	50	8.8	8.8	30
$L_e220 d 8.8$	100	100	50	50	8.8	8.8	20
$L_e210 d 8.8$	100	100	50	50	8.8	8.8	10
$L_e205 d 8.8$	100	100	50	50	8.8	8.8	5
$L_e201 d 8.8$	100	100	50	50	8.8	8.8	1
$L_e250 d 10$	100	100	50	50	10	10	50
$L_e240 d 10$	100	100	50	50	10	10	40
$L_e230 d 10$	100	100	50	50	10	10	30
$L_e220 d 10$	100	100	50	50	10	10	20
$L_e210 d 10$	100	100	50	50	10	10	10
$L_e205 d 10$	100	100	50	50	10	10	5
$L_e201 d 10$	100	100	50	50	10	10	1
$L_e250 d 13$	100	100	50	50	13	13	50
$L_e240 d 13$	100	100	50	50	13	13	40
$L_e230 d 13$	100	100	50	50	13	13	30
$L_e220 d 13$	100	100	50	50	13	13	20
$L_e210 d 13$	100	100	50	50	13	13	10
$L_e205 d 13$	100	100	50	50	13	13	5

Table 4.4: Comparison of the ASME B31G burst pressure and FEA burst pressure

<b>Case Name</b>	<b>ASME <math>P_b</math> (MPa)</b>	<b>FEA <math>P_b</math>(MPa)</b>
$L_e250 d 8.8$	17.57	21.81
$L_e240 d 8.8$	17.71	21.67
$L_e230 d 8.8$	17.86	21.455
$L_e220 d 8.8$	18.04	21.24
$L_e210 d 8.8$	18.21	21.05
$L_e205 d 8.8$	18.31	20.92
$L_e201 d 8.8$	18.39	20.9
$L_e250 d 10$	16.40	21.35
$L_e240 d 10$	16.56	21
$L_e230 d 10$	16.73	20.55
$L_e220 d 10$	16.92	20.25
$L_e210 d 10$	17.12	19.99
$L_e205 d 10$	17.23	19.92
$L_e201 d 10$	17.31	19.8
$L_e250 d 13$	13.04	18.9
$L_e240 d 13$	13.22	18.25
$L_e230 d 13$	13.42	17.65
$L_e220 d 13$	13.64	17.25
$L_e210 d 13$	13.88	16.95
$L_e205 d 13$	14	16.76

## 5 3D RECONSTRUCTION, SYSTEM INTEGRATION AND OPERATIONAL PROCEDURES

### 5.1 RGB-D 3D Reconstruction of a pipe using Simultaneous Localization and Mapping (SLAM)

3D reconstruction of the pipe was performed using RGB-D Graph-based SLAM approach called RTAB (Real-Time Appearance-Based) mapping. This technique is used to construct a 3-D map using the loop closure detection method. Loop Closure detection is the process of finding matches between current and previously visited locations by the RGBD sensor, which in our case was the Intel Realsense D435i. This approach has been integrated with Robot Operating Systems (ROS) as the “rtabmap\_ros” package.

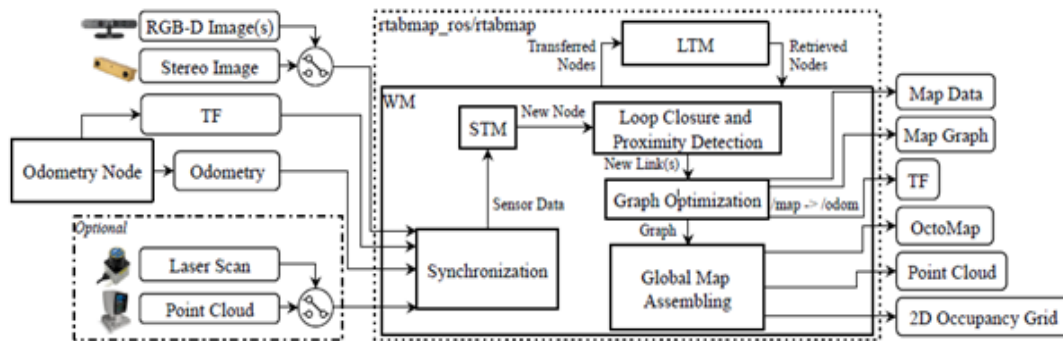


Figure 5.1: Overall system diagram for RTAB-Map (Labbé & Michaud, 2019)

The process of 3-d reconstruction is:

- The input to this algorithm is RGB-D or stereo images from the camera, odometry data, and optionally laser scan or point cloud from 2D or 3D lidar. In our case, RGB-D image and IMU (Inertial Measurement Unit) was used for odometry input. All the input data was provided to the RTAB module in a synchronized time frame.
- Then the pose information and the RGBD- data are stored in the STM (short-term memory) module. Since it is a graph-based map, it creates and stores this information in the form of a tree node. The nodes are created at the detection rate of 1 Hz, ensuring that the successive nodes overlap.
- The odometry information was taken from proprioceptive and non-propriocptive methods i.e., using visual odometry and the IMU in the D435 camera, respectively. Visual odometry

was calculated using the Frame-to-map approach. Frame-to-map registers the current frame against a local map of features from the last keyframes.

- Whenever a new image and its odometry information are received as a node in STM, a neighbor link is created between the consecutive frames. This ensures that whenever loop closure is detected, the graph optimization algorithm propagates the error to each connected node via links to decrease the odometry drift.
- Nodes are stored in STM till this buffer is filled with a range of 10 nodes before it is moved to WM (working memory). Working memory is the memory where the nodes are compared to make a loop closure if the nodes are similar.
- Nodes are transferred from WM to LTM (Long Term Memory) to ensure that the map update time does not become too slow. The parameters for this are set at the default states of 0ms (0ms implies indefinite time) for the update time threshold. To determine which nodes to be transferred to LTM from WM, a weight is initialized to the nodes as 0 when it enters STM and compares with the last node in the graph. If the neighboring nodes are similar, the weight is increased by a value of the weight of the last node plus 1. The oldest of the smallest weighted nodes is transferred to LTM. Whenever a loop closure is detected by comparing the nodes in WM, the neighboring nodes of the location in WM are brought back from LTM to WM for more loop closures. The output that we got in our task was the map data which had the map of the pipe obtained by rotation and translation of the camera.

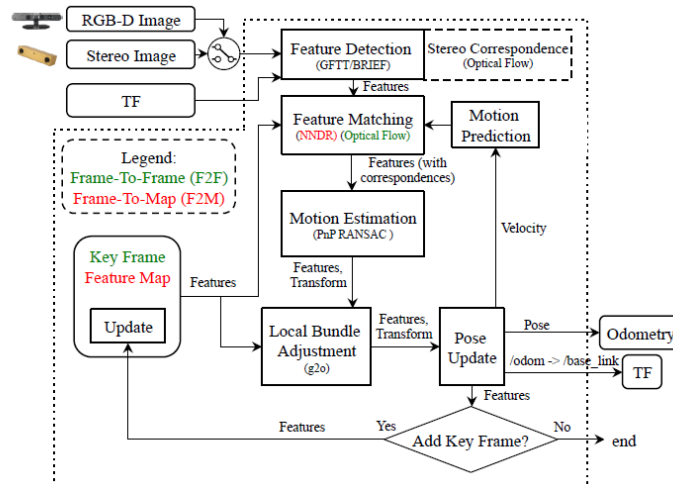


Figure 5.2: RTAB-Map SLAM: Visual Odometry system components (Labbé & Michaud, 2019)

The visual odometry was calculated using the following procedure:

The frame was processed based on GFTT (Good Features To Track) features (J. Shi & Tomasi, 1994; G. Zhang & Vela, 2015). Feature matching is done by NNDR ratio (Nearest Neighbor Distance Ratio) test using BRISK descriptors, and to increase accuracy, the NNDR is kept low at 0.2. Higher values will get more matching pairs, but the accuracy will reduce. A wrong match can cause the algorithm to fail in feature-poor environments with similar subregions. After the correspondences are calculated, the transformation is calculated using Perspective-n-point RANSAC. But the number of correspondences should be more than six to calculate the transformation. With the estimated transformation, the camera's pose is updated whenever a new node is created. Suppose the number of correspondences is less than 30% of the total number of features. In that case, the feature map is updated using local bundle adjustment, which reduces the reprojection error due to the lack of enough matching features.

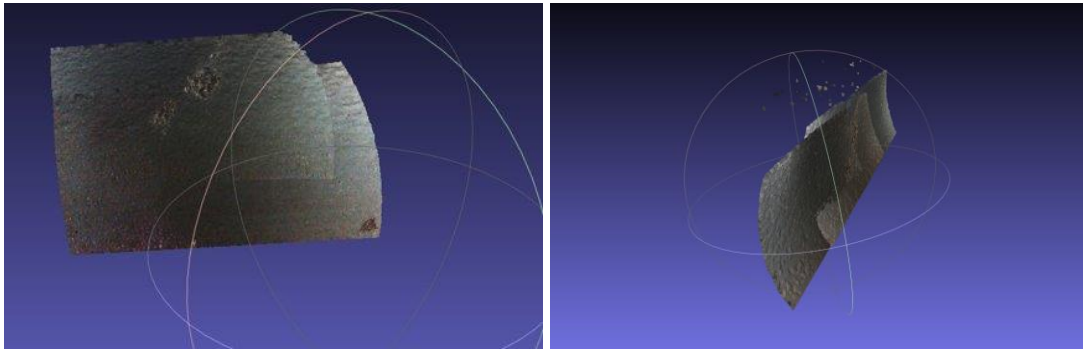


Figure 5.3: Frame stitching using SLAM in a feature-sparse pipeline environment

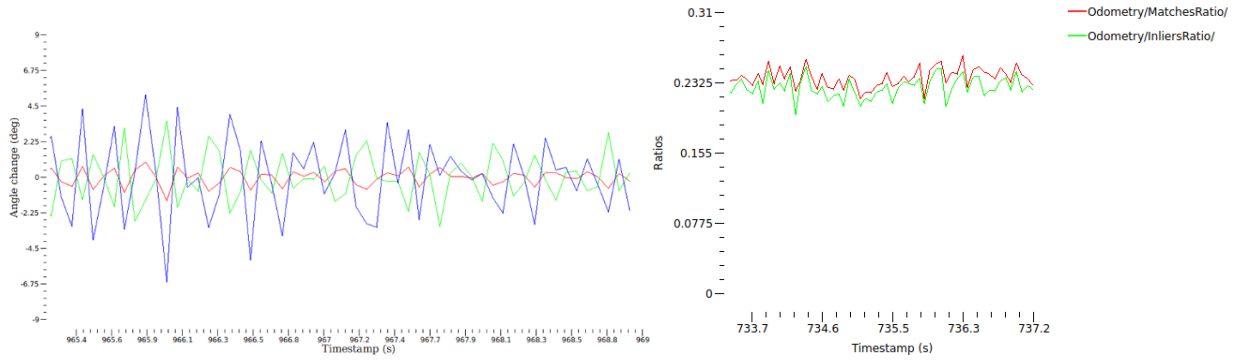
The demonstration above illustrates the challenge of obtaining an accurate 3D reconstruction when minimal distinguishing feature correspondences exist. The image on the left of Figure 5.3 shows a rotation, and the one on the right shows a translation. The problem encountered during our attempt at obtaining a reconstruction was the lack of enough inliers. Even if there were enough inliers for one frame, the odometry could not accurately predict the camera's motion due to the small number of features. Therefore, it caused the overlap and reprojection errors shown in Figure 5.3.

Next, a corroded steel pipe sample was acquired, which was used to perform an initial baseline analysis of the pipe scanning method when the constraints surrounding the earlier robotic platform were removed. The steel pipe sample, however, is less than 1m long, and detailed mapping quality analyses across time would ideally require longer samples. This would help understand the pipe

scanning method's accuracy and robustness to factors such as IMU drift. In addition, the plastic pipe has far fewer features than the corroded steel pipe, which posed a challenge to obtaining high-quality odometry consistently, as shown below. Therefore, for tests in the report, the features used were GFTT+ORB (J. Shi & Tomasi, 1994), and visual-inertial odometry using the frame-to-map method (Labbé & Michaud, 2019) was done.

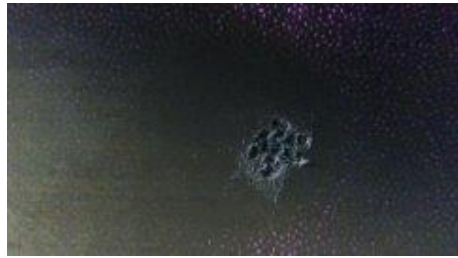
### **Test 1: Plastic pipe: Data and Discussion**

Two tests on the plastic pipe are described herein. The first test is a stationary holding test to capture the variability in the odometry, specifically the angular odometry in the presence of limited features. On clean sections of the plastic pipe, the results continued to be inconsistent despite the addition of external lighting and the stabilization of the camera position on the axis of the pipe using the stepper motor. Odometry tended to be lost easily, as the RTAB-Map package could not maintain a consistent series of matching frames. The camera view was then shifted to a location where there was an indication of a defect on the plastic pipe, as shown in Figure 5.4. In this view, the odometry became significantly better, with the feature matching algorithm consistently obtaining between 100 to 300 inliers after the iterative RANSAC processing of consecutive frames. The error in odometry is also lower than in the featureless case, resulting in no loss of odometry. However, there is some noise in this measurement, but this noise was not significant enough to cause poor odometry and trajectory outputs.



(a)

(b)



(c)

Figure 5.4: Demonstrative odometry measurements and inlier detection in real-time from the stationary test on the plastic pipe: (a) Angle data (b) Inlier ratio (c) Frame with defect features

The second test shows the limitations of the current technology for precise odometry in featureless environments and the effects of specular reflection on depth quality. This test was done by rotating the camera 360 degrees about the plastic pipe. The inliers were found to be inconsistent, and odometry was repeatedly lost in intervals of the pipe where there were few matches, as shown in Figure 5.5. External lighting also affects reflective surfaces' depth maps, as specular reflections can cause poor depth quality, which also leads to poor odometry results. The light source was mounted behind the robot to illuminate the entire pipe section to correct this. Despite modifying the light source, the odometry consistency did not improve.



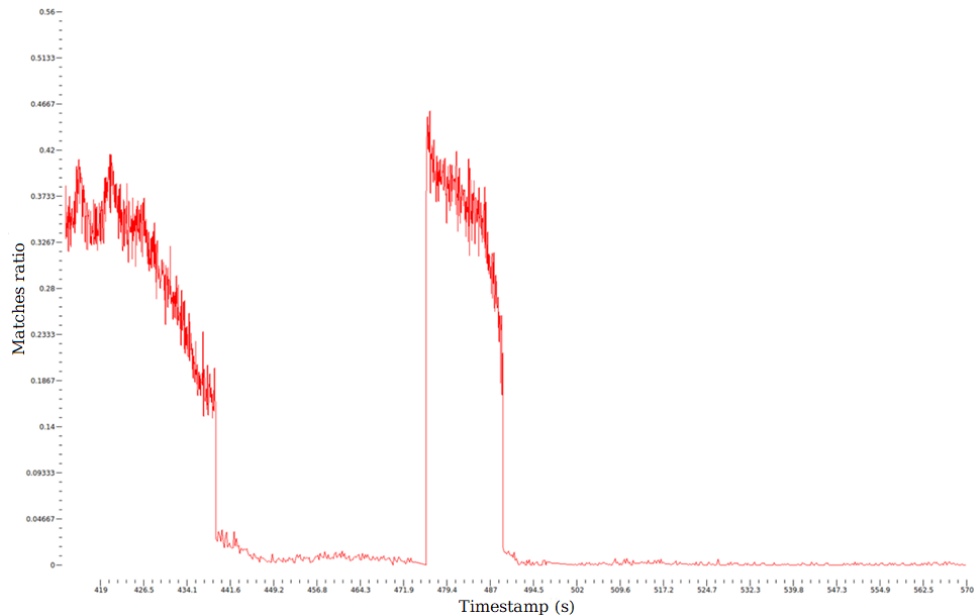
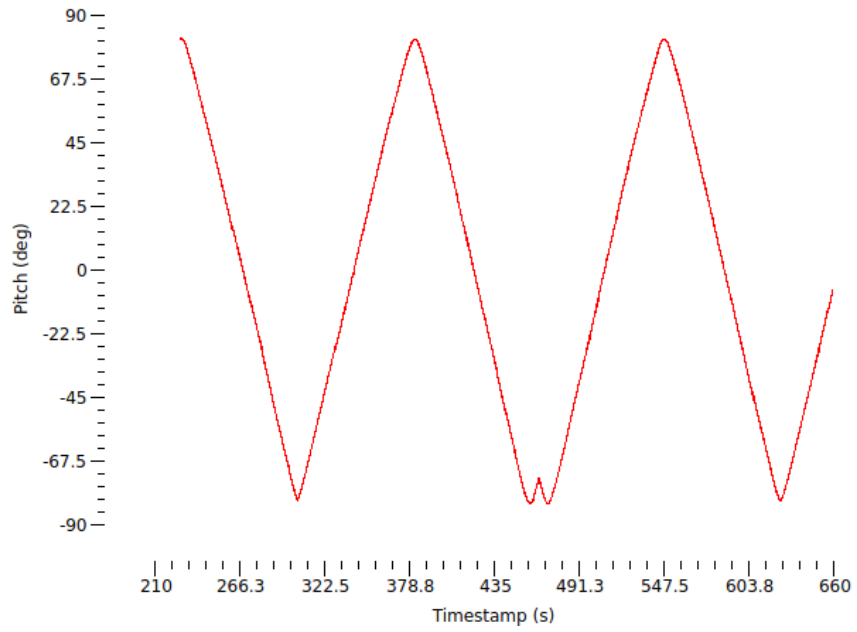


Figure 5.5: Rotation of the camera on the axis of the plastic pipe causes an intermittent loss of odometry due to the sparsity of feature matches between subsequent frames

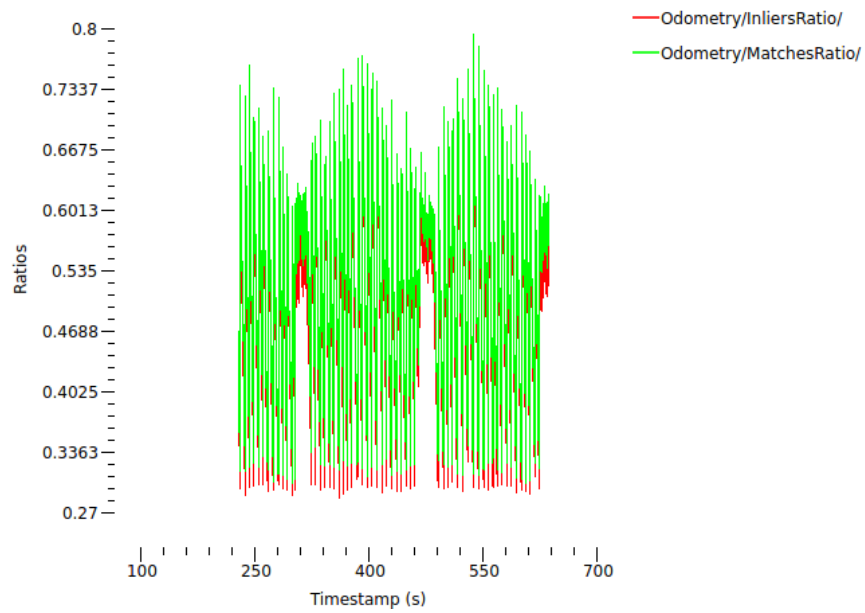
## Test 2: Corroded steel pipe: Data and Discussion

The stationary test for the steel pipe yielded more than twice the number of matches between subsequent frames, causing the odometry computation to be very stable compared to that seen in the plastic pipe. On the other hand, the pure rotation test yielded accurate odometry due to the high numbers of matches and inliers, as shown in Figure 5.6. The reason for the oscillation and variability in the inlier and match ratios is not determined yet. It needs to be investigated in the future within the context of the algorithms used in RTAB-Map.

