

# CAAP Annual Report

Date of Report: September 29, 2020

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

Contract Number: 693JK31950002CAAP

Project Title: AI-enabled Interactive Threats Detection using a Multi-camera Stereo Vision System

Prepared by: Arizona State University

Contact Information:

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

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

For fiscal year ending: September 29, 2020

## **Business and Activity Section**

### **(a) Summary of Research Activities in the First Fiscal Year**

#### **Q1 (09/30/2019 to 12/29/2019)**

For Task 1, stereo vision algorithm that can generate real-time depth map of the pipeline inner surface was developed. The initial design and assembly of the prototype device was performed. A preliminary demonstration using the developed hardware and software was conducted. For Task 2, a literature review on AI-based methods for anomaly detection was performed and suitable methods for pipeline threats detection using depth map information were explored. Please see the first quarterly report for detailed description of work performed.

#### **Q2 (12/30/2019 to 03/29/2020)**

For Task 1, an algorithm for distance estimation using stereo vision was developed to measure the defect dimensions. Experiments were conducted to verify the effectiveness and accuracy of the developed algorithm. For Task 2, an algorithm to perform unsupervised segmentation of scenes using colored image and disparity map was developed and the proposed image segmentation algorithm was tested on a benchmark dataset. Please see the second quarterly report for detailed description of work performed.

#### **Q3 (03/30/2019 to 06/29/2020)**

For Task 1, a post-processing algorithm was introduced to improve the quality of depth map and the improved depth map was used to test the unsupervised segmentation algorithm developed in Q2 and perform distance estimation using the segmentation results. For Task 2, a fully supervised algorithm for RGB-D semantic segmentation was developed to detect pipeline anomalies and a benchmark dataset was adopted to test the fully supervised segmentation algorithm. Please see the third quarterly report for detailed description of work performed.

#### **Q4 (06/30/2019 to 09/29/2020)**

For Task 1, a hole filling algorithm was proposed to improve the post-processing algorithm developed in Q3 and the improved algorithm was adopted to perform extensive parametric analysis to examine the effects of different camera parameters on the accuracy of defect size estimation in order to recommend the optimal parameters for inspection. Furthermore, the 3D reconstruction of the pipeline inner surface was achieved using the RGB-D SLAM technique. In addition, modification to the in-line inspection (ILI) prototype

design was also proposed. For Task 2, the fully supervised algorithm developed in the previous quarter was integrated with the improved post-processing algorithm in Task 1 to perform pipeline defect detection using experimental data. Detailed description of work performed is provided below.

### **(b) Contract Activity of the Fourth Quarter**

Discussion about contract modifications or proposed modifications:

None

Discussion about materials purchased:

1. Intel® RealSense™ camera

### **(c) Status Update for the Fourth Quarter**

In this quarter, the research team works on Task 1.1 and Task 2.1 to develop algorithms for defect size estimation using stereo vision and supervised image segmentation using disparity map and color images. Experiments were conducted to verify the effectiveness and accuracy of the developed algorithms.

#### **Student Training Activities**

- Sampriti Neog (MS student) works on stereo vision algorithm development for pipeline imaging, defect size estimation, and preliminary demonstration to test the effectiveness and accuracy of the developed algorithm. (Task 1)
- Rahul Rathnakumar (PhD student) works on developing an unsupervised algorithm for image segmentation as a method for preprocessing the image for pipeline defect detection. (Task 2)

### **(d) Cost Share Activity**

All cost share requirements have been satisfied in the past quarter and detailed financial report will be submitted by ASU financial department.

### **(d) Detailed Description of Work Performed**

#### **1. Background and Objectives**

Pipeline anomalies such as fatigue cracks, stress corrosion cracking, corrosion pits, and seam weld defects are major threats to the integrity of pipeline systems. The detection and characterization of these pipeline anomalies are critical for the safe operation of pipeline infrastructure, which is the objective of this ongoing project. The objective of this project is to develop a vision-based inspection tool using stereo vision and AI-enabled computer vision algorithms to address the pipeline anomaly detection and characterization.

Stereo vision uses two or more cameras to extract three-dimensional (3D) information by estimating the relative depth of points observed in digital images. The principle of stereo vision is illustrated in Fig. 1. In Fig. 1(a), C1 and C2 represent the optical centers of two cameras;  $b$  is the baseline distance between two cameras; P is the object point; and P1 and P2 are the projection of point P in the image plane. Points C1, C2, and P form a plane known as the epipolar plane. Fig. 1(b) shows a top view of the epipolar plane where  $f$  is the focal length. Based on similar triangles, we have:

$$\frac{z}{f} = \frac{x}{x_l} \quad \frac{z}{f} = \frac{x-b}{x_r} \quad \frac{z}{f} = \frac{y}{y_l} = \frac{y}{y_r} \quad (1)$$

where  $(x,y,z)$  is the global coordinate of the object point P, and  $(x_l,y_l,z_l)$  and  $(x_r,y_r,z_r)$  is the coordinate of the projection of point P in the left and right image planes, respectively.

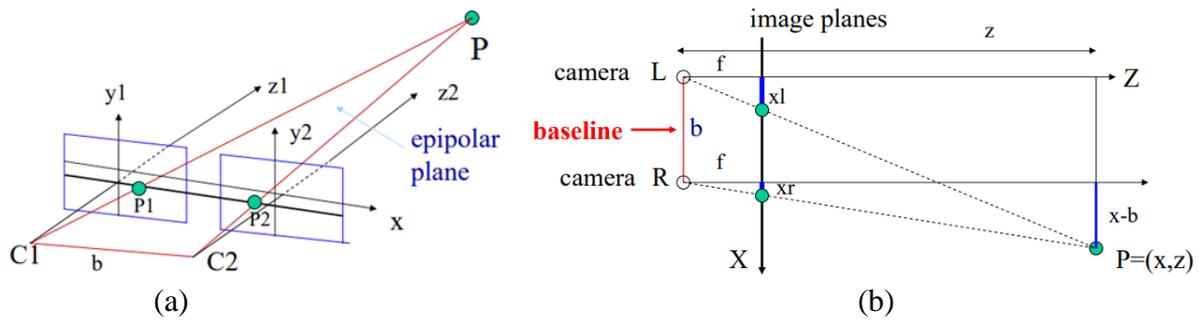


Figure 1: Principle of stereo vision: (a) epipolar plane; (b) triangulation

Based on the relationships given in Eq. (1), the global coordinate of point P can be calculated as:

$$z = \frac{fb}{(x_l - x_r)} = \frac{fb}{d} \quad x = \frac{b}{d}x_l \quad y = \frac{b}{d}y_l \quad (2)$$

where the difference  $d = (x_l - x_r)$  is known as the disparity. Using Eq. (2), we can determine the depth of any scene point and thus construct a depth map of the observed scene. This method of determining depth from disparity is called triangulation. In practice, triangulation will be used to find the 3D locations of critical points on pipeline defects, from which we can estimate the distances between critical points to accurately characterize pipeline defects.

AI-enabled methods for threat detection can serve a key role in risk assessment of structures. Multi-modal analysis of a scene gives us multiple sources of information from which we can determine pertinent risks and threats. Fusing these sources could potentially give us enhanced assessments of system condition. In this report, we discuss a method to fuse multi-modal information sources for scene segmentation. Data obtained from multiple sensors can be processed, combined and manipulated in innovative ways to provide better predictions. The most commonly used definition of data fusion was proposed by the Joint Directors of Laboratories (JDL) workshop: “A multi-level process dealing with the association, correlation, combination of data and information from single and multiple sources to achieve refined position, identify estimates and complete and timely assessments of situations, threats and their significance.” This is relevant for our project as we use depth and color information jointly for making predictions on pipeline condition. Currently, we implement a simple fusion technique for both data sources after minimal processing. However, future work would focus more on improving the ways in which we combine data sources. The method explored in this report is unsupervised, and parameter free, which yields a fully automatic analysis of the input scene.

The objective of the research in the 4th quarter is to: (1) Improve the hole filling algorithm proposed in the previous quarter; (2) Propose a modification to the ILI tool design; (3) Demonstrate the 3D reconstruction of a pipeline section using RGB-D SLAM Techniques; (4) Perform sensitivity analysis to determine optimal camera parameters for object size estimation using depth map; (5) Test the fully supervised algorithm proposed in the previous quarter with acquired depth map.

## 2. Task 1: Development of A Novel Multi-Camera Stereo Vision System for Pipeline Inline Inspection

### 2.1 Post-processing the depth map: Hole Filling Algorithm

In the previous quarter, 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 quarter 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. The method was originally published by Levin et al [1] 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 similar to using Gaussian pyramids, although quantitatively different. The algorithm accomplishes this by minimizing the sum of 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.

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

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

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

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:

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

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 segmentations.

The resulting depth map is then multiplied with the maximum depth value in the original normalized depth map to obtain the final filled output as shown in Figure 2 where the holes in the depth map are marked by red circles.