The next test for the corroded steel pipe was a complete scan with both rotation and translation. This is demonstrated and attached as a video as part of the supplementary material. The scan speed was varied and speeds up to one full pipe revolution per minute were achieved in this quarter. It is important to note that the rotation speed of the stepper motor is inversely correlated with the quality of the odometry. Speeds greater than approximately one rev/min resulted in unreliable or poor-quality maps. A demonstrative image of the pipe traversal is shown in Figure 5.7.



(a)



(b)

Figure 5.6: (a) Pitch angle of the camera (b) Inlier ratios for a rotation cycle on the corroded steel pipe sample.

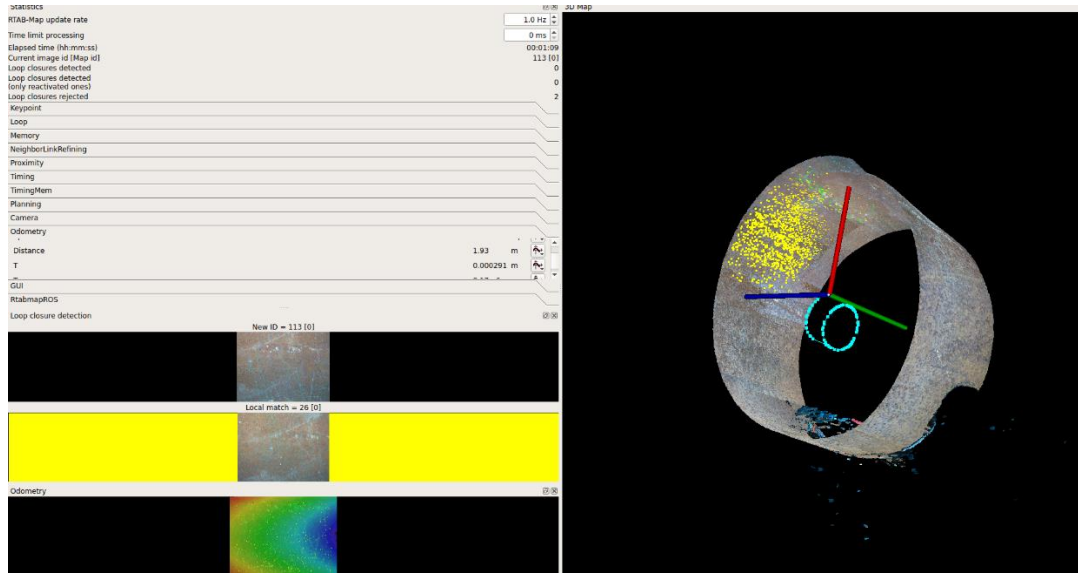


Figure 5.7: Demonstration of pipe scan with rotation of the camera about the axis of the pipe, and translation of the robot along the pipe. The trajectory of the camera system is estimated and shown in white

## 5.2 Orientation Estimation using a Kalman Filter versus a Complementary Filter

Let  $\alpha, \beta, \gamma$  be the pitch, roll, and yaw angles of a rigid body such as a camera, respectively. The motion sensor provides three gyroscopic measurements and three accelerometer measurements. The gyroscope provides the angular rate components, and the accelerometer provides the acceleration components of the sensor. To estimate orientation, the initialization of the orientation is done using the first measurements of the accelerometer as follows:

$$\mathbf{a} = \sqrt{a_x^2 + a_y^2 + a_z^2} \quad 5.1$$

$$\alpha_0 = \cos^{-1} \left( \frac{a_x}{\mathbf{a}} \right) \quad 5.2$$

$$\beta_0 = \cos^{-1} \left( \frac{a_y}{\mathbf{a}} \right) \quad 5.3$$

$$\gamma_0 = \cos^{-1} \left( \frac{a_z}{\mathbf{a}} \right) \quad 5.4$$

To avoid the accumulation of errors while computing orientation, and to fuse the sensors data from

both the accelerometer and the gyroscope, the complementary filter is used. The complementary filter is a combination of a high pass and a low pass filter, applied to the gyroscopic and accelerometer components respectively. The low pass filter is used on the accelerometer as it is sensitive to small accelerations in the short time scale that can be caused due to abrupt motions of the camera due to servo startup and vibrations during movement. Gyroscopes in the long-term can be prone to drift, which is an accumulating error. Therefore, the high pass filter on the gyroscope enables the use of these values in a shorter time window, and the low pass filter on the accelerometer reading makes use of it as an aggregate long-term measurement. For a general angle  $\theta$  and sensor measurements  $\theta_{accel}$  and  $\theta_{gyro}$ , and the filter parameter  $\phi$  is used to control the drift and the noise weighting levels. For this example, we take this value to be 0.995:

$$\theta = \phi\theta_{gyro}dt + (1 - \phi)\theta_{accel} \quad 5.5$$

The complementary filter produces results for a camera rotating 3 rad (180 degrees) and returning to its original position. These results are shown in Figure 5.8. The pitch rate is captured accurately, however there is an error of 35 degrees introduced in the yaw component and a smaller error of around 16 degrees (0.28 rad) in the roll component. To improve upon this, the Kalman filter is used in this quarter to compare the results to the complementary filter.

The Kalman filter is a predictor-corrector estimator, which models a general process  $x$  as follows:

$$x_k = F_k x_{k-1} + w_k \quad 5.6$$

$$x_k = [\theta_k, b_k, a_k]^T \quad 5.7$$

$\theta_k$  – 3x1 orientation error vector at step k

$b_k$  – 3x1 gyro bias vector, at step k

$a_k$  – 3x1 acceleration error vector at step k

$w_k$  – 9x1 additive noise vector

$F_k$  – state transition model

The prior and the state transition model starts out at zero as  $x_k$  is the error process. For the Kalman filter algorithm, the  $x_k$  state vector is only calculable indirectly from a measurement  $z_k$ , which is

linearly related to  $x_k$  as follows:

$$z_k = C_k x_k + v_k \quad 5.8$$

For the Kalman Filter, we obtain a 3x1 vector of differences in the sensor frame of reference between the gravity estimate from the gyroscope and the accelerometer. The objective of the Kalman filter is to compute an unbiased posterior estimate  $\hat{x}_k^+$  of the underlying process  $x_k$  from i) extrapolation from the previous iteration's posterior estimate  $\hat{x}_{k-1}^+$  and ii) from the current measurement  $z_k$ :

$$\hat{x}_k^+ = (1 - K_k)\hat{x}_{k-1}^+ + K_k z_k \quad 5.9$$

$K_k$  – Kalman gain

The Kalman gain is given by:

$$K = P_k^- H_k^T S_k^{-1} \quad 5.10$$

$$P_k^- = Q_k \quad 5.11$$

Here,  $H_k$  is the 3x9 true state observation model, which is a function of the 3x1 gravity vector estimated from the orientation of the sensor.  $S_k$  is called the innovation covariance, used to track the variability of the measurements, a function of the true state observation model  $H$ , the predicted apriori estimate of the covariance of  $H$  calculated in the previous iteration, and the covariance of the noise of the observation model.  $Q$  is the 9x9 error estimate covariance that tracks the variability in the state vector. For the demonstration, a simulated trajectory is used for the measurement data, as shown in Figure 5.9. This is a set of pure rotations on all three axes. Applying the Kalman filter to the incoming measurements from the IMU, the results are as follows, shown in Figure 5.10. The parameters of the Kalman Filter have been set empirically, so this needs to be studied further in the future. In contrast, the output from the complementary filter is less accurate and suffers from higher deviations in comparison to the Kalman filter.

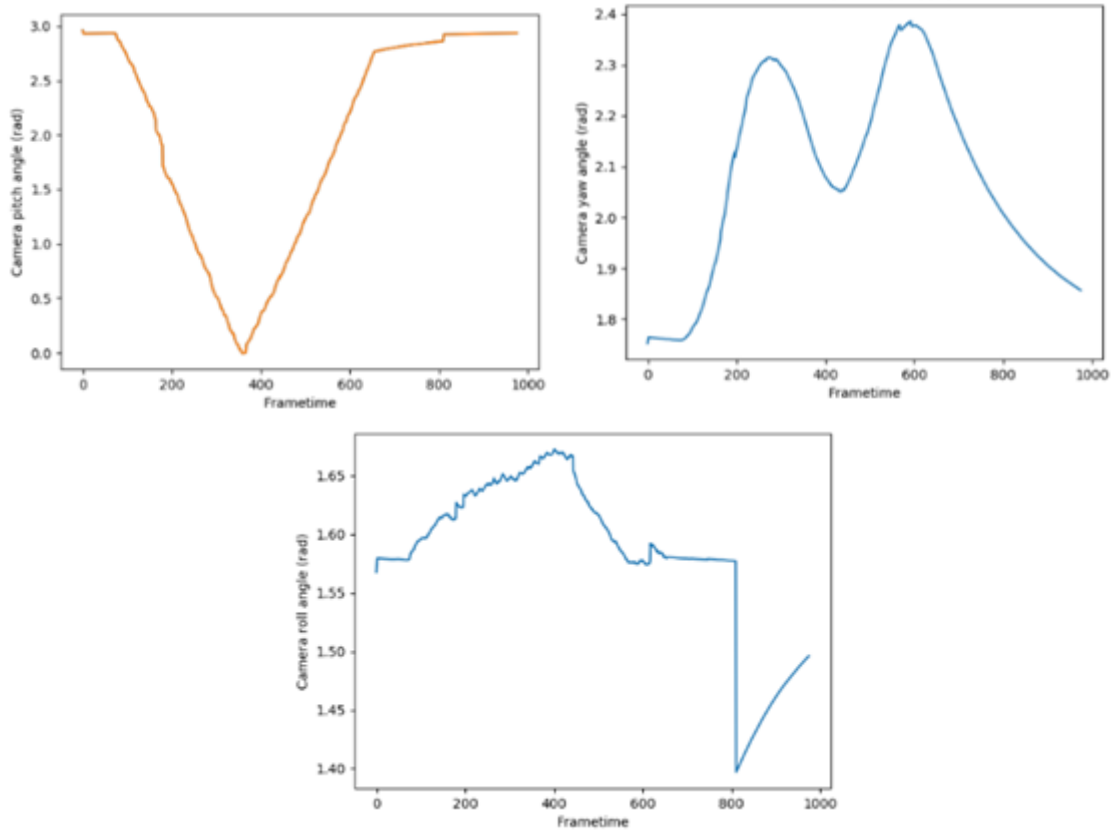


Figure 5.8: Pitch, yaw and roll angles computed using the motion module

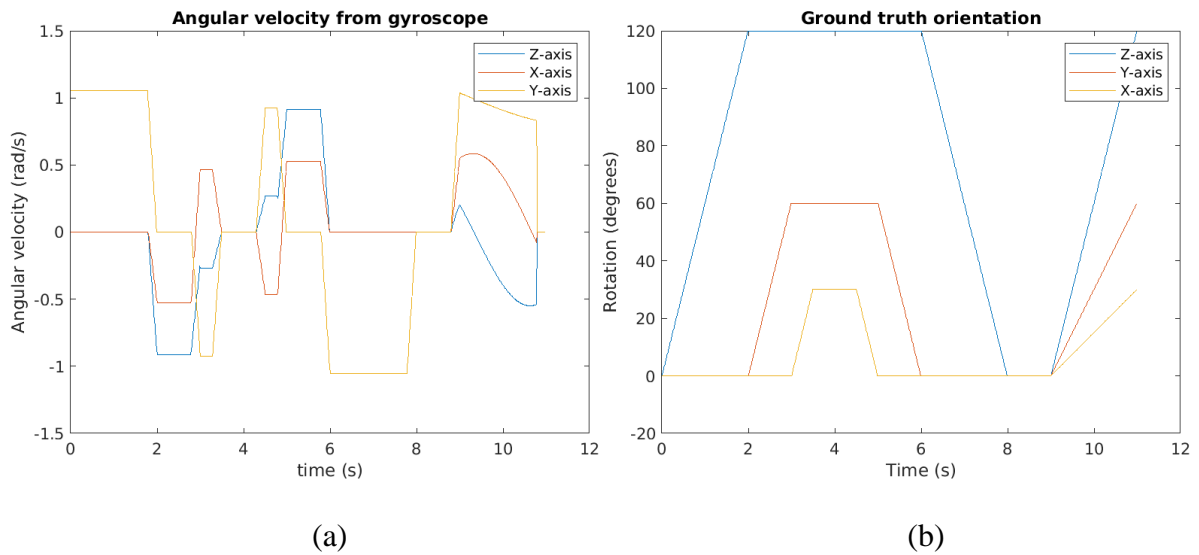
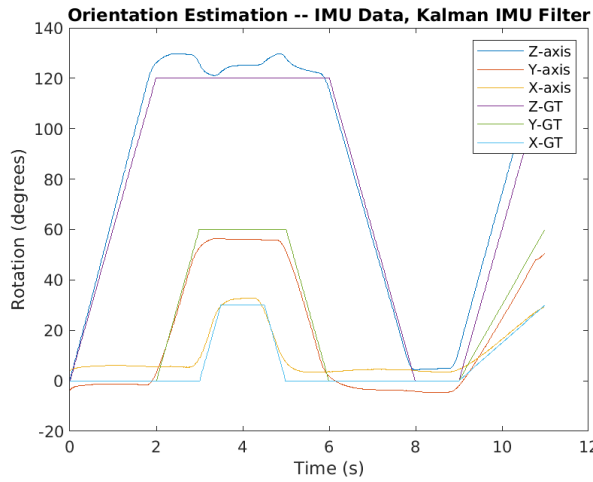
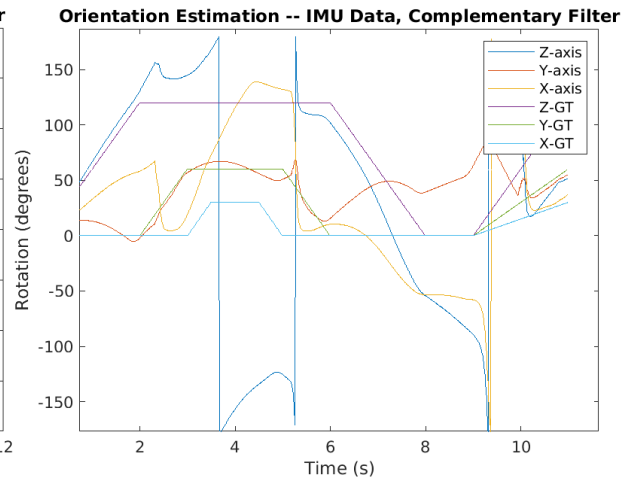


Figure 5.9: (a) Angular velocity measurement data from gyroscope (b) Ground truth orientation for the sensor



(a)



(b)

Figure 5.10: (a) Orientation prediction and ground truth using Kalman Filter (b) Orientation prediction and ground truth using Complementary Filter