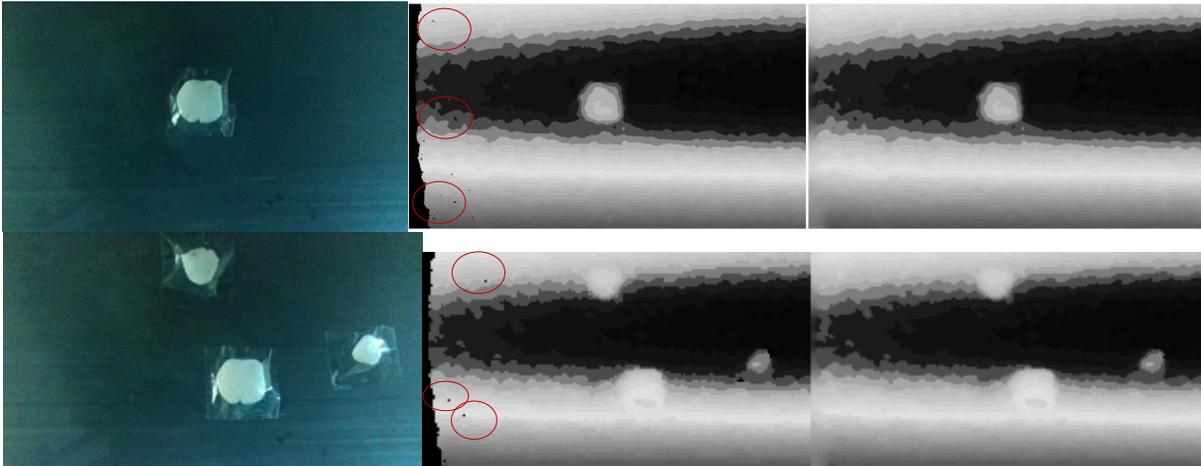


Figure 2: Depth map hole filling demonstrations: (Left): RGB Image; (Middle): Depth map with holes marked by red circles (Right): Filled depth map

## 2.2 Design Proposal for the in-line inspection tool

In this section we propose a modification to the multi-camera design of the ILI tool. 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. This proposal presents a set of design complications. Moreover, ILI tools with cameras that are not oriented towards the pipe wall present poor detection capabilities in the depth map, as shown in the Figure 3.

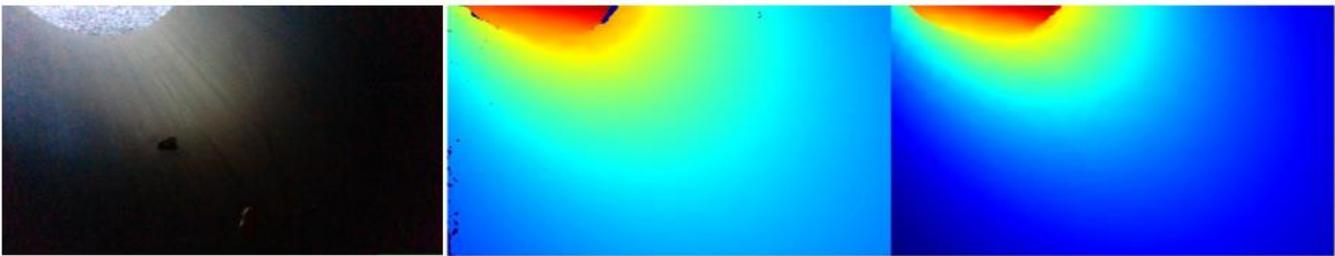
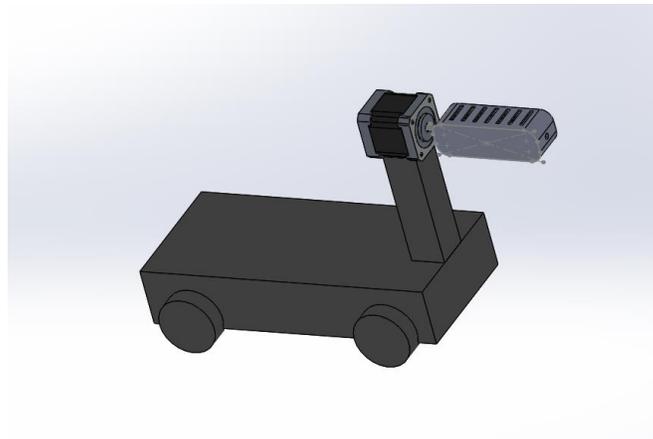


Figure 3: Demonstration of angled view: Left: RGB image; Center: Original depth map; Right: Filled depth map

The above figure has a nut that is 2cm long and has approximate depth of 1cm. The depth map does not show any indication of the object being present, despite it being much larger than the lowest pixel resolution provided by the depth map.

A simpler design is therefore proposed. The fixed multi-camera system is replaced by a rotating single camera system. The rotation is about an axis parallel to the axis of the pipe, so that the camera is always facing normal 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. A representative model of the proposed ILI prototype is shown in Figure 4.



(a)



(c)

Figure 4: (a) CAD Prototype model representative of the single-camera rotating system (b) 3D Printed camera housing

The proposed system is designed to communicate directly with a host computer, which would perform all

the computationally heavy machine learning tasks. An on-board Raspberry Pi client computer receives depth, odometry and image data from the RealSense D435i camera and transmits this to the server. The camera rotation is enabled by a stepper motor, which can offer precise angular control. The motor is driven using an Arduino microcontroller connected to a driver chip, as shown in the Figure 5.

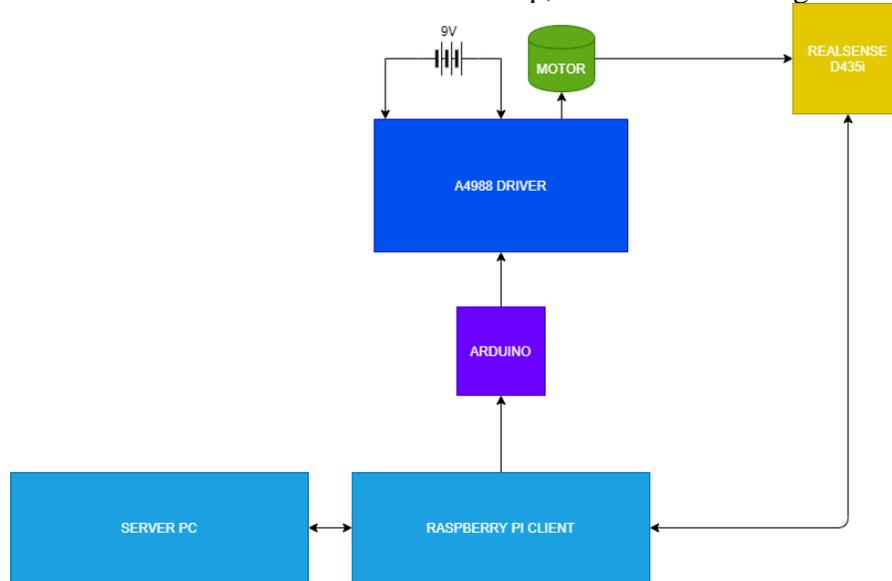


Figure 5: Proposed system architecture for data transmission and camera housing control

### 2.3 Demonstration: RGB-D 3D Reconstruction of a pipe using ORB Features

The following section uses an opensource ROS (Robotic Operating System) based toolkit, RTAB-Map (Real-Time Appearance-Based Mapping) [2], [3] to construct a 3D reconstruction of the pipe sample using the data stream obtained from the RGB-D camera. This is a graph-based Simultaneous Localization and Mapping (SLAM) approach to create 3D reconstructions of scenes in real time, using a loop closure detection approach.

RTAB-Map follows the process below to compute a 3D Reconstruction:

- Loop closure detection: Loop closure detection is used to detect whether the frame that is currently being processed comes from a previously visited location and is done using a bag of visual words approach. A computational strategy has been developed in [3] to ensure that the memory consumption for long-term and scalable loop closure detection does not exceed real-time computational requirements.
- Feature extraction: A set of Oriented Fast and Oriented BRIEF(ORB) features in each frame are used from OpenCV to incrementally construct a dictionary. The ORB feature descriptors must be manually passed as a “DetectorStrategy” parameter to the terminal, as shown in the script attached below. From the set of detector strategies from OpenCV, ORB can be chosen by setting “--Kp/DetectorStrategy=2” when launching RTAB-Map. ORB is a fusion of the FAST keypoint detector and the BRIEF descriptor and provides an alternative to the patented SIFT and SURF descriptors in OpenCV in terms of matching performance. The method first uses Feature from Accelerated Segment Test (FAST), a corner detection method, to extract keypoints from the image. The resulting keypoints are ranked according to the Harris corner response and non-max suppression algorithm, to filter weak corners. These corners are then used to create feature descriptors. Binary strings are efficient descriptors for the extracted keypoints. These descriptors are known as Binary Robust Independent Elementary Features (BRIEF). Each detected keypoint would be encoded by a descriptor, and it acts as a differentiator between these keypoints. The neighborhood around each feature is known as a patch. BRIEF descriptor calculations are fast, making them well-suited for real-time SLAM. BRIEF takes in a Gaussian smoothed image patch and selects a set of  $n_d$  location pairs, which would correspond to the length of the feature vector. A test  $\tau$  on patch  $p$  of size  $S \times S$  is

performed:

$$\tau(p, x, y) := \begin{cases} 1, & p(x) < p(y) \\ 0, & \text{otherwise} \end{cases}$$

where  $p(x)$  is the pixel intensity in a smoothed version of  $p$  at  $x = (u,v)$ .