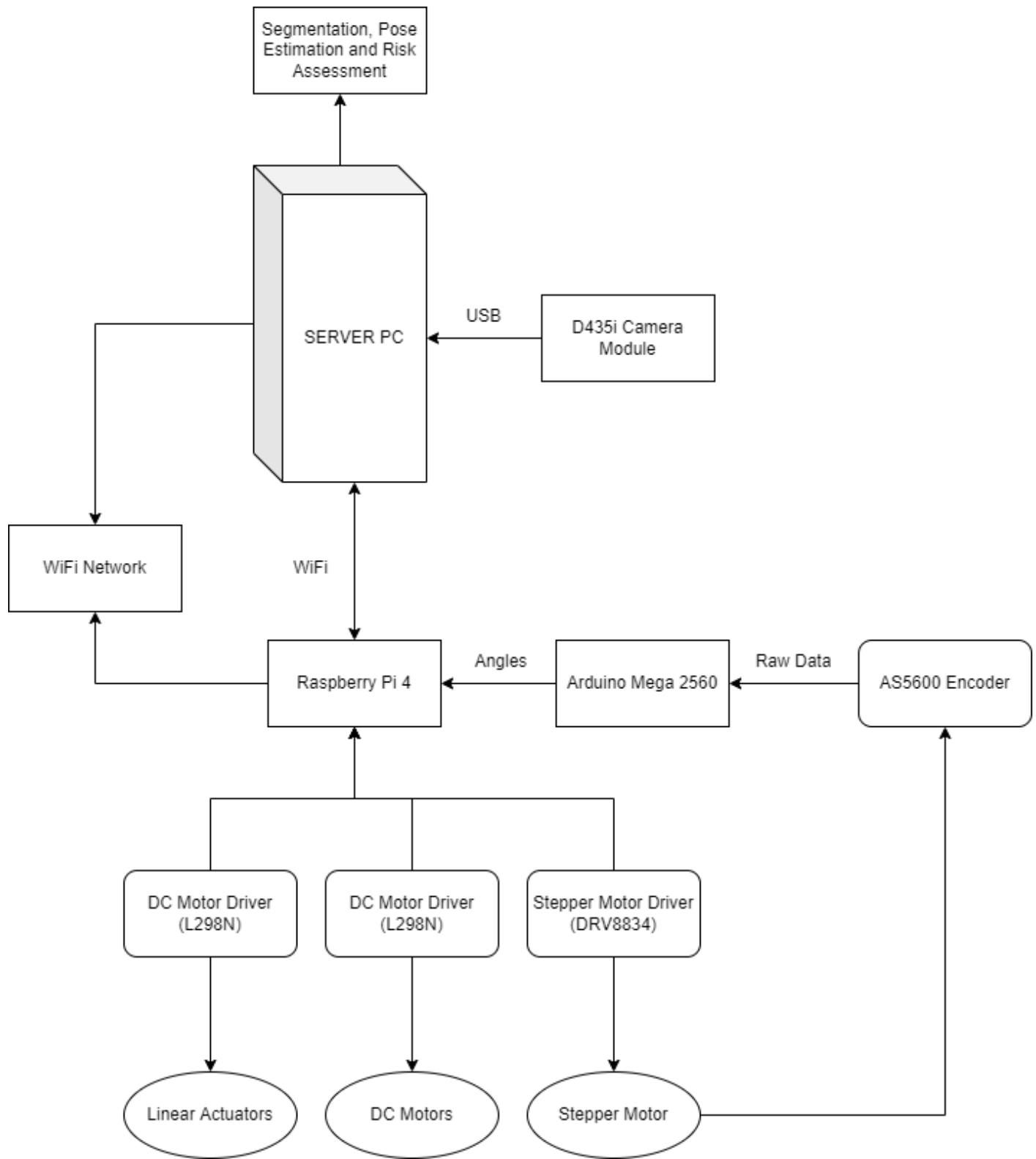


Figure 5.11: Flow chart of Hardware-Software Integration



### 5.3 Robot Operation Procedure

Figure 5.11 shows the overall flowchart of the hardware-software integrated prototype. We assembled the robot as described in the hardware specification in Chapter 3, section 2. The raspberry pi and the server computer are connected to the same network for the purposes of transferring the relevant stepper motor angle data. The imaging data is obtained directly from the D435i camera module, connected via USB cable to the server PC. We first obtain the RGB, X, Y and Z frame data, along with the IMU data and AS5600 stepper encoder data. These files are then processed offline for the downstream segmentation and risk assessment tasks.

The procedure to set up the robot is as follows:

- Power up the robot using the DC/Battery supply and connect the Raspberry Pi to the Wi-Fi network.
- Connect the server PC to the same Wi-Fi network as the Raspberry Pi.
- Under configuration settings, enable VNC server.
- Using the VNC Viewer, connect with the IP address of the Raspberry Pi, it will open the Raspbian OS
- In the Raspbian OS, under Bluetooth option, switch on and then connect with the Bluetooth of DualShock PS4 Wireless Controller.
- Open “hardwareIntegration.py” and run the program.

Now that the robot is ready to move, it is placed at the open end of the pipe. The working logic of the code is explained below as follows:

- The stepper motor's shaft axis is coaxial with the pipe's axis by altering the actuator's height.
- The robot is then moved inside the pipe up to a distance where the camera's field of view (FOV) covers the first axial section of the pipe.
- The camera is rotated clockwise to capture data from the imaging, motion, and stepper encoder modules. The camera records images at 30Hz and IMU data at 90Hz. This data is transmitted to the server computer synchronously using a blocking call implemented in the RealSense SDK named “wait\_for\_frames()”. The AS5600 rotary encoder installed at the backend of the stepper motor provides angle data to the server computer via the Arduino

and the Raspberry Pi using serial communication.

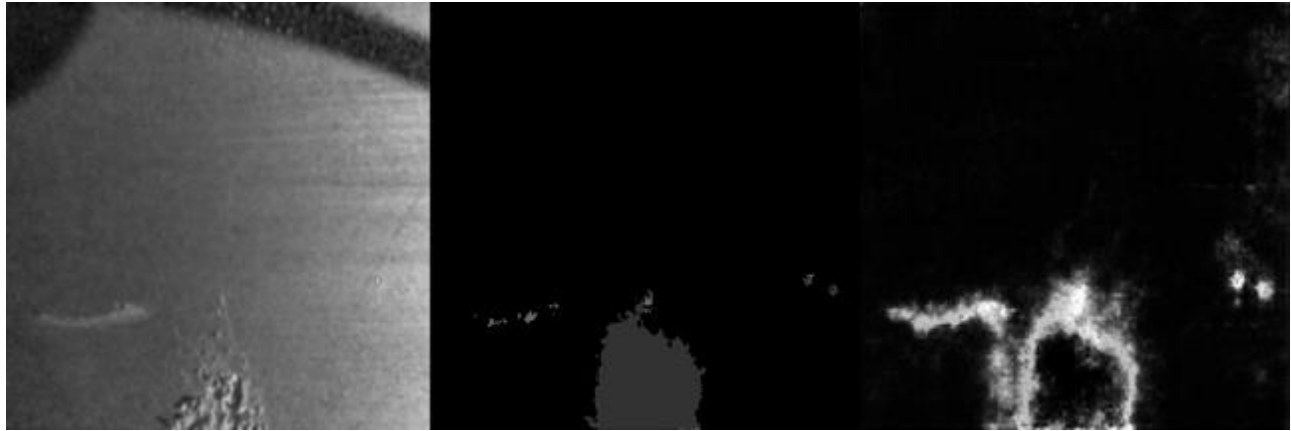
- After a full rotation cycle, the robot is moved to the pipe's next axial section, and the above process repeats.
- This is done until the entire pipe is covered.

#### **5.4 Integrating semantic segmentation and risk assessment**

The data acquired from the processes described in the checklist are ingested into the server computer and saved synchronously. Image data is then used sequentially using the PyTorch Dataloader class for inference. The obtained segmentation results are then used to compute epistemic and aleatoric uncertainty for the prediction as described earlier. The risk assessment module then computes a frame-wise prediction of remaining service life based on the largest defect detected by the segmentation network. The model for our prototype demonstration has been trained on the same pipe as the demonstrated example, owing to the lack of additional pipe samples for pitting defects. The testing distribution comes from the same distribution as the training set. The generalization performance of this model has been reported as part of task 2.2.

The demonstration shown in Figure 5.12 shows the integration results. The image is fed into the neural network sequentially to produce detections. The ASME B31G and NG 18 algorithms are then used to quantify the failure pressure associated with corrosion defects. The current implementation also includes the stepper motor encoder data, which provides us with an accurate estimate of the stepper angular position.

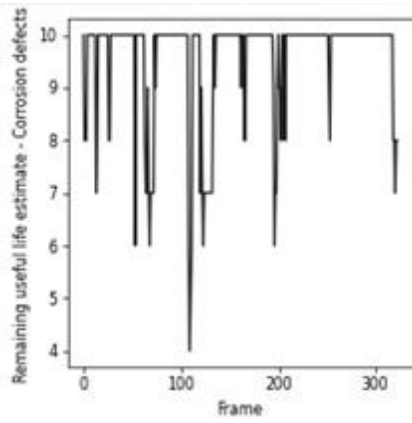
The final demonstration in this section would be an FEM analysis of the surface captured from the camera. The point cloud from the camera is saved as an STL file. The STL file was post-processed using ANSYS SpaceClaim. The imported facets were converted into a surface and then extruded to make a solid. This solid was then used to create a 3D pipe model with ground truth shown in Figure 5.13. Program-controlled non-linear meshing with mesh refinement near the corrosion pit was performed. The mesh was generated with 95665 nodes and 58842 elements. The von mises stress was recorded when an internal pressure of 35 MPa was applied. Figure 5.13 shows that the peaks in the Von-Mises stress occurs around the surface defects.



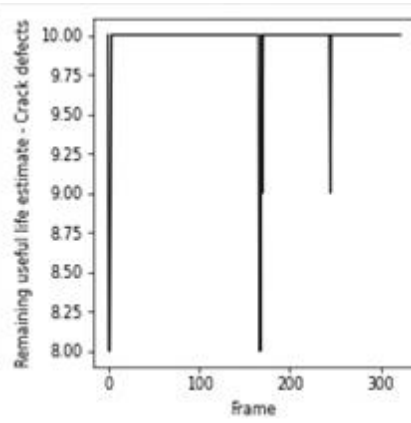
(a)

(b)

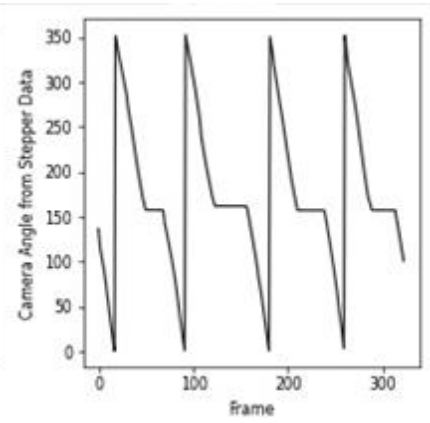
(c)



(d)



(e)



(f)

Figure 5.12: Demonstration of inspection in a sample pipe with pit and crack defects. (a) Image (b) Detection (c) Uncertainty (d) Remaining Useful Life estimate– Corrosion (e) Remaining Useful Life Estimate – Crack (f) Camera Orientation from Stepper Encoder

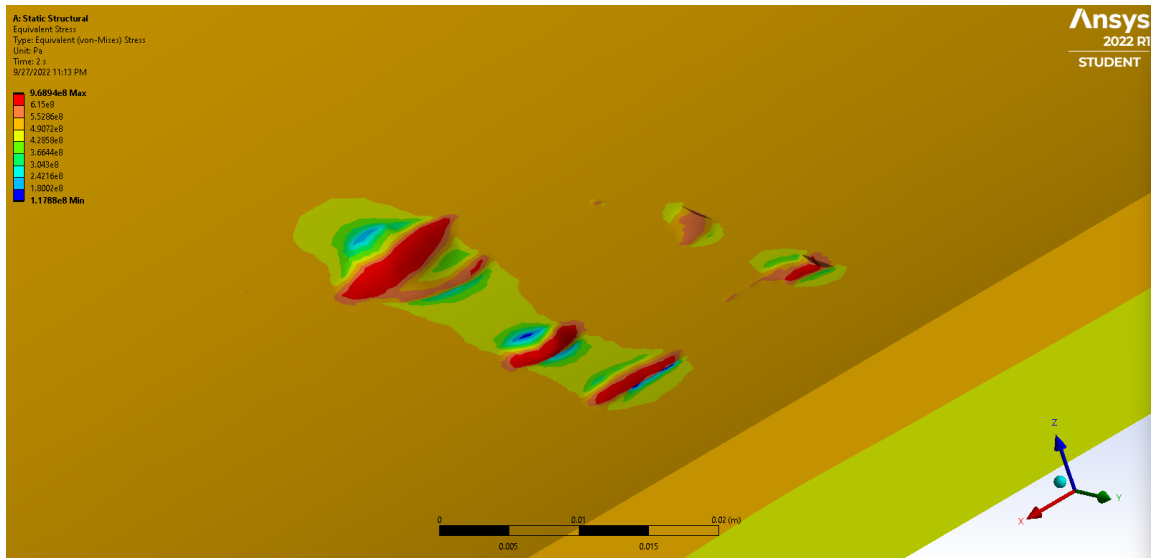
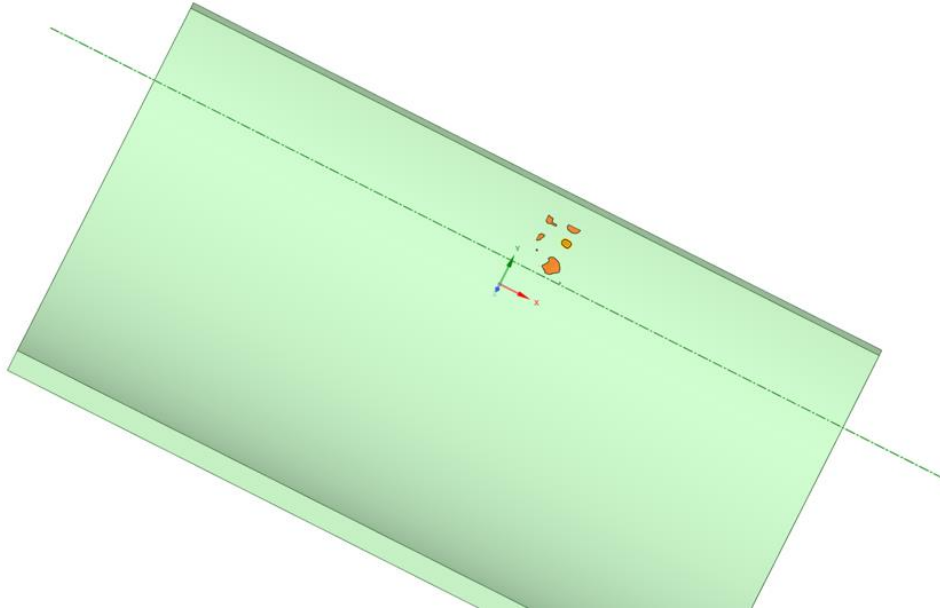


Figure 5.13: Converted model in ANSYS and Von-Mises stress around the internal defect

## 6 CONCLUSIONS

This report has summarized the contributions made toward prototyping a mobile robot that performs data-driven evaluations of pipeline surfaces for defect detection, quantification, and downstream risk assessment.

The first contribution of this project was to evaluate a commercially available stereo-vision sensor for in-line inspection purposes. Analyses were performed on the sensitivity of parameters to depth image resolution, and it was determined that the camera could capture defects in pipes up to 12 inches in diameter using disparity shift adjustments. The depth resolution was at 1mm, with signal degradations below this limit, making the sensor unreliable for sub-millimeter depth evaluations. The area resolution was sub-millimeter, making the sensor suitable for measuring small pits and narrow crack widths if required. The project also designed the prototype to utilize one camera for inspection instead of multiple cameras, resulting in significant weight reduction for the prototype device and simplified downstream data analyses. The prototype design incorporates various off-the-shelf electronics and drivers to achieve remote control of camera position and robot translation.

The second contribution of the project was the proposal and evaluation of a fully supervised semantic segmentation network with heteroscedastic uncertainty estimates using a sampling-based approach, the MC-dropout. Dropout provides an easily integrated means of evaluating the model parameters' epistemic uncertainty; however, this work's additional contribution was to separate the epistemic uncertainty and the aleatoric uncertainty by highlighting the effect of adding additional image data on reducing predictive uncertainty.

The third contribution of this project was the proposal and evaluation of a semi-supervised semantic segmentation technique based on the principles of interpolation and consistency regularization. The lack of enough training data for this domain makes weakly-supervised and semi-supervised learning techniques a crucial part of model development cost reduction. The proposed model's penalties on performance with equivalent fully supervised machine learning, and deep learning models in the literature were evaluated. The results have indicated a 20% reduction in performance compared to fully labeled datasets, with only 5% of the data being labeled for the semi-supervised case.

The fourth contribution of this project was integrating existing semi-empirical methods into the

defect detection workflow. One of the disadvantages of the implemented ASME B31G model is that the predictions for maintenance scheduling are overly conservative. Therefore, a FEM-based surrogate model with training data was used to compare the results of remaining useful life with the semi-empirical ASME B31G. The surrogate model was shown to predict failures earlier than the semi-empirical model, as it considered situations with two interacting defects: cracks and corrosion pits, compared to the ASME/NG-18 model, which considers these threats separately. Finally, we compare the results of the ASME B31G model against FEA analysis for interacting corrosion pit defects. The results indicated that the ASME B31G model was more conservative for all cases except for very deep pits, where the ASME B31G model was marginally less conservative.