- Feature matching: Feature matching across two frames is done by nearest neighbor search [4] with nearest neighbor distance ratio (NNDR)[5] using the extracted BRIEF descriptors.
- Motion prediction: A motion model is used to predict where the features should be in the current frame, based on the previous motion transformation. This limits the search window for feature matching to provide better matches in environments with repetitive textures.
- Motion Estimation: When correspondences between two frames are computed, Perspective-n-Point (PnP) RANSAC algorithm in OpenCV is used to compute the transformation of the current frame from the features obtained in the preceding step. A minimum number of inliers is required to accept the transformation, and this is set using the argument “Vis/MinInliers”. A lower threshold for this parameter leads to a greater chance of being able to accept consecutive frames in textureless regions, at the cost of matching quality. With the estimated transformation, a pose update is performed. In the next frame, the algorithm matches features from the previous frame and uses the matching features to merge the two images.

The RTAB-Map algorithm was chosen to demonstrate 3D reconstruction as it is scalable and works in real time. Potential pitfalls of the RTAB-Map feature matching approach are in textureless regions, where finding feature descriptors can be challenging. However, the machine learning algorithm will not directly use the RTAB-Map approach to detect defects. Instead we capture images as they come in, and store defective regions in the form of raw coordinate data. Figure 6 shows a demonstration of the pipe sample reconstructed using RTAB-Map. RTAB-Map works well provided it gets enough feature matches and struggles to perform in textureless regions, as the number of features being captured per frame was less than required to get good matches across frames. It can capture defects as unique feature detections and is demonstrated in Figure 6(a) and 6(b). Movement of the camera needs to be smooth, as jerky movements tend to disrupt the matching process and distort the surface reconstruction across time.



Figure 6: Demonstration of sample pipeline 3D reconstruction using RTAB-Map

## 2.4 Sensitivity Analysis for the Intel RealSense Camera

### 2.4.1. Background

The camera used in this project is Intel RealSense Camera D435i which uses advanced algorithms to process raw image streams from the depth cameras and computes high resolution 3D depth maps without the need for a dedicated GPU or host processor. It has a right imaging camera and left imaging camera, IR projector and a RGB module. The stereo camera projects Infrared Light pattern onto the scene to increase the texture of the low texture scenes. It also has two sensors on the left and right, also known as imager which are at a small distance apart. The stereo camera takes two images from the sensor, compares them and calculates the depth using triangulation method. Benefit of using infrared light is that it can work on any lightning

conditions for perceiving the depth. The distance(baseline) that can be measured by this camera depends on the spacing between the two sensors- wider is the baseline, camera can see farther objects and determine its depth. Figure 7 shows Intel Realsense Depth camera with its visible parts such as IR module, left and right cameras and the RGB module.



Figure 7: Intel Realsense D435i Camera System (adopted from <https://www.intelrealsense.com/zh-hans/depth-camera-d435i/>)

Sensitivity analysis is performed to test 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. In this project, the performance of the Intel Realsense Camera for visualizing an object is analyzed. The analysis is done by varying some of the camera parameters and determining the accuracy of detecting an object from a depth map.

The considered parameters include:

- Gain: Amplifies the entire image signal
- Exposure: the amount of light per unit area reaching the surface of an electronic image sensor.
- Laser Power: Power provided to the IR projector
- DS\_SECOND\_PEAK\_THRESHOLD: Determines how different the second highest matches can be from the first highest matches for stereo depth computation
- DS\_NEIGHBOR\_THRESHOLD: Determines the 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): 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)

## 2.4.2. Methodology

### Contour extraction for the defect from RGB and depth image

Contour extraction is used to extract the region of interest containing the detected object boundary from the depth map. A contour is a curve joining all continuous points having the same color or intensity. Contour detection is an image segmentation method which requires very less computational time and is a rough approximation of any unsupervised or supervised machine learning algorithm which is going to be used for our image segmentation purpose. It is helpful to detect any shape in an image by extracting curves from 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 its luminosity, texture, perceptual grouping from background. Mathematically, the edge/boundary detection is performed by convolving with local filters such as Sobel, Prewitt, Laplacian, Canny and then finding the pixels with the highest gradient magnitude with respect to the local neighborhood pixels.

## Contour formation for depth map

Prior to performing contour detection, hole filling is performed on the depth map using the built-in 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 horizontal direction ( $G_x$ ) and vertical direction ( $G_y$ ). From these, we get edge gradient and angle of the pixels.

$$G = \sqrt{G_x^2 + G_y^2}$$
$$\theta = \text{atan}\left(\frac{G_y}{G_x}\right)$$

Then, a non-max suppression is done on unwanted pixels for the edge. In the end, the threshold value that was 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. Image moments from the contour are used to find its centroid, which is used as a reference measurement.

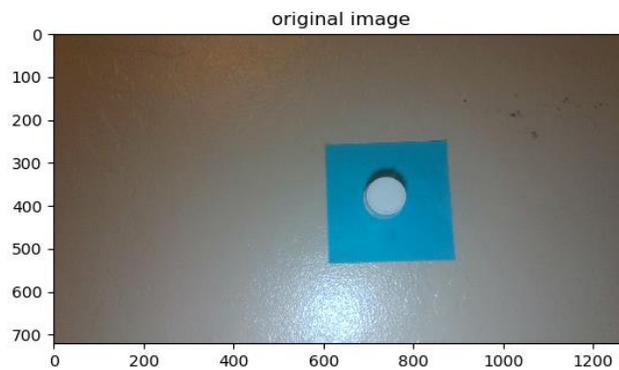


Figure 8: Original RGB image

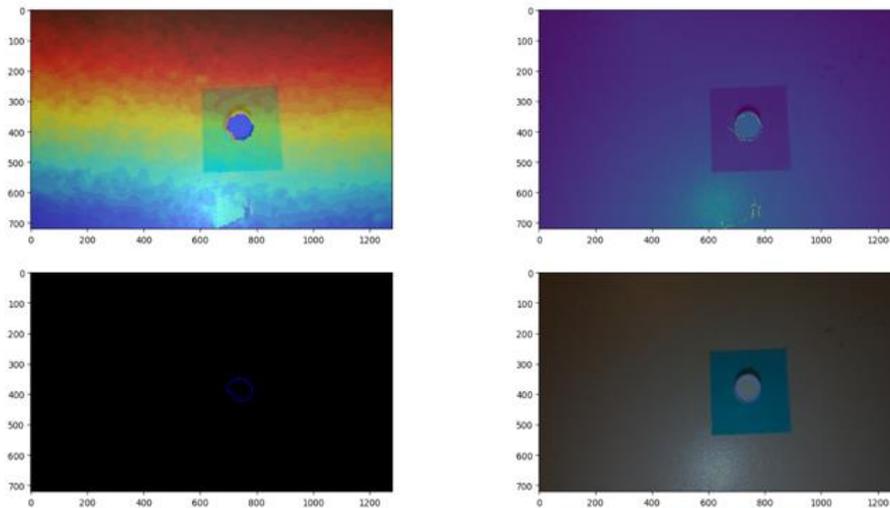


Figure 9: Contour extraction from the depth map

Figure 8 shows the original RGB image capture by the camera. In Figure 9, 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. The lower right image shows the contour superimposed on the RGB image for comparison.

### Contour formation for 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 in such a way that the region of interest can be easily segmented out using classical thresholding operations. The binary image is used to find the contours.