## 7 APPENDIX A

### 7.1 Empirical evaluation of stereo-matching parameters for the Intel RealSense D435i Camera

The camera used in this project is the Intel Realsense Camera D435i. Intel RealSense Vision Processor acquires raw image streams from the depth cameras and computes high-resolution 3D depth maps without needing a dedicated Graphics Processing Unit (GPU) or host processor. It has a right imaging camera, left imaging camera, infrared (IR) projector, and an RGB module. The stereo camera projects an infrared light pattern onto the scene to increase the texture of the low-texture scenes. The benefit of using IR is that it can work on any lightning conditions for perceiving depth. The depth depends on the spacing between the two sensors (baseline)- the wider the baseline, the farther the depth computation. Figure 7.1 shows the Intel Realsense Depth camera with its visible parts, such as IR module, left and right cameras, and the RGB module.



Figure 7.1: Intel Realsense D435i camera system (Intel RealSense, 2022)

In this section, we report the results of empirical sensitivity analysis. This is performed to evaluate the robustness of the depth map to changes in camera parameters and to determine the parameter range at which the depth map most closely resembles the real-world object dimensions. Bad sensor parameters can lead to object boundary bleeding, missing values (holes) in the depth map, and noisy output.

The considered parameters are:

- Exposure ( $\frac{W}{m^2} s$ ): the amount of light per unit area reaching the surface of an electronic image sensor.
- Gain (dB): Amplifies the entire image signal
- Laser Power (W): Power provided to the IR projector

- Second Peak Threshold: Determines how different the second-highest matches can be from the first highest matches for stereo depth computation
- Neighbor Threshold: Determines the number of neighboring pixels to be considered in the left image, which will be compared with the right image for depth computation
- Disparity Shift (Min-Z and Max-Z change in Pixels): Controls the modification of the Z-min and Z-max values for the camera to visualize.
- Resolution: The fineness of detail in an image measured in pixels per inch (ppi)

We first analyze how these key parameters influence object area and then analyze how they influence object depth computation.

### ***7.1.1 Sensitivity analysis of object area measurements to the camera parameters***

#### **7.1.1.1 Methodology**

Contour extraction extracts the region of interest containing the detected object boundary from the depth map. Contours are extracted by joining pixels with the same color or intensity. This computationally cheap approach can detect specific shape features in the image. Contours are derived from traditional computer vision techniques like edge detection and boundary detection. A boundary/edge is detected based on pixel difference in luminosity, texture, and perceptual grouping from the background. Mathematically, the edge/boundary detection is performed by convolving with local filters such as Sobel, Prewitt, Laplacian, Canny, etc., and then finding the pixels with the highest gradient magnitude with respect to the local neighborhood pixels.

Prior to performing contour detection, hole filling is performed on the depth map using the inbuilt Pyrealsense hole filling algorithm. Canny edge detection is performed on the image whose thresholding is determined using the pixel histogram.

Canny edge detection is done in stages. Noise is removed from the image using 5x5 Gaussian Filter. The image is then filtered using the Sobel operator, which finds derivatives in the horizontal direction ( $G_x$ ) and vertical direction ( $G_y$ ). From there, we get the edge gradient and angle of the pixels.

$$G = \sqrt{G_x^2 + G_y^2}$$



$$\theta = \text{atan}\left(\frac{Gy}{Gx}\right)$$

Then, the non-max suppression operation is done on unwanted pixels for the edge. In the end, the threshold value provided to the filter is used to find the correct and unwanted edge pixels.

After canny edge detection, morphological dilation is performed on the edges to ensure continuity. Contour detection is then performed on the dilated image to find the boundary points. The boundary points are a discrete set of pixel coordinates, and a curve is generated through these points to create the best approximation. Finally, image moments from the contour are used to find its centroid, which is used as a reference depth measurement.

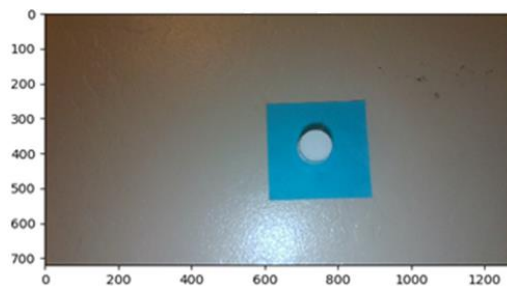


Figure 7.2: Original RGB image

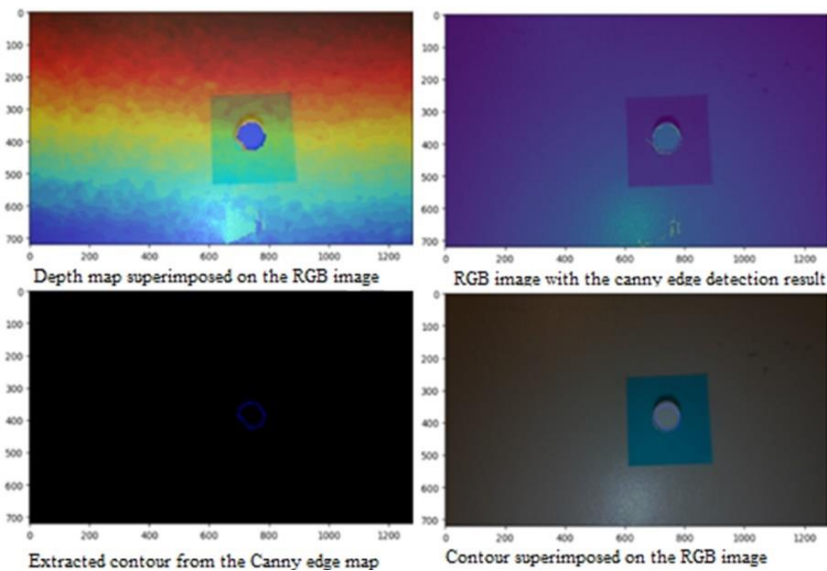


Figure 7.3: Contour extraction from the depth map

Figure 7.2 shows the original RGB image captured by the camera. In Figure 7.3, the upper left image shows the depth map after hole filling superimposed on the RGB map for comparison. The upper right image shows the Canny edge detection from the depth map superimposed on the RGB

map for comparison. The lower left image shows the contour extracted from the dilated version of the canny edge detected map. For comparison, the lower right image shows the contour superimposed on the RGB image.

The RGB image is first converted into grayscale, and then the pixel histogram is used to threshold this grayscale image into a binary image. The binary threshold segments out the region of interest from the background. The experiment was set up so that the region of interest could easily be segmented using classical thresholding operations. Finally, the binary image is used to find the contours. Figure 7.4 (a) shows the original RGB image. This image is overlaid with the extracted contour in Figure 7.4 (b), where the green contour line represents an approximation to the region boundary of the object of interest. In this method, the ground truth area was measured manually, and the area of the depth map was converted from the pixel space into the world coordinate system using the camera's intrinsic parameters. The camera intrinsics do not provide an accurate field of view. It always has an error in the range of (-3,3) degrees. Thus, the field of view was calculated from the focal length and frame dimensions as in Equation 7.1 and Equation 7.2.

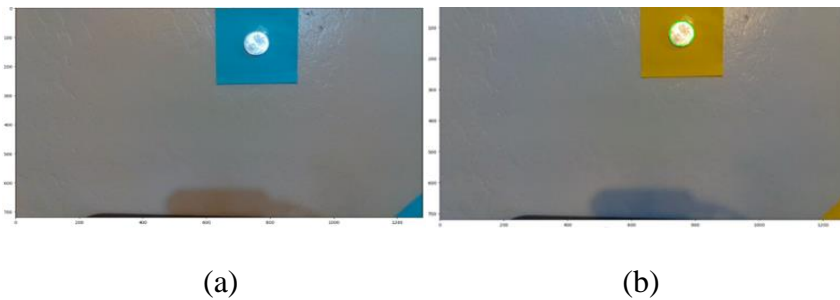


Figure 7.4: (a) Original RGB image of a coin (b) Image with overlaid contour

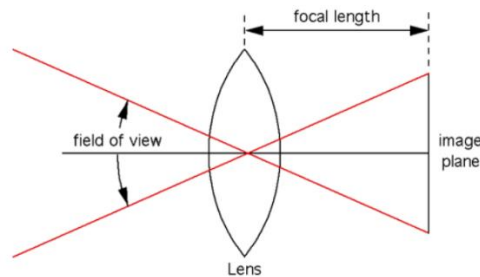


Figure 7.5: Geometry of a camera lens

Figure 7.5 shows the basic geometry of a camera lens.

$$HFOV = 2 * \tan^{-1} \left( 0.5 * \frac{W}{f} \right) \quad 7.1$$

$$VFOV = 2 * \tan^{-1} \left( 0.5 * \frac{H}{f} \right) \quad 7.2$$

Focal length is calculated from the depth camera intrinsic parameters, and width and height are the frame width and height in pixels. The depth coordinate system is orthogonal with the origin and orientation of the depth camera. We compute the height and width of 1 pixel as shown in Equations 7.3, 7.4 and 7.5.

$$W(1px) = \frac{depth * \tan \left( \frac{HFOV}{2} \right)}{FrameWidth} \quad 7.3$$

$$H(1px) = \frac{depth * \tan \left( \frac{VFOV}{2} \right)}{FrameHeight} \quad 7.4$$

$$Area(mm^2) = Area(Px) * W(1px) * H(1px) \quad 7.5$$

$Area(mm^2)$  represents the area of the object (area of the cap in this experiment) in world coordinates and  $Area(Px)$  represents the area in pixels.  $W(1px)$  and  $H(1px)$  represents the width and height of 1 pixel in world coordinates. The pixel area  $Area(Px)$  is obtained by computing the 0<sup>th</sup> order image moment  $M_{00}$  on the binary image, for the region corresponding to the specific contour in consideration:

$$M_{00} = \sum_{i,j} x^i y^j I(x, y) \quad 7.6$$

where  $x$  and  $y$  are the pixel index locations and  $I(x, y)$  is the binary intensity of the pixel. This effectively computes the number of pixels inside the contour. To verify whether the algorithm works as expected and provides area computation estimates close to the ground truth, we first apply it to an RGB image, where the contour estimate of the object overlaps very closely with the object boundary. This is done because the parameter estimation on the depth map would lead to

changing object boundaries in the depth map. Once we verify that the area calculation method returns accurate values with respect to the ground truth, we apply it to the depth map. The measured contour area came out to be  $4.44\text{ cm}^2$  using the above calculations. The ground truth area is  $4.84\text{ cm}^2$ , which leads to an error of 8%. The calculated fields of view, horizontal and vertical, are  $69.2^\circ$  and  $42.4^\circ$ , respectively. This experiment was performed to show that the proposed algorithm for conversion of the area in pixels to world coordinates metrics ( $\text{mm}^2$ ) gives consistent area in  $\text{mm}^2$  by calculating the area of the same object from the depth map at varying distances from the camera.

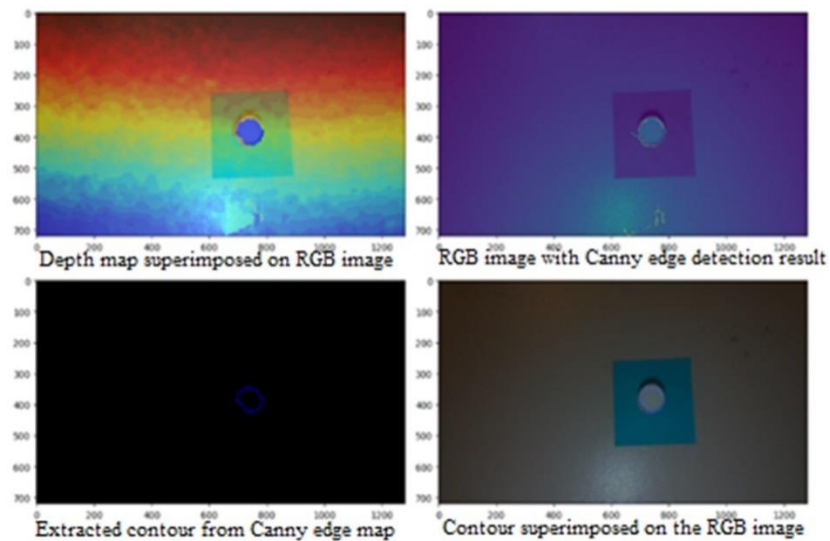


Figure 7.6: Contour area calculation for depth map

Figure 7.6 shows a demonstrative example of contour extraction from the depth map at a distance  $d$  from the wall. The parameters were set by visual inspection such that the quality of the depth map was reasonable. The distance of the camera from the wall was then varied. Table 7.1 shows that even though the camera's distance from the object varies, the area in  $\text{cm}^2$  is the same. There is a slight variation due to the sensor's temporal noise. The ground truth area for this image is  $379.94\text{ mm}^2$ . The measured area of the contour from the depth map was approximately  $32\text{ mm}^2$ , and the error was 15%.

Table 7.1: Contour area at various distances in pixel coordinates and world coordinates

Distance in cm	Area of contour in pixels	Area of contour in $mm^2$
23.1	5268	327.7
24.4	4686	325.23
27.2	3718	320
28.1	3545	326
29.9	3116.5	324
30	3082	323

Since the area error needed to be checked against a wide range of parameter values for any parameter, capturing the image of an object (a bottle cap in this experiment) and uniform variation of the parameters was automated using a python script. For every parameter value, the contour area was calculated in the world coordinate system(cm), and error was found with respect to the ground truth area. The ground truth area was measured using a measuring tape.

Ground Truth Area:  $379.94 \text{ mm}^2$

Error percent is calculated as:

$$Error(\%) = \left( \frac{Area_{calculated} - Area_{GroundTruth}}{Area_{GroundTruth}} \right) * 100\% \quad 7.7$$

### 7.1.1.2 Results and Discussion

The resultant image of the contours formed around the object is displayed by superimposing the contour extracted from the depth map on the original RGB image for easy visual comparison.

#### Error in area of the cap with respect to change in gain (dB)

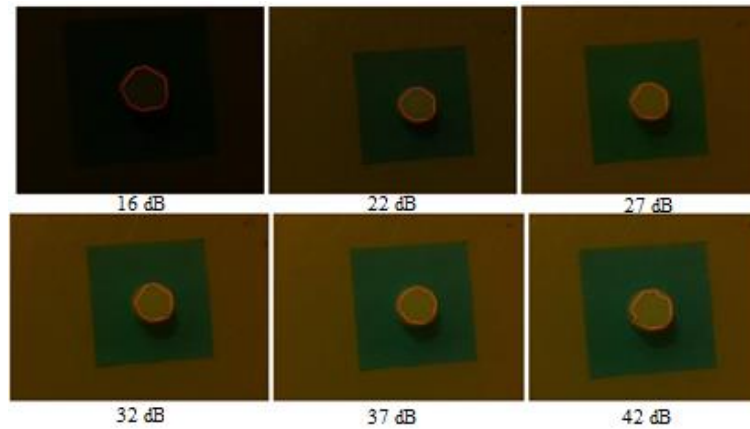


Figure 7.7: Representative contours for various values of gain.

Row 1 (Left to Right): Gain (dB) = [16, 22, 27]

Row 2 (Left to Right): Gain (dB) = [32, 37, 42]

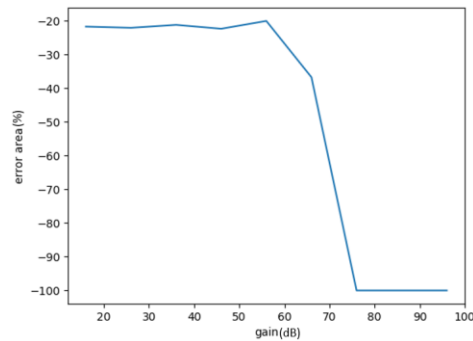


Figure 7.8: Contour area error variation with respect to camera gain

The gain is the amplification of the image signal obtained from the image sensor. In Figure 7.7, the resultant images show contours extracted from the depth map superimposed on an RGB image for comparison. The images are obtained while varying the camera gain value.

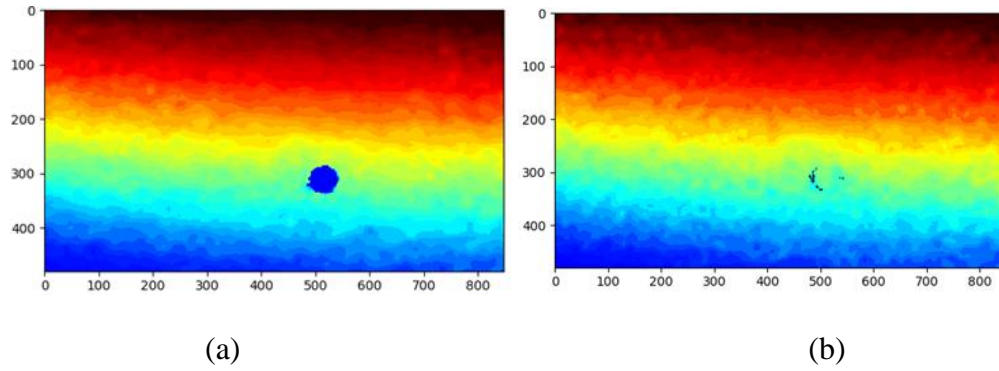


Figure 7.9: Depth frame for gain in range (16-55 dB)

From Figure 7.8, it can be concluded that the depth map performs very well in the low gain range (16-55). More gain introduces noise, which is the reason for depth map distortion for gains above 55dB. Please note that any error beyond 100% was clamped for better readability. Figure 7.9 (a) shows the depth map after hole filling for gain in range 16-55 dB, and Figure 7.9 (b) shows the depth map after hole filling for gain outside the range 16-55 dB.