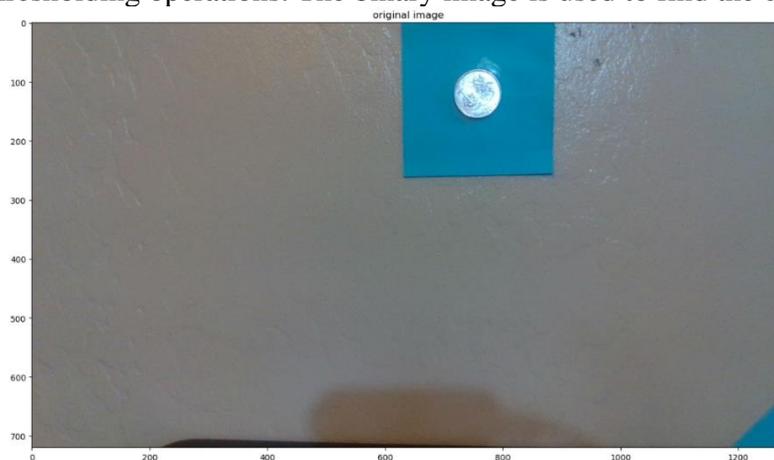


Figure 10: Original RGB image of a coin

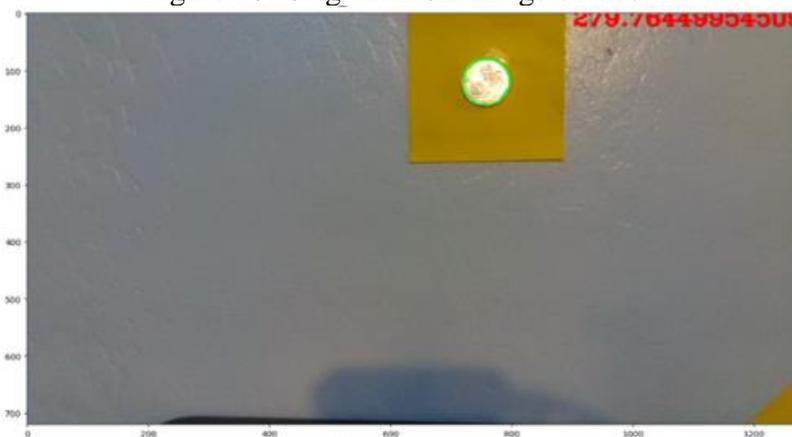


Figure 11: Contour extraction from RGB image

Figure 10 shows the original RGB image. This image is overlaid with the extracted contour in Figure 11, where the green contour line represents an approximation to the region boundary of the object of interest.

### Measuring the area of contour in cm from depth map

In this method, the ground truth area was measured manually, and the area of depth map was converted from the pixel space into the real world unit using the intrinsic parameters of the camera. The camera intrinsics do not provide the 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. Figure 12 shows the basic geometry of a camera lens.

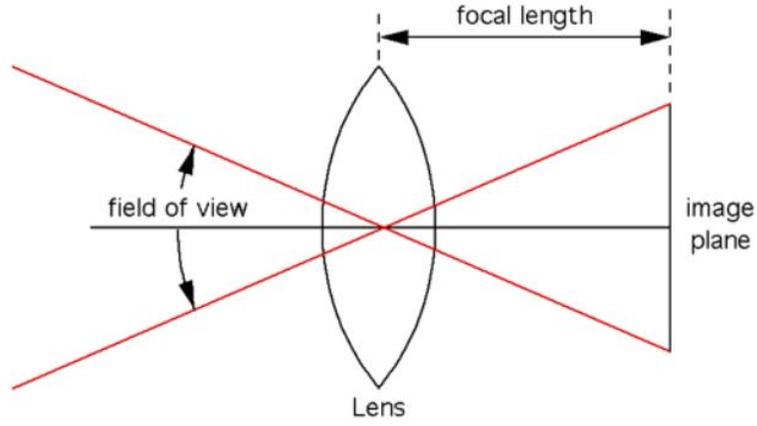


Figure 12: Geometry of a camera lens

From Figure 12, we have

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

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

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 its origin and orientation of the depth camera.

Horizontally,  $\tan\left(\frac{HFOV}{2}\right) = \frac{\frac{FrameWidth}{2}}{depth}$

Vertically,  $\tan\left(\frac{VFOV}{2}\right) = \frac{\frac{FrameHeight}{2}}{depth}$



Figure 13: Simplified geometry of the Depth Camera

Figure 13 shows the simplified geometry of a depth camera which gives us the height and width of one pixel as:

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

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

$$Area(cm^2) = Area(Px) * W(1px) * H(1px)$$

where  $Area(cm^2)$  represents the area of the object (area of the cap in this experiment) in real world unit.  $W(1px)$  and  $H(1px)$  represent the width and height of a pixel in real-world unit, respectively. The pixel area  $Area(Px)$  is obtained by computing the zero-order image moment  $M_{00}$  on the binary image, for the region corresponding to the specific contour in consideration:

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

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.

### Testing using RGB image

To verify whether the algorithm works as expected and provides area computation estimates close to the ground truth, we first apply it on 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. Therefore, once we verify that the area calculation method returns reasonable values with respect to the ground truth, we apply it to the depth map.

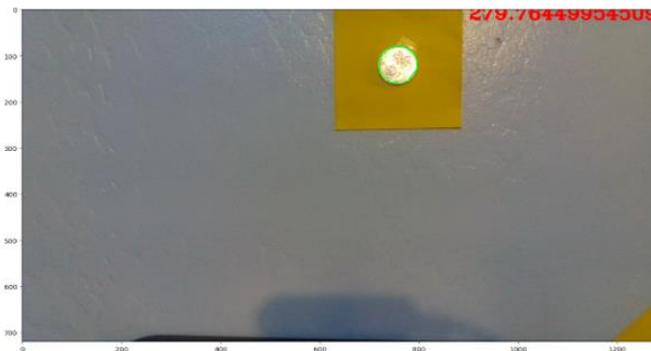


Figure 14: Contour approximation of RGB image

Figure 14 shows the contour detection of the object of interest in the RGB image. The measured contour area came out to be  $4.44 \text{ cm}^2$  using the calculations detailed above. The ground truth area is  $4.84 \text{ cm}^2$ , which leads to an error of 8%. The calculated horizontal and vertical fields of view are  $69.2^\circ$  and  $42.4^\circ$  respectively.