### **Error in area of the cap with respect to change in Exposure**

Exposure is the amount of light per unit area reaching the surface of an electronic image sensor. It depends on how long the camera shutter is opened for the light to reach the camera. It is measured in  $(\frac{W}{m^2} s)$ . A high setting for exposure in a bright scene would lead to an overexposed image, where the fine-grained details of the image are lost. Similarly, a low setting for exposure in a dark scene would not give the sensor sufficient time to capture the salient features in the scene. Figure 7.10 shows the contours extracted from the depth maps when the exposure was changed from 0-25000  $(\frac{W}{m^2} s)$ . The depth map on the first two images shows a contour that does not match the object. This is because the camera's exposure was too low to produce the object's shape, as shown in the last 4 images. On the other hand, exposures greater than 50,000  $(\frac{W}{m^2} s)$  makes the image overexposed, which also leads to poor detection and area computation.

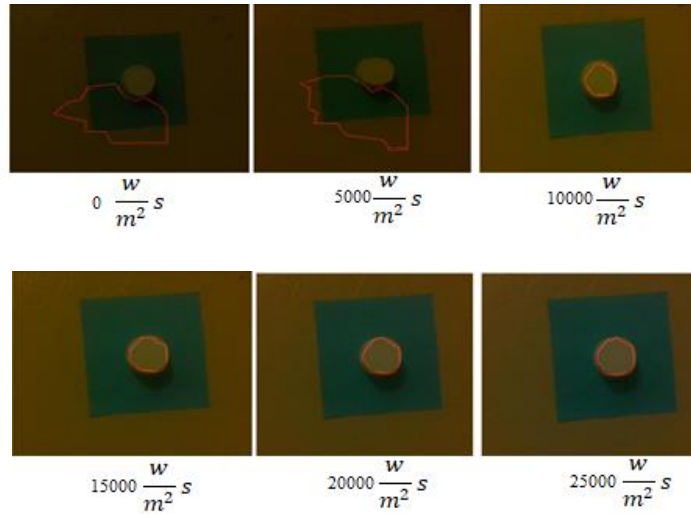


Figure 7.10: Representative contours for various values of gains

Row 1 (Left to Right): Exposure ( $\frac{W}{m^2} s$ ) = [0, 5000, 10000]

Row 2 (Left to Right): Exposure ( $\frac{W}{m^2} s$ ) = [15000, 20000, 25000]

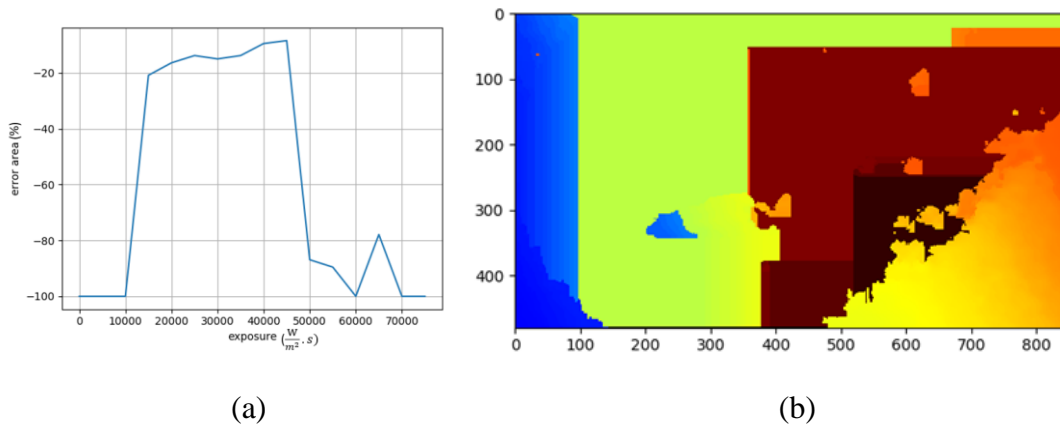


Figure 7.11: (a) Contour area error variation for varying exposure  $\frac{W}{m^2} s$  (b) Distorted raw depth frame for exposure  $> 45000 \frac{W}{m^2} s$

From Figure 7.11 (a), it can be concluded that the depth map gives a reasonable estimation of the area when the exposure lies in the range 10000-45000 ( $\frac{W}{m^2} s$ ). However, exposure depends on the light incident on the camera. More light creates a brighter image for both left and right cameras, which loses the depth of information. Similarly, the image is too dark for less light, which again leads to loss of depth information. So, it works only in a particular range depending on the external



lighting conditions. Figure 7.11 (b) shows the distorted depth map obtained when the exposure is greater than  $45000 \left(\frac{W}{m^2} s\right)$ .

**Error in area of cap with respect to change in Laser Power**

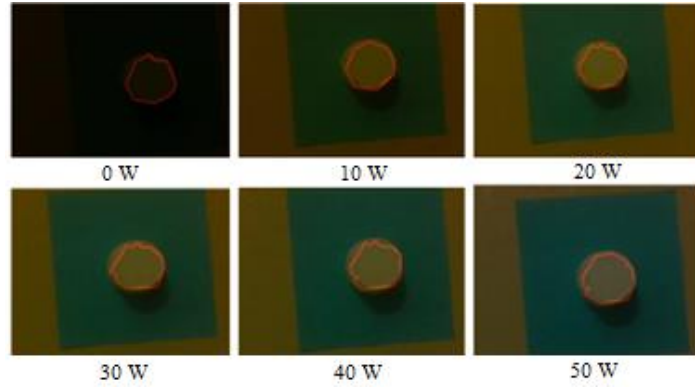


Figure 7.12: Representative contours for various values of Laser Power

Row 1 (Left to Right): Laser power (W) = [0,10,20]

Row 2 (Left to Right): Laser power (W) = [30, 40, 50]

Laser power is the power provided to the IR projector. It is measured in Watts (W). Figure 7.12 shows the contours extracted from the depth maps when the exposure was changed from 0-50 W. Figure 7.13 shows that the contour error variation is low for the entire range of laser power. Laser power is required for long-range viewing and viewing in low texture and low brightness environments. Since the object was at a constant distance and under sufficient lighting conditions, there was no strong effect in area computation when the laser power was varied.

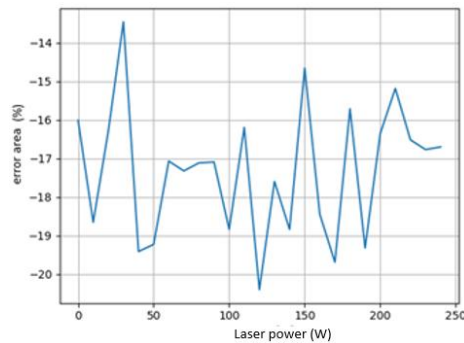


Figure 7.13: Contour area error variation for varying laser power

## Error in the area of the cap with respect to change in second peak threshold

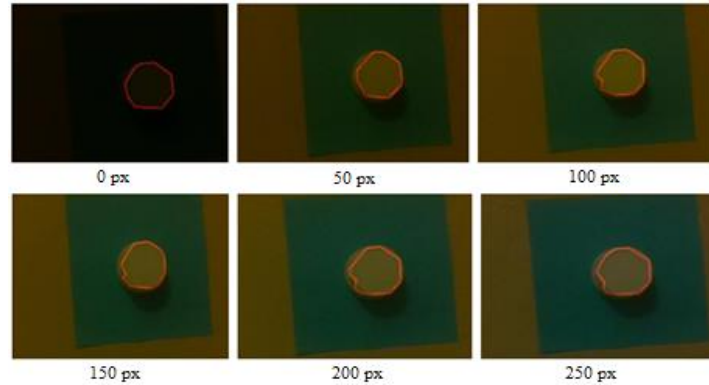


Figure 7.14: Representative contours for various values of second peak threshold

Row 1 (Left to Right): Second peak threshold (pixels) = [0,50, 100]

Row 2 (Left to Right): Second peak threshold (pixels) = [150, 200, 250]

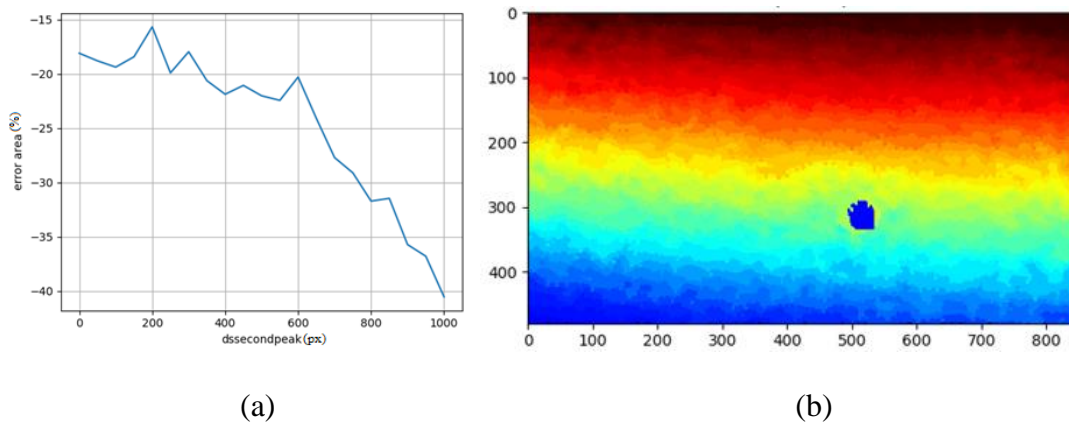


Figure 7.15: (a) Contour area error variation for varying the second peak threshold (b) Depth map for second peak threshold beyond 600 pixels

The second-peak threshold is a parameter used in the stereo-matching algorithm for depth calculation. In the RealSense Software Development Kit (SDK), this parameter is referred to as “dssecondpeak” - Figure 7.14 shows the contours extracted from the depth maps when the Second Peak Threshold was changed from 0-250 pixels. As the second peak threshold tells how different the first peak matches and second peak matches between the left and right camera need to be, as the threshold increases, more matches are regarded as aliasing, and thus depth result is 0 in most pixels. The plot in Figure 7.15 (a) shows a steep increase in error as we increase the second peak threshold beyond 600 pixels. Figure 7.15 (b) shows the corresponding depth map for a high second

peak threshold.

### Error in area of cap with respect to change in the neighbor threshold

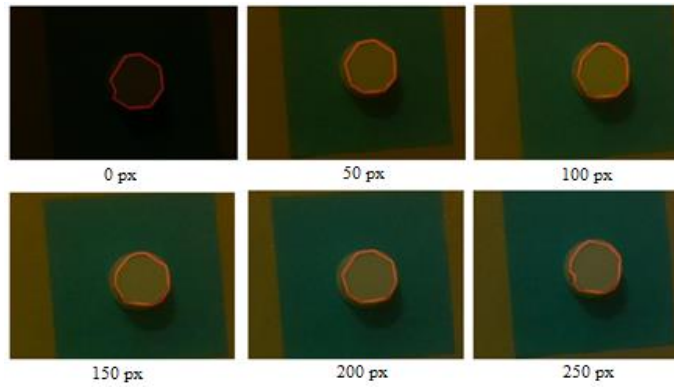


Figure 7.16: Representative contours for various values of neighbor threshold

Row 1 (Left to Right): Neighbor threshold (pixels) = [0,50, 100]

Row 2 (Left to Right): Neighbor threshold (pixels) = [150, 200, 250]

The neighbor threshold is a parameter used in the computation of the stereo matching algorithm for the depth map. In the RealSense SDK, this parameter is referred to as “dsneighbor”. Figure 7.16 shows the contours extracted from the depth maps when the neighbor threshold was changed from 0-250 pixels. The plot in Figure 7.17 (a) shows that the depth map gives a good estimate of the area when the neighbor threshold lies in the range of 0-400 pixels. Higher neighbor thresholds produce many missing values in the depth map computation, rendering a poor-quality depth map, as shown in Figure 7.17 (b).

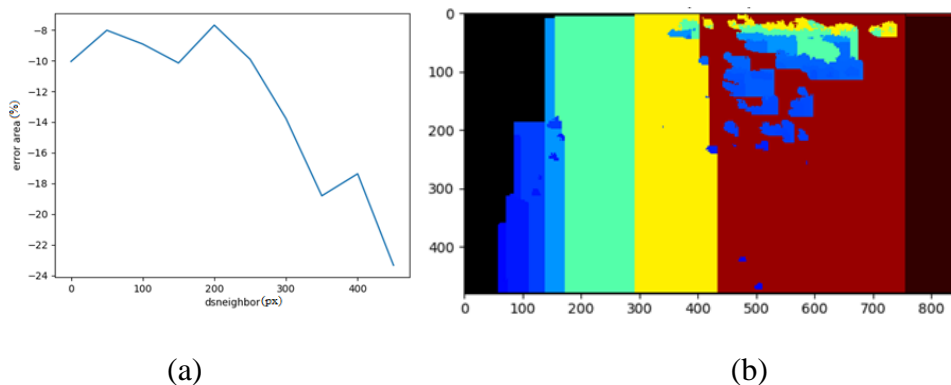


Figure 7.17: (a) Contour area error variation for varying neighbor threshold (b) Distorted depth map for neighbor threshold above 400 pixels

### Error in area of cap with respect to change in Disparity Shift

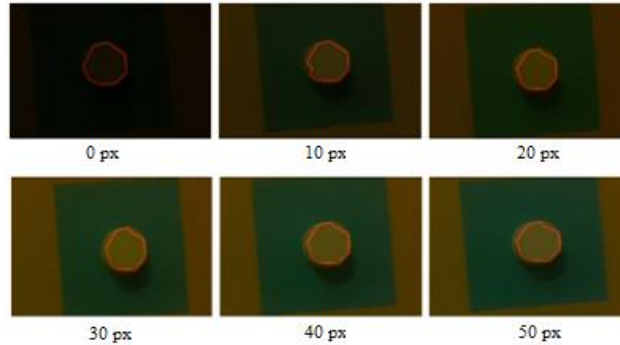


Figure 7.18: Representative contours for various values of disparity shift

Row 1 (Left to Right): Disparity shift (pixels) = [0, 10, 20]

Row 2 (Left to Right): Disparity shift (pixels) = [30, 40, 50]

Disparity Shift changes the minimum and maximum depth that can be registered from the camera. It is measured in pixels. Figure 7.18 shows the contours extracted from the depth maps when the Disparity Shift was changed from 0-50 pixels. Figure 7.19 (a) shows that the depth map gives a good estimate of object area for the disparity shift range 0-120 pixels. Beyond 120 pixels, the camera ends up receiving a very narrow band of depth values ( $Z_{min}$  and  $Z_{max}$ ), which ends up creating a depth map with many values missing. Figure 7.19 (b) demonstrates this phenomenon.

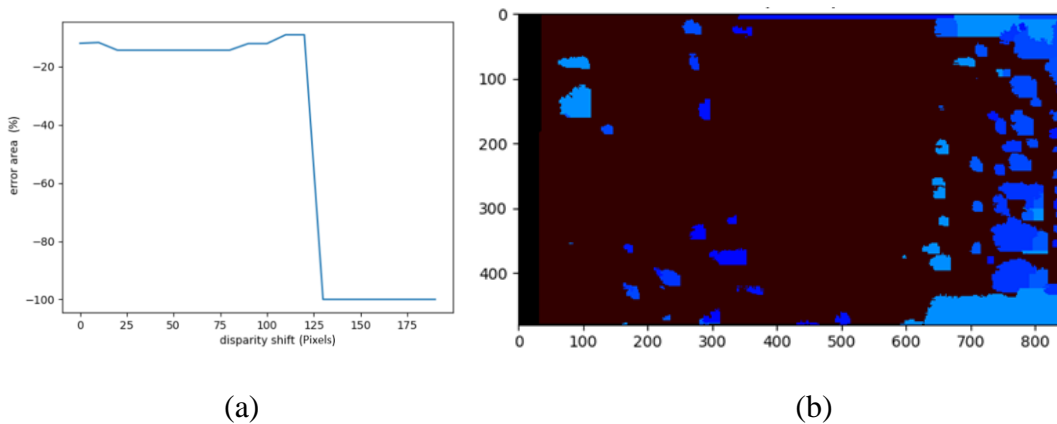


Figure 7.19: (a) Contour area error variation for varying disparity shift (b) Distorted raw depth map beyond disparity shift of 120 pixels

### Error in the area when resolution is changed

Resolution is the fineness of detail in an image measured in pixels per inch (ppi). Higher resolution

images correspond to images that can capture finer features and details in the image, whereas lower resolution images take lower memory space, but are coarser. Table 7.2 shows the error variation in surface area when the image's resolution is changed.

Table 7.2: Variation of error in area computation with resolution

Resolution	Error in area
848x480	-13.29%
640x480	-11.76%
1280x720	-11.95%

### **Error in the area when the surface area of a custom object is changed**

A custom object is created using clay dough. The shape of the object was a cuboid. The length and breadth of a custom object are changed while the thickness is kept constant. In this experiment, objects of varying sizes were placed in front of the camera, and the object's area was found using the contour approximation method. Table 7.3 shows how the error percentage changes as the surface area changes while keeping the object's thickness constant. The experiment gradually reduced the object's size from (1.9x1.5) cm to (0.9x0.5) cm. The reduction in the object's size does not independently impact the measurement accuracy, given that the object's thickness is held constant. This is because a thicker object protrudes further out from the surface and has a greater contrast from the background depths. The resolution limit determines per pixel area, which will impact the size of the object that can be detected.

Table 7.3: Variation of error in area computation with surface area change

Length(mm)	Breadth(mm)	Thickness(mm)	Error%
19	15	13	8.804
14	14	13	-4.329
9	5	13	9.64

## Error in the area when object thickness is changed

Figure 7.20 shows the contours extracted from depth images for varied sizes of object thickness. As shown in Table 7.4, we kept the length and width of the object constant and varied the depth of the defect to find the error rate as the object's thickness is reduced. As expected, the error increases when we reduce the object thickness. For example, at an object thickness of 1cm, the area error increases to 15.69%. The shape characteristic in the detected contour for an object thickness of 18mm captures the morphology of the actual defect. However, as the object thickness is decreased to 10mm, the depth map does not preserve the object's shape. This leads to a much higher error. This error would keep increasing in the sub-millimeter range.

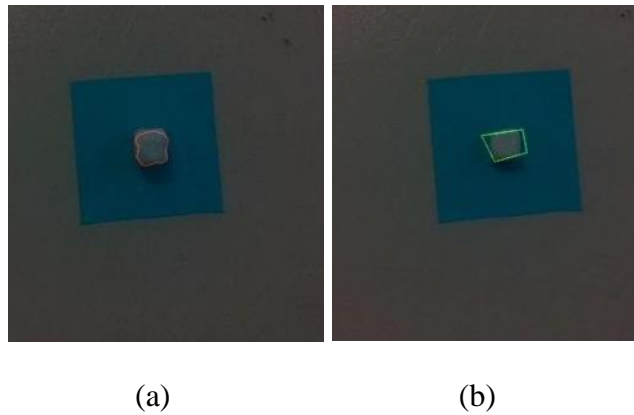


Figure 7.20: Contour capture at an object thickness of (a) 1.8cm and (b) 1cm

Table 7.4: Variation of error in area computation with change in object thickness

Length(mm)	Breadth(mm)	Thickness(mm)	Error%
15	14	18	4.4
15	14	10	15.6%

## 7.2 Sensitivity analysis of object depth measurements to the camera parameters

### 7.2.1 Methodology

For performing these experiments, an object of 18 mm thickness and 11-mm radius is placed on a relatively flat wall in front of the camera as shown in Figure 7.21 and Figure 7.22. First, hole filling is performed on the raw depth map from the camera using the built-in Pyrealsense hole-filling

algorithm. Canny edge detection is performed on the image whose thresholding is determined using the pixel histogram. After canny edge detection, morphological dilation is performed on the edges to ensure continuity. Contour detection is then performed on the dilated image to find the boundary points. The boundary points are a discrete set of pixel coordinates, and a curve is generated through these points to create the best approximation. The procedure is shown in Figure 7.23.

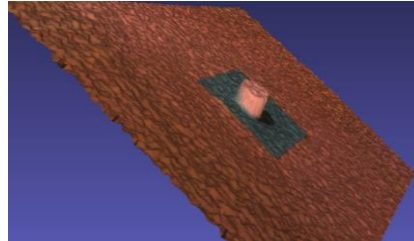


Figure 7.21: Object of thickness 18 mm thickness used for sensitivity analysis

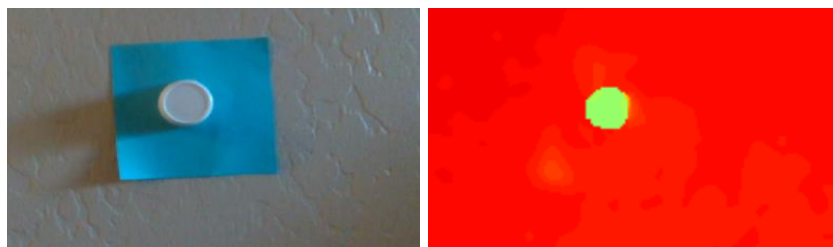


Figure 7.22: RGB and depth map corresponding to the 3D map

After finding the contour and filling the inside part with a uniform color opposite the background, the pixels inside the contour are separated from the background using the color difference. Then, the distance from the camera to the object is obtained by calculating the distance of the camera from the pixels inside the contour and the distance from the camera to the background is obtained by calculating the distance from pixels in the background. Then the erroneous distance for each pixel is removed before calculating the average distance (for all the pixels) from the background and the object separately. The thickness of the object is estimated by the differences between the background depth and the object depth.

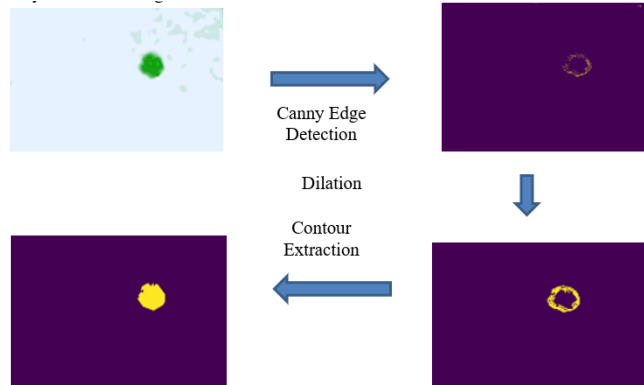


Figure 7.23: Contour extraction process for thickness estimation

### 7.2.2 Results and Discussion

#### Error in the thickness estimation with respect to change in gain

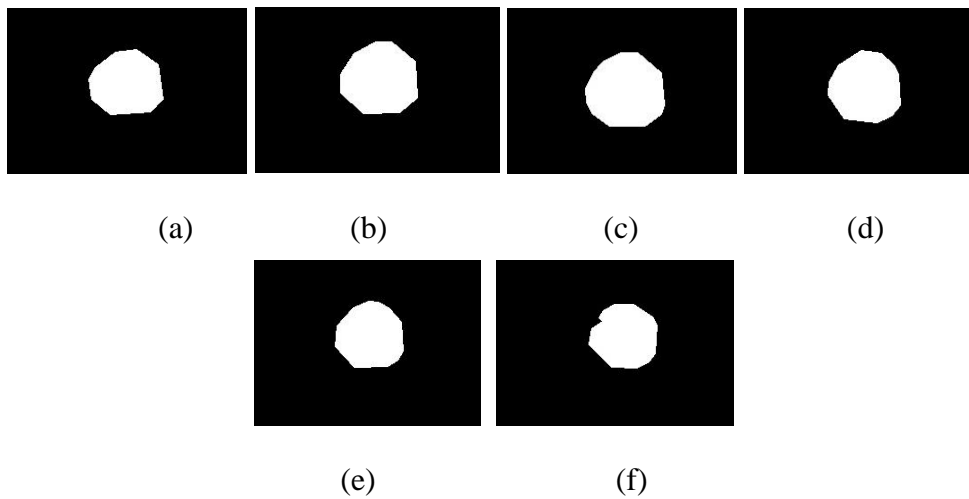


Figure 7.24: Contours across gains (dB) (a) 16 (b) 36 (c) 56 (d) 76 (e) 96 (f) 116

Figure 7.24 shows the resulting object contours and Figure 7.25 shows that we can get the depth information with reasonable accuracy in the Gain range of 16-140 dB with a resolution of 848x480.



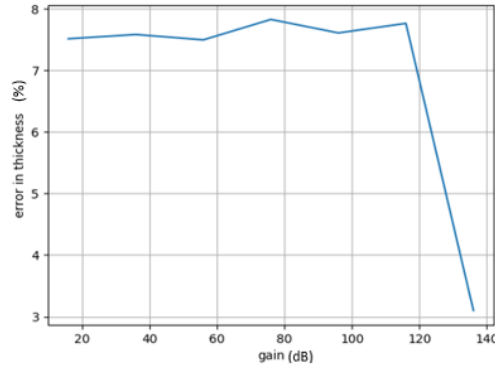


Figure 7.25: Plot of error in thickness estimation versus camera gain

**Error in the thickness estimation with respect to change in exposure**

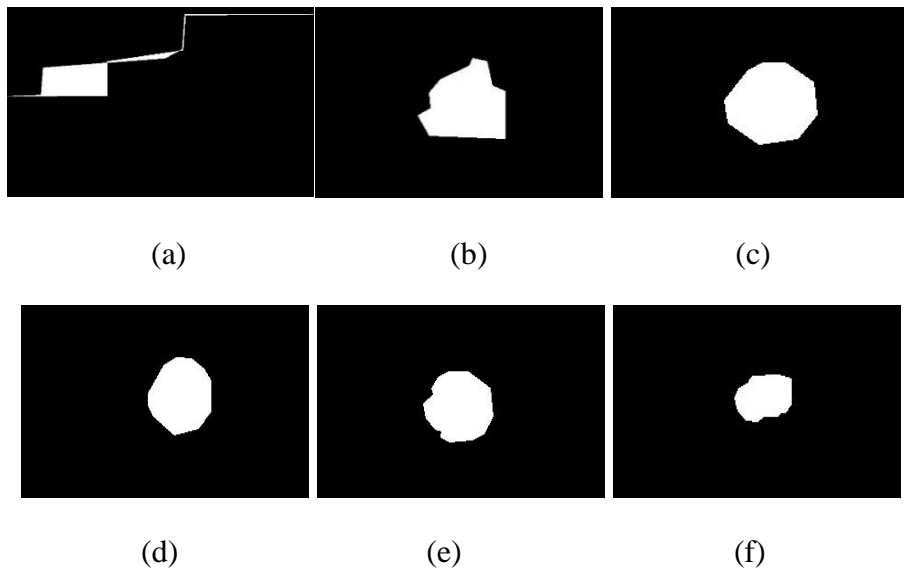


Figure 7.26: Contours across exposures ( $\frac{W}{m^2} s$ ): (a) 0 (b) 500 (c) 1000 (d) 7500 (e) 14500 (f) 16000

Figure 7.26 shows the extracted contours for various exposure settings. As shown in Figure 7.27, the thickness estimation is valid only in the range of 2000 to 16000  $\frac{W}{m^2} s$  with a resolution of 848x480 as the plot shows that the error is low in this range. The contour and raw depth map below 2000  $\frac{W}{m^2} s$  are as shown in Figure 7.28. This shows that exposure below 2000  $\frac{W}{m^2} s$  may not be suitable for thickness estimation. Exposure depends on lighting conditions, and these might vary with different lighting conditions, and auto-exposure algorithms are already present in the Realsense camera system to calibrate this according to the scene brightness. The contour formation

for exposure between  $2000\text{--}16000 \frac{W}{m^2} s$  is shown in Figure 7.29. The contour formation for exposure beyond  $16000 \frac{W}{m^2} s$  is plotted in Figure 7.30. The object contour is not detected in this case, as for every scene, there is a range of exposures for which the depth map produces reasonably accurate object contours, beyond which the camera's performance is severely degraded.

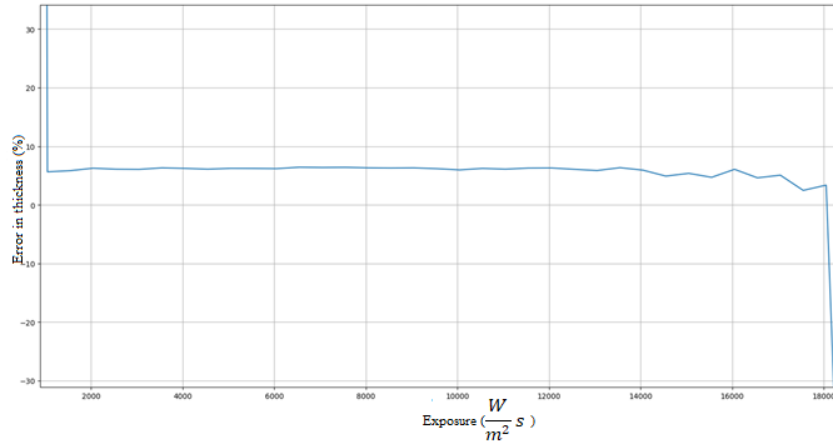


Figure 7.27: Plot of exposure versus error in thickness estimation

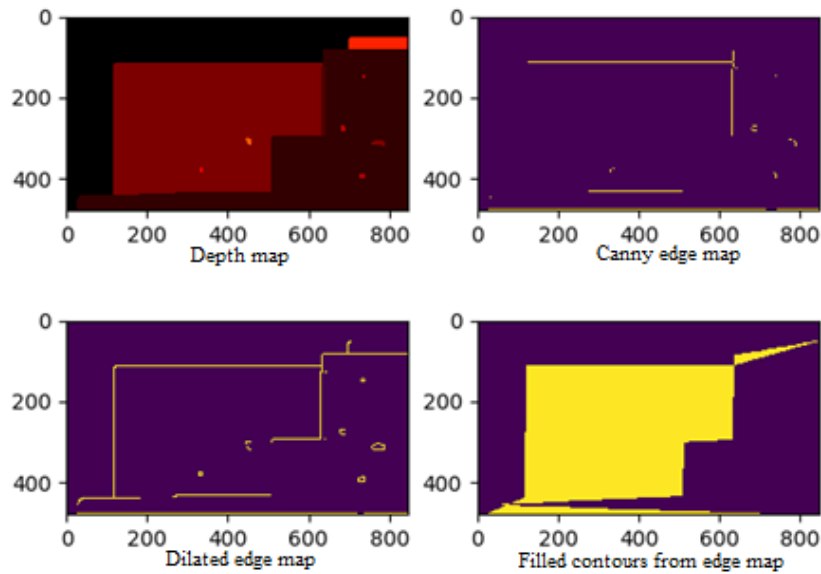


Figure 7.28: Contours and depth map for exposure below  $2000 \frac{W}{m^2} s$

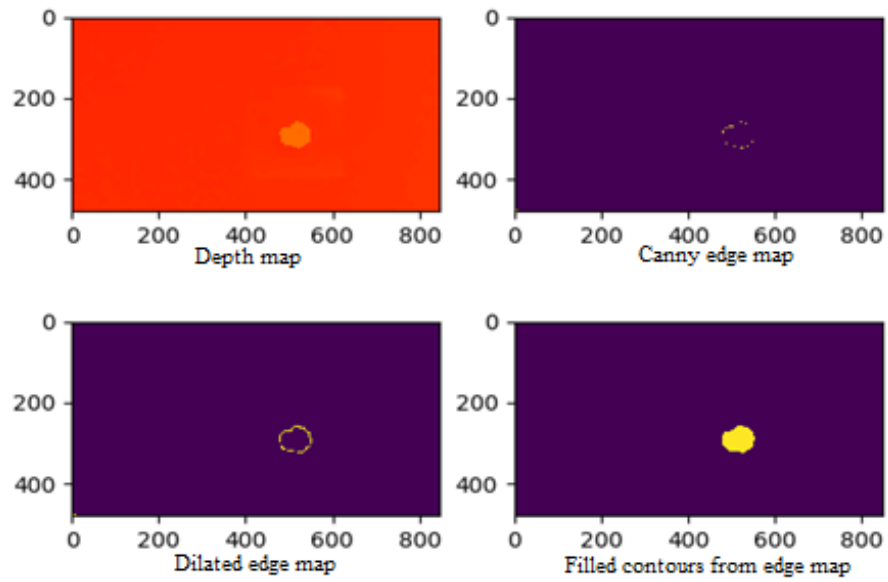


Figure 7.29: Contours and depth map for exposure between  $2000-16000 \frac{W}{m^2} s$

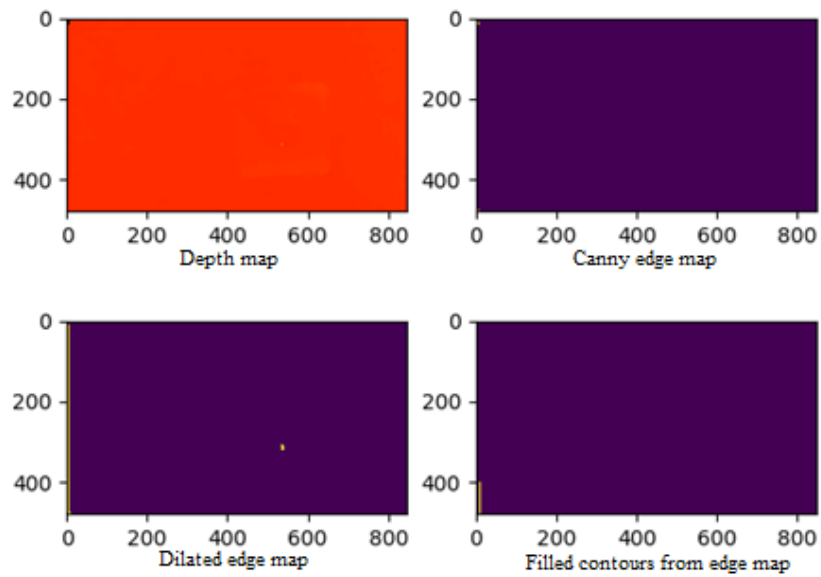


Figure 7.30: Contours and depth map for exposure above  $16000 \frac{W}{m^2} s$

### Error in the thickness estimation with respect to change in second peak threshold

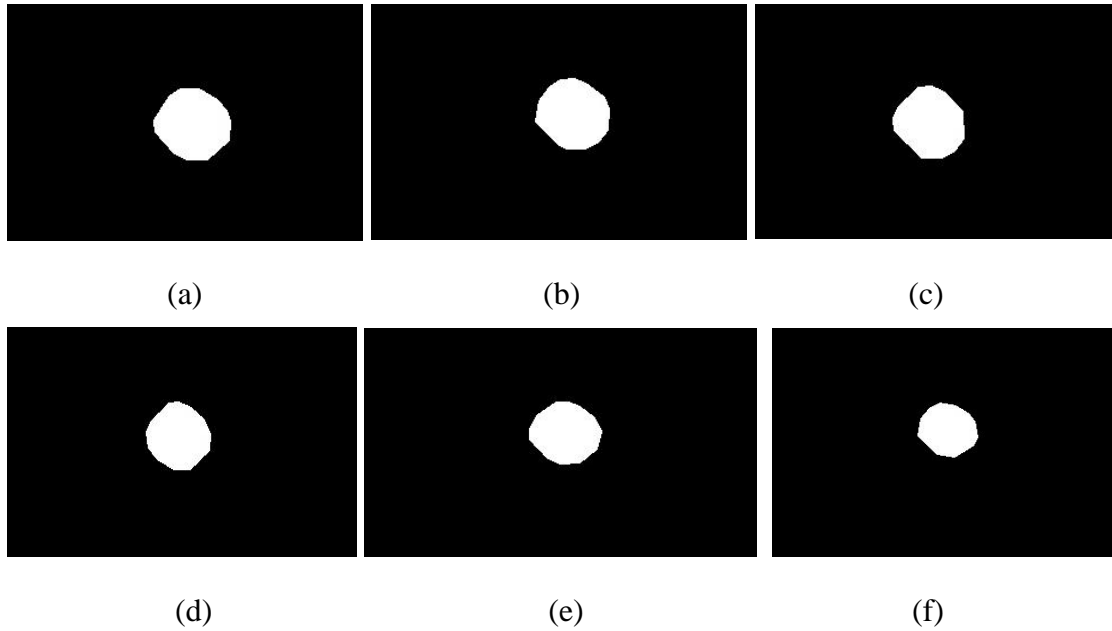


Figure 7.31: Contours across second peak thresholds (pixels):

(a) 0 (b) 100 (c) 200 (d) 300 (e) 400 (f) 500

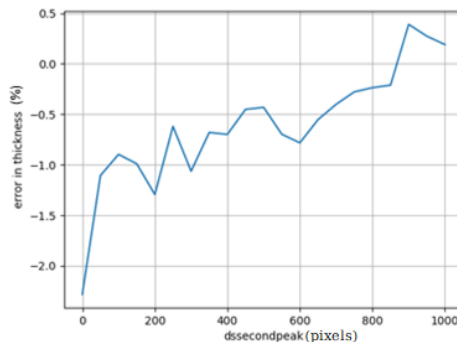


Figure 7.32: Plot of second peak threshold versus error in thickness estimation

As shown in Figure 7.32, does not show a large variation in error magnitude as the second peak threshold is varied. Unlike exposure and gain, which are heavily influenced by factors such as the brightness of the scene, the thickness error is low through the entire range of the second peak threshold. This is because the second peak threshold only influences the fill rate of the depth map, that is, the number of holes, which affects area computation more than thickness. This is because there are variations in external lighting conditions due to these experiments being conducted in the home environment with loose lighting controls in contrast to the controls in the lab.

### Error in the thickness estimation with respect to change in disparity shift

As shown in Figure 7.33 and Figure 7.34, the thickness estimation is good in the entire range of 0 to 150 pixels under the resolution of 848x480. However, this setting may cause holes in the depth map if the camera is located further away from the inspected surface, in which case a setting of zero is more appropriate. Lower depth values require higher disparity shifts for ensuring good depth map quality.

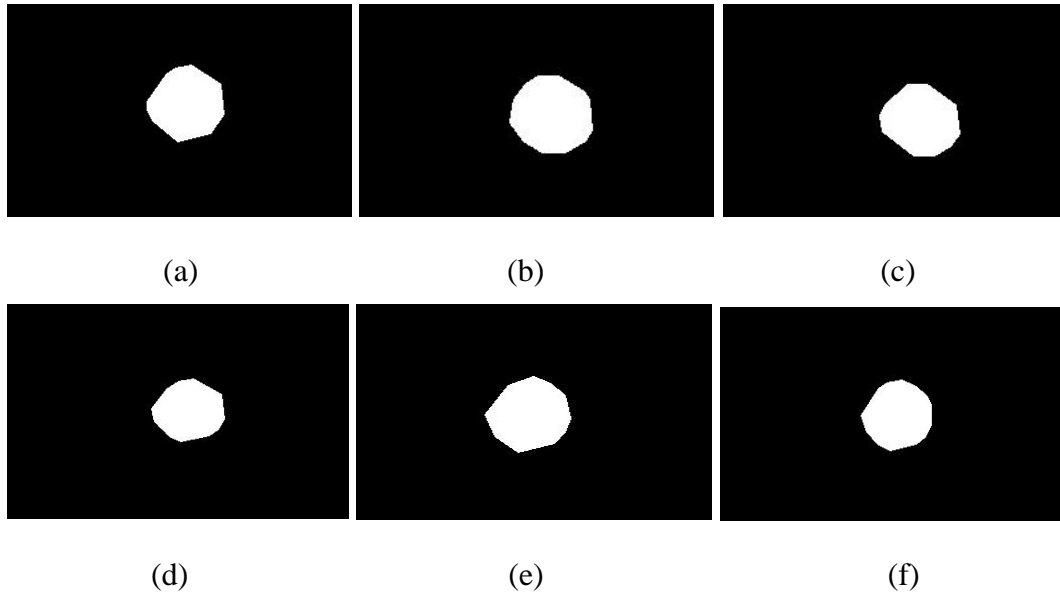


Figure 7.33: Contours across disparity shift (pixel): (a) 0 (b) 50 (c) 100 (d) 150 (e) 200 (f) 250

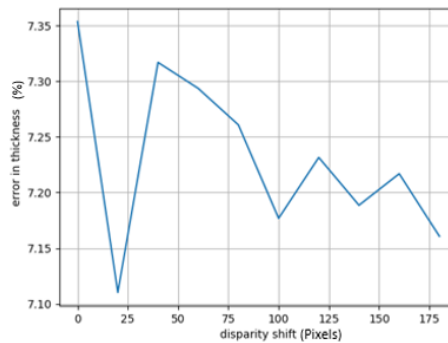


Figure 7.34: Plot of disparity shift versus error in thickness estimation

### Error in the thickness estimation with respect to change in neighbor threshold

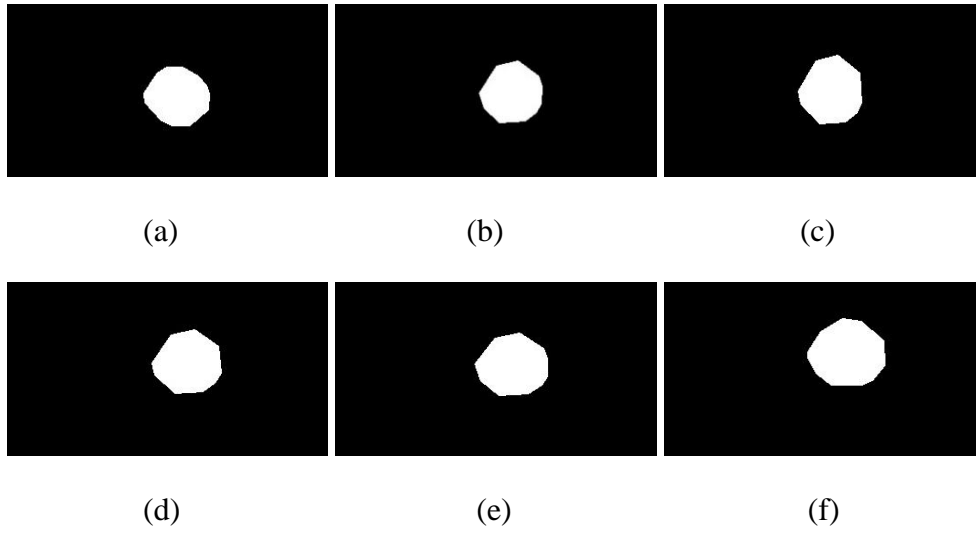


Figure 7.35: Contours across neighbor threshold (pixel): (a) 0 (b) 50 (c) 100 (d) 150 (e) 200 (f) 250

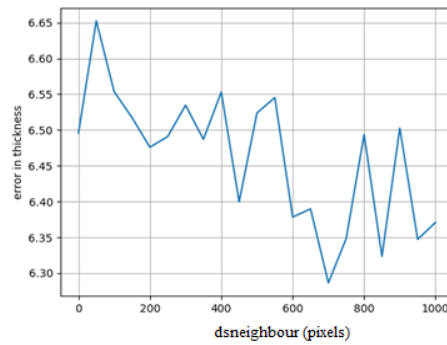


Figure 7.36: Plot of neighbor threshold versus error in thickness

As shown in figures Figure 7.35 and Figure 7.36, there is no significant improvement to the thickness error estimation when the neighbor threshold is varied from 0 to 1000 pixels.

### Error in the thickness estimation with respect to change in laser power

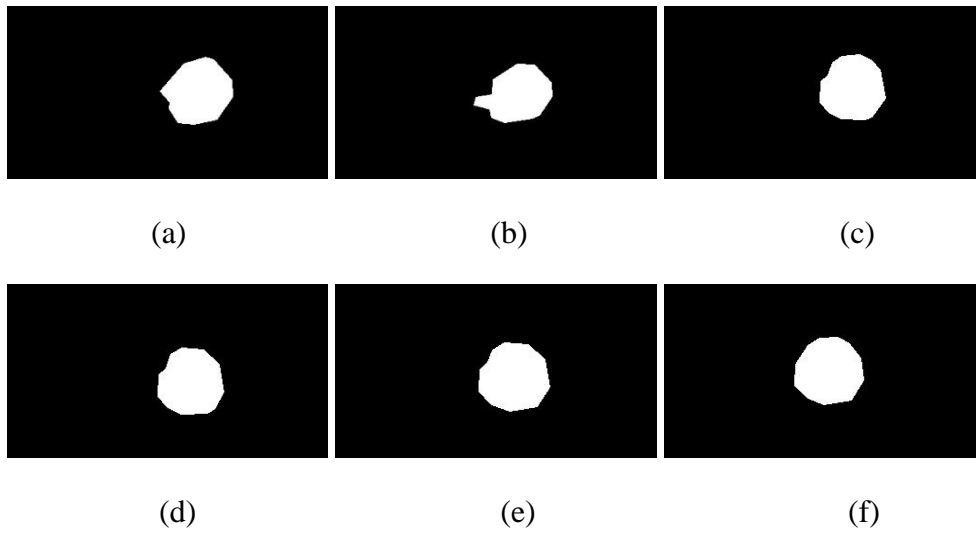


Figure 7.37: Contours across laser power (W): (a) 0 (b) 50 (c) 100 (d) 150 (e) 200 (f) 250

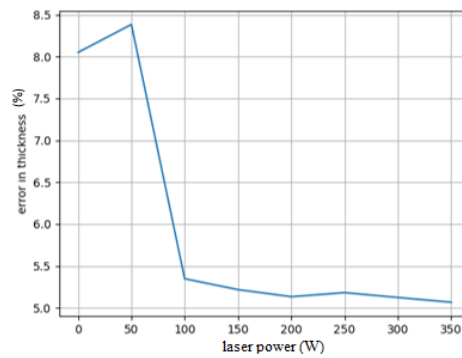


Figure 7.38: Plot of laser power versus error in thickness

As shown in Figure 7.37 and Figure 7.38, the change in laser power impacts the depth map quality and thickness estimation accuracy, with increased laser power showing better performance. This shows the benefit of using active stereo over conventional stereovision for optical metrology.

## Error in the thickness estimation with respect to change in resolution

Table 7.5: Error in thickness estimation versus Resolution

Resolution	Error in thickness	Pixel length (mm)
640x360	-11%	0.5963
848x480	-7.5%	0.4472
1280x720	-8.02%	0.3003

Table 7.5 shows that the resolution has an insignificant effect on thickness but affects the pixel length. It should be noted that the object's thickness is 18 mm, which is above the minimum depth detectable by the camera. Therefore, changing the resolution does not significantly affect the thickness. In addition, in the resolution of 1280x720, the disparity shift needs to be above 20 pixels to get proper information about the object depth.

## Error in the thickness estimation of an object of small thickness

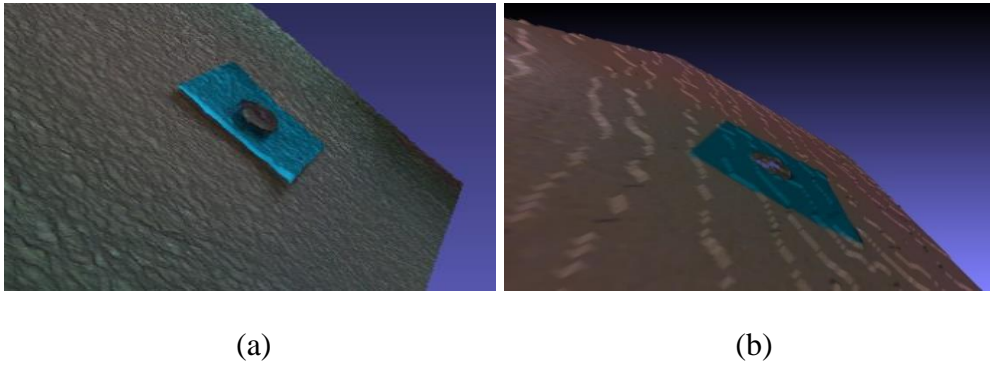


Figure 7.39: (a) 3-D visualization of a 5 mm thickness coin (b) 3-D visualization of a coin of a thickness of 1.75 mm

Figure 7.39 (a) shows the 3D reconstruction of the scene for the object of 5-mm thickness and Figure 7.39 (b) shows the 3D reconstruction of the scene for the object of 1.75-mm thickness. The surface reconstruction is obtained by using the point cloud generated by the depth frame aligned with the RGB camera frame. Then, the RGB texture is applied to the point cloud instead of the depth texture. The 5-mm thick object was created by stacking multiple coins on top of each other for this experiment. For thickness estimation of an object of 5-mm thickness and 24-mm diameter,



the depth visualization is adjusted according to the distance of the object from the camera. Then contour analysis was performed on the raw depth map, as shown in Figure 7.40. The error was found to be around 22% under the resolution of 848x480. The error in the thickness of an object of 24-mm diameter and 1.74-mm thickness was found to be 18.85%. The results are shown in Figure 7.41.

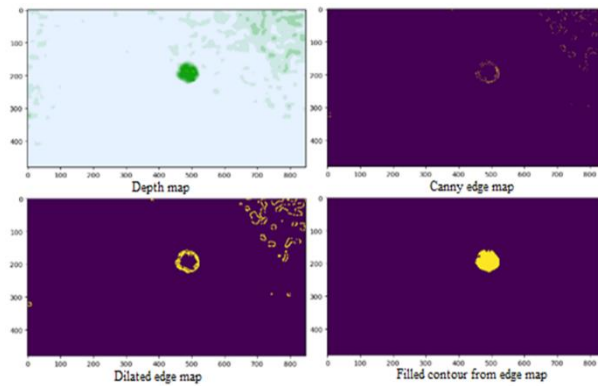


Figure 7.40: Contour extraction from the depth map for the object of 5 mm thickness

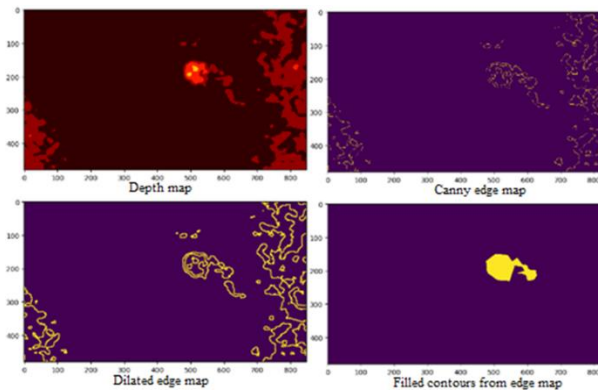


Figure 7.41: Contour extraction from the depth map for the object of 1.74 mm thickness

### **Error in the thickness estimation of an object of a small area**

Contour extraction did not yield reliable results for an object of 8mm diameter and 15mm thickness, so we chose to get the depth information from the Intel RealSense viewer by inspection, as shown in Figure 7.42.

The manual measurement of the depth using the RealSense viewer meant that the number of significant figures that the depth could be computed to was at the millimeter scale. Secondly, it is important to note that the shape of the object was not captured, and only one sample that indicated

a deformation in the surface corresponding to the location of the object was taken. An object of 21-mm thickness and 7.5-mm diameter was difficult to process with our contour algorithm, so we chose to get the depth information from the Intel RealSense viewer, which is used to visualize the raw depth map directly from the Camera, as shown in Figure 7.42 (d). The error was found to be -9.1%.

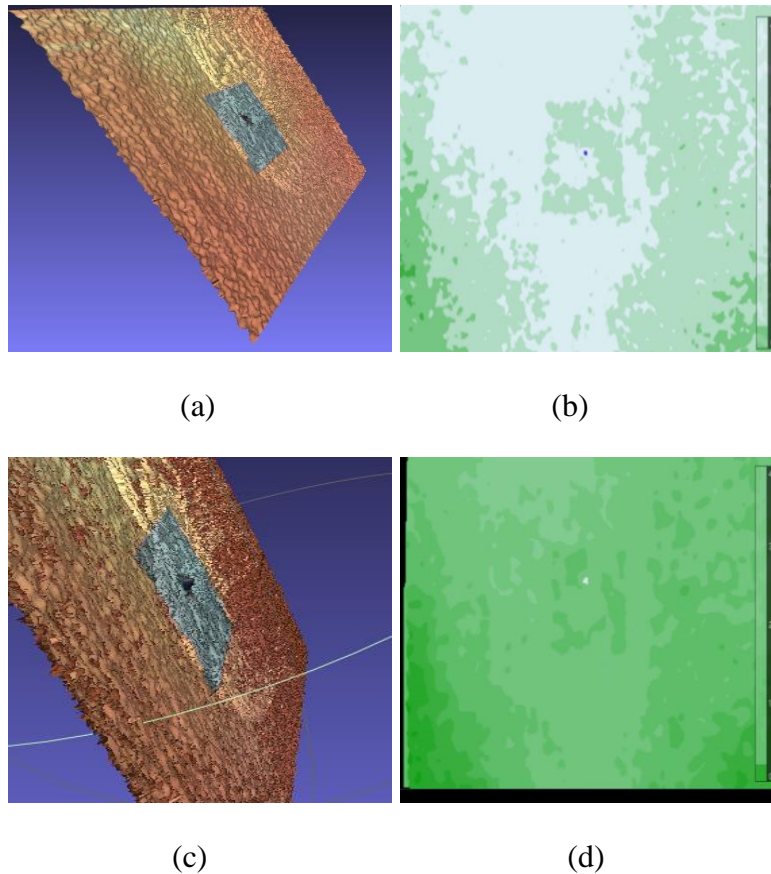


Figure 7.42: (a) 3-D RGB textured point cloud; (b) Depth map for the object of 8 mm diameter  
(c) 3-D RGB textured point cloud; (d) Depth map for the object of 7.5 mm diameter

## 8 APPENDIX B

### 8.1 Hardware prototype 1

We selected a commercial eight degrees of freedom (DOF) vehicle robot with 8-Axis RC robotic arm for the robot carrier. The specification for this robot carrier is given below:

Y100 Tank chassis parameters

- Name: New Y-100 Tank Chassis
- Main body: aluminum alloy
- Surface: sandblasting oxidation
- Track: engineering plastics
- Color: Silver
- Size: 300\*240\*122mm (length \* width \* height)
- Weight: 1.16kg
- Motor: 9V 150rpm with encoder motor

#### 9V 150rpm with encoder motor

- Working voltage: 9V
- Output rate: 150±10% rpm
- Load current: 200mA (Max)
- Stall current: 4500 mA (max)
- Stalling torque: 9.5kg
- Load speed: 100±10% rpm
- Load torque: 3000Ncm
- Load current: 1200mA (Max)
- Encode parameters: 2 pulses / circle
- Sensor operating voltage: 3-5V

#### Robotic arm parameters

- Mechanical arm body material: aluminum alloy
- Weight: 0.82kg (including servo weight)
- Color: silver
- Servo optional: MG996R metal gear large servo
- Features: The mechanical arm can be equipped with a mechanical gripper the weight of the grip is about 500g; the base of the robot arm can be rotated 360 degrees.

#### MG996R servo parameters

- Name: MG996R
- Net weight: 55 g

- Size: 40.7\*19.7\*42.9 mm
- Pulling force: 9.4 kg/cm (4.8 V), 11 kg/cm (6 V)
- Reaction rate: 0.17 sec / 60 degree (4.8 V), 0.14 sec / 60 degree (6 V)
- Working voltage: 4.8-7.2 V
- Working temperature: 0°C-55°C
- Gear form: metal gear

Working dead zone: 5  $\mu$ s

### **Bearing wheel installation**

The materials required to assemble the bearing wheel are M3\*8 screw, 17 mm copper hex spacers, M2 screw, bearings, wheel discs, and stainless-steel connector.

### **Driving wheel installation**

For the driving wheel, the required materials are 28 mm copper hex screws, M3\*8 screw, stainless-steel connectors, jackscrew, geared wheel discs, aluminum alloy coupling, and M4\*16 screws.

### **Tank Chassis installation**

The required materials to assemble the Tank Chassis include 9V 150 rpm encoder motor, LED lights, tracks, chassis panels, power cable, M3\*12 screw, M3 nut, M3\*10 screw are necessary for the assembly of the tank. There are a total of ten bearing wheels and two driving wheels for this model. The track is installed and since the tracks are individually connected with needles, the length can be adjusted according to the needs.

### **Assembly of Robot Arm**

The robot arm consists of a rotating base, left and right swing arm, a rocking arm, and a control board.

#### **Rotating Base**

The materials required are M4\*11 double pass coupling, M4\*6 flat head screws, M3\*8 inner hexagon screws, M3 nut, tray bearing, single pass coupling, 25T disc metal horns, and steering gear.

#### **Left and Right Swing Arm**

M3\*8 inner hexagon screws, 25T metal horn, M4\*6 screws, pendulum frame, M3\*5 screws, steering gear, and left, and right frames are assembled. And this assembly is installed on the rotating base.

### **Rocking arm**

M3\*8 inner hexagon screws, U bracket, 25T metal horn, M3\*8 flat head screw, steering gear, M4\*6 screws, steering gear bracket, swing fixing bracket, rocking fixing bracket, U shaped support frame, connecting rod are installed together according to the manual. This assembled part is installed to the previously connected servo arm with a rotating base.

### **Control Board**

The control board tray bracket is attached to the rotating base. Materials required are M3\*8 screws, M3\*6 screws, and a steering gear bracket. Bearing is attached to the steering gear bracket along with steering gear.

### **Metal claw**

The assembly of the claw is divided into 3 parts, which are claw arm 1, claw arm 2, and base of claw. The parts include servo motor MG995, 25T metal horn, M3\*8 hexagon screw, M3 nut, single pass couplings, M3\*6 flat head screw, M3\*12 screw, and bearings.

### Hardware specifications of the fish-eye stereo camera

Performance: 1920x1080 MJPEG@30fps      S/N Ratio: 39db

Sensor: AR0330      Dynamic range: 72.4db

Pixel Size: 5.07um X 3.38um

Lens Parameter: M12 fisheye lens

Board size: 86x23 mm

Mini illumination: 0.1lux

Sensitivity: 2.0 v/lux-sec @550nm

Voltage: DC5V

Current: 220mA-280mA



Figure 8.1: Fisheye stereo camera

## 8.2 Hardware prototype 2

We list here the details of the components for the second prototype.

Table 8.1: Components list for the height adjustment mechanism

Component	No of pieces
Linear Actuators (PA-07)	2
Bottom mount for linear actuators	2
Top mount for linear actuators	2
Linear actuator support	2
Camera platform	1
Camera platform support	2
M4*35mm bolts	4
M4 nuts	8
M4 washers	4

Table 8.2: Technical specifications of the 12V brushed DC motors

<b>Specifications</b>	<b>Values</b>
Gear ratio	20.4:1
Torque at maximum efficiency	1.1 kg.cm
Maximum power	9.4 W
Speed at maximum efficiency	420 rpm

Table 8.3: Technical specifications of the PA-07 linear actuators

<b>Specifications</b>	<b>Values</b>
Driver voltage	5-35V
Driver current	2A
Max supply voltage (Maximum)	46V
Motor supply current (Maximum)	2A

Table 8.4: Truth Table for L298N Motor Driver Module for Direction Control

<b>Input 1</b>	<b>Input 2</b>	<b>Spinning Direction</b>
Low	Low	Motor OFF
High	Low	Forward
Low	High	Backward
High	High	Motor OFF

Table 8.5: Technical specifications of the DRV8834 Stepper Motor Driver Module

Specifications	Values
Continuous current per phase	1.5A
Max current per phase	2A
Max logic voltage	5.5V
Max power supply voltage	11.8V

Table 8.6: Truth Table for DRV8834 Stepper Motor Driver Module for Micro stepping

M0	M1	Micro step resolution
Floating	Low	1/4 step
Low	High	1/8 step
High	High	1/16 step
Floating	High	1/32 step

Table 8.7: PS4 Controller button mappings to robot functions.

Key press	Corresponding motion of the robot
L1	Forward Stroke of Actuators
R1	Backward Stroke of Actuators
Start	Start Automated Scanning
Options	Stop Automated Scanning
Triangle	Enabling 1/4 Micro stepping
Circle	Enabling 1/8 Micro stepping
Square	Enabling 1/32 Micro stepping
Forward	Forward movement of robot
Backward	Backward movement of robot



## **References**

- Adelson, E. H., Anderson, C. H., Bergen, J. R., Burt, P. J., & Ogden, J. M. (1984). Pyramid methods in image processing. *RCA Engineer*, 29(6), 33–41.
- Alzuhiri, M., Farrag, K., Lever, E., & Deng, Y. (2021). An Electronically Stabilized Multi-Color Multi-Ring Structured Light Sensor for Gas Pipelines Internal Surface Inspection. *IEEE Sensors Journal*, 21(17), 19416–19426. <https://doi.org/10.1109/JSEN.2021.3086415>
- Bishop, C. M. (2014). Bishop - Pattern Recognition And Machine Learning - Springer 2006. *Antimicrobial Agents and Chemotherapy*, 58(12).
- Brach, K., Sick, B., & Urr, O. D. (2020). *Single Shot MC Dropout Approximation*. <https://doi.org/10.48550/arxiv.2007.03293>
- Chen, W., Gao, Y., Gao, L., & Li, X. (2018). A New Ensemble Approach based on Deep Convolutional Neural Networks for Steel Surface Defect classification. *Procedia CIRP*. <https://doi.org/10.1016/j.procir.2018.03.264>
- Cheng, X., & Yu, J. (2021). RetinaNet with Difference Channel Attention and Adaptively Spatial Feature Fusion for Steel Surface Defect Detection. *IEEE Transactions on Instrumentation and Measurement*. <https://doi.org/10.1109/TIM.2020.3040485>
- Choi, J. B., Goo, B. K., Kim, J. C., Kim, Y. J., & Kim, W. S. (2003). Development of limit load solutions for corroded gas pipelines. *International Journal of Pressure Vessels and Piping*, 80(2), 121–128. [https://doi.org/10.1016/S0308-0161\(03\)00005-X](https://doi.org/10.1016/S0308-0161(03)00005-X)
- Di, H., Ke, X., Peng, Z., & Dongdong, Z. (2019). Surface defect classification of steels with a new semi-supervised learning method. *Optics and Lasers in Engineering*. <https://doi.org/10.1016/j.optlaseng.2019.01.011>
- Dong, H., Song, K., He, Y., Xu, J., Yan, Y., & Meng, Q. (2020). PGA-Net: Pyramid Feature Fusion and Global Context Attention Network for Automated Surface Defect Detection. *IEEE Transactions on Industrial Informatics*. <https://doi.org/10.1109/TII.2019.2958826>
- Echard, B., Gayton, N., & Lemaire, M. (2011). AK-MCS: An active learning reliability method combining Kriging and Monte Carlo Simulation. *Structural Safety*, 33(2), 145–154. <https://doi.org/10.1016/J.STRUSAFE.2011.01.002>

- French, G., Laine, S., Aila, T., Mackiewicz, M., & Finlayson, G. (2019). *Semi-supervised semantic segmentation needs strong, varied perturbations*.
- Gal, Y., & Ghahramani, Z. (n.d.). *Dropout as a Bayesian Approximation: Appendix*.
- Gal, Y., & Ghahramani, Z. (2016). *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning* (pp. 1050–1059). PMLR. <https://proceedings.mlr.press/v48/gal16.html>
- Hartley, R., & Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press. <https://doi.org/10.1017/CBO9780511811685>
- He, K., Gkioxari, G., Dollar, P., & Girshick, R. (2017). Mask R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*. <https://doi.org/10.1109/ICCV.2017.322>
- He, Z., & Liu, Q. (2020). Deep Regression Neural Network for Industrial Surface Defect Detection. *IEEE Access*. <https://doi.org/10.1109/ACCESS.2020.2975030>
- Intel RealSense. (2022). *Intel RealSense D400 Series Product Family Datasheet*.
- Kendall, A., & Gal, Y. (2017). What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 5580–5590.
- Labbé, M., & Michaud, F. (2019). RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*. <https://doi.org/10.1002/rob.21831>
- Laine, S., & Aila, T. (2017). Temporal ensembling for semi-supervised learning. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*.
- Lee, D.-H. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. *ICML 2013 Workshop: Challenges in Representation Learning*.
- Levin, A., Lischinski, D., & Weiss, Y. (2004). Colorization using optimization. *ACM Transactions on Graphics*. <https://doi.org/10.1145/1015706.1015780>
- Li, J., Su, Z., Geng, J., & Yin, Y. (2018). Real-time Detection of Steel Strip Surface Defects Based on Improved YOLO Detection Network. *IFAC-PapersOnLine*.

<https://doi.org/10.1016/j.ifacol.2018.09.412>

- Li, Y., Han, Z., Xu, H., Liu, L., Li, X., & Zhang, K. (2019). YOLOv3-lite: A lightweight crack detection network for aircraft structure based on depthwise separable convolutions. *Applied Sciences (Switzerland)*. <https://doi.org/10.3390/app9183781>
- Liu, M., Liu, Y., Hu, H., & Nie, L. (2016). Genetic algorithm and mathematical morphology based binarization method for strip steel defect image with non-uniform illumination. *Journal of Visual Communication and Image Representation*. <https://doi.org/10.1016/j.jvcir.2015.04.005>
- Liu, Y., Xu, K., & Xu, J. (2019). An improved MB-LBP defect recognition approach for the surface of steel plates. *Applied Sciences (Switzerland)*. <https://doi.org/10.3390/app9204222>
- Liu, Y., Yuan, Y., Balta, C., & Liu, J. (2020). A light-weight deep-learning model with multi-scale features for steel surface defect classification. *Materials*. <https://doi.org/10.3390/ma13204629>
- Miyato, T., Maeda, S. I., Koyama, M., & Ishii, S. (2019). Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. <https://doi.org/10.1109/TPAMI.2018.2858821>
- of Mechanical Engineers, A. S. (2012). *Manual for Determining the Remaining Strength of Corroded Pipelines: A Supplement to ASME B31 Code for Pressure Piping: an American National Standard*. American Society of Mechanical Engineers.
- Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*. <http://arxiv.org/abs/1804.02767>
- Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. <https://doi.org/10.1109/TPAMI.2016.2577031>
- Saiz, F. A., Serrano, I., Barandiaran, I., & Sanchez, J. R. (2018). A Robust and Fast Deep Learning-Based Method for Defect Classification in Steel Surfaces. *9th International Conference on Intelligent Systems 2018: Theory, Research and Innovation in Applications, IS 2018 - Proceedings*. <https://doi.org/10.1109/IS.2018.8710501>

- Scheinker, A., Scheinker, D., Alexander Scheinker, C., & Alamos, L. (2021). Extremum seeking for optimal control problems with unknown time-varying systems and unknown objective functions. *International Journal of Adaptive Control and Signal Processing*, 35(7), 1143–1161. <https://doi.org/10.1002/ACS.3097>
- Semi-supervised semantic segmentation needs strong, varied perturbations.* (n.d.).
- Shelhamer, E., Long, J., & Darrell, T. (2017). Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. <https://doi.org/10.1109/TPAMI.2016.2572683>
- Shi, J., & Tomasi, C. (1994). Good features to track. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. <https://doi.org/10.1109/cvpr.1994.323794>
- Shi, Y., Cui, L., Qi, Z., Meng, F., & Chen, Z. (2016). Automatic road crack detection using random structured forests. *IEEE Transactions on Intelligent Transportation Systems*. <https://doi.org/10.1109/TITS.2016.2552248>
- Srivastava, N., Hinton, G., Krizhevsky, A., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15, 1929–1958.
- Suvdaa, B., Ahn, J., & Ko, J. (2012). Steel surface defects detection and classification using SIFT and voting strategy. *International Journal of Software Engineering and Its Applications*.
- Tabernik, D., Šela, S., Skvarč, J., & Skočaj, D. (2020). Segmentation-based deep-learning approach for surface-defect detection. *Journal of Intelligent Manufacturing*. <https://doi.org/10.1007/s10845-019-01476-x>
- Tarvainen, A., & Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in Neural Information Processing Systems*.
- The American Society of Mechanical Engineers. (2009). ASME B31G - Manual for Determining the Remaining Strength of Corroded Pipelines. *American National Standard*.
- Vanaei, H. R., Eslami, A., & Egbewande, A. (2017). A review on pipeline corrosion, in-line

- inspection (ILI), and corrosion growth rate models. In *International Journal of Pressure Vessels and Piping*. <https://doi.org/10.1016/j.ijpvp.2016.11.007>
- Verma, V., Lamb, A., Kannala, J., Bengio, Y., & Lopez-Paz, D. (2019). Interpolation consistency training for semi-supervised learning. *IJCAI International Joint Conference on Artificial Intelligence*. <https://doi.org/10.24963/ijcai.2019/504>
- Wang, Y., Gao, L., Gao, Y., & Li, X. (2021). A new graph-based semi-supervised method for surface defect classification. *Robotics and Computer-Integrated Manufacturing*. <https://doi.org/10.1016/j.rcim.2020.102083>
- Yang, F., Zhang, L., Yu, S., Prokhorov, D., Mei, X., & Ling, H. (2020). Feature Pyramid and Hierarchical Boosting Network for Pavement Crack Detection. *IEEE Transactions on Intelligent Transportation Systems*. <https://doi.org/10.1109/TITS.2019.2910595>
- Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., & Yoo, Y. (2019). *CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features*.
- Zhang, B., & Ma, X. L. (2019). A review—Pitting corrosion initiation investigated by TEM. In *Journal of Materials Science and Technology* (Vol. 35, Issue 7). <https://doi.org/10.1016/j.jmst.2019.01.013>
- Zhang, G., & Vela, P. A. (2015). Good features to track for visual SLAM. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. <https://doi.org/10.1109/CVPR.2015.7298743>
- Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11), 1330–1334.
- Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning Deep Features for Discriminative Localization. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. <https://doi.org/10.1109/CVPR.2016.319>