### Testing using depth Map

This experiment was performed to show that the proposed algorithm for conversion of area in pixels to real world unit gives consistent area estimation by calculating the area of the same object from the depth map at varying distance from the camera. Figure 15 shows the contour extraction from the depth map.

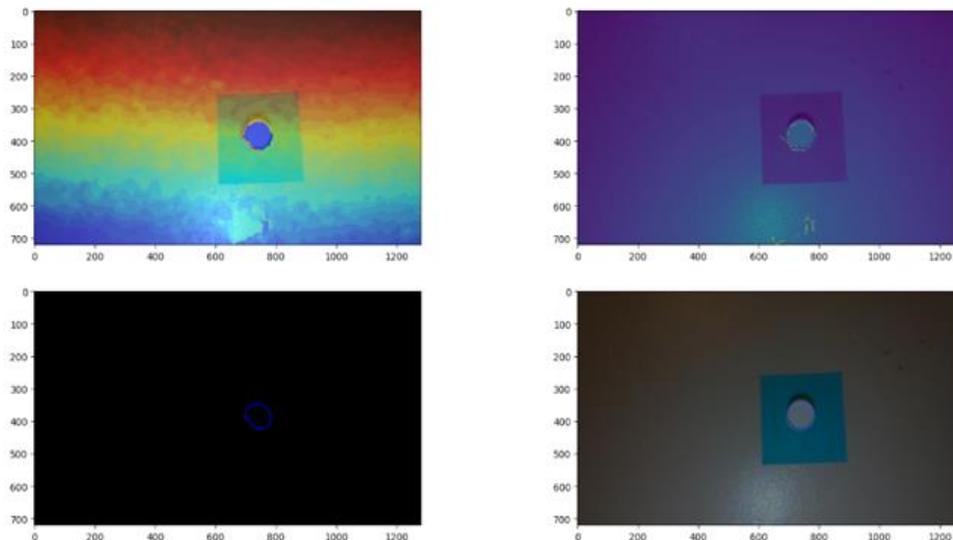


Figure 15: Contour area calculation for depth map

Table 1: Contour area at varying distance

Distance in cm	Area of contour in pixels	Area of contour in cm <sup>2</sup>
23.1	5268	3.277
24.4	4686	3.2523
27.2	3718	3.2
28.1	3545	3.26
29.9	3116.5	3.24
30	3082	3.23

Table 1 shows that even though the distance of the camera from the object is varying, the area in cm<sup>2</sup> is the same. There is slight variation due to the sensor temporal noise. The ground truth area for this image is 3.7994 cm<sup>2</sup>. The measured area of the contour from the depth map is approximately 3.2 cm<sup>2</sup>, and the error is around 15%.

### Calculation of the area estimation error

Since, the area error needed to be checked against a wide range of parameter values for any particular 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 cm<sup>2</sup> and error was found with respect to the ground truth area. Ground truth area was measured as 3.7994 cm<sup>2</sup>.

The error percentage is calculated as:

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

### 2.4.3. Results and Discussions

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 visual comparison.

#### Error in area estimation with respect to change in Gain

Gain is the amplification of the image signal obtained from the image sensor.

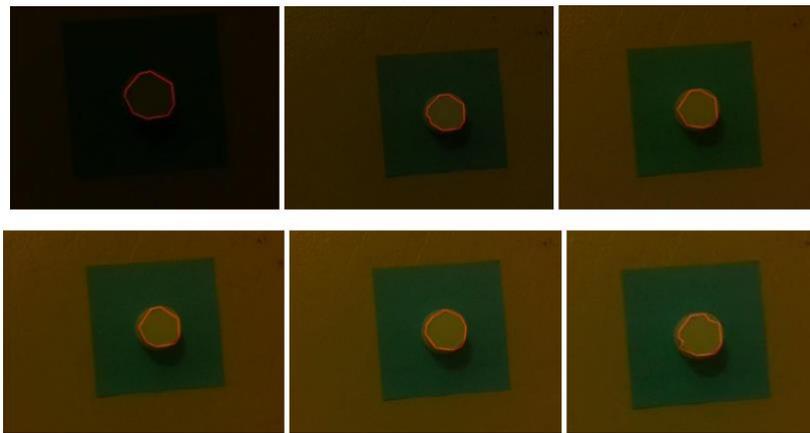


Figure 16: Representative contours for various values of Gains. From top Left: Gain = {16,22,27,32,37,42}

Figure 16 shows contours extracted from the depth map superimposed on a RGB image for comparison. The images are obtained while varying the camera gain value.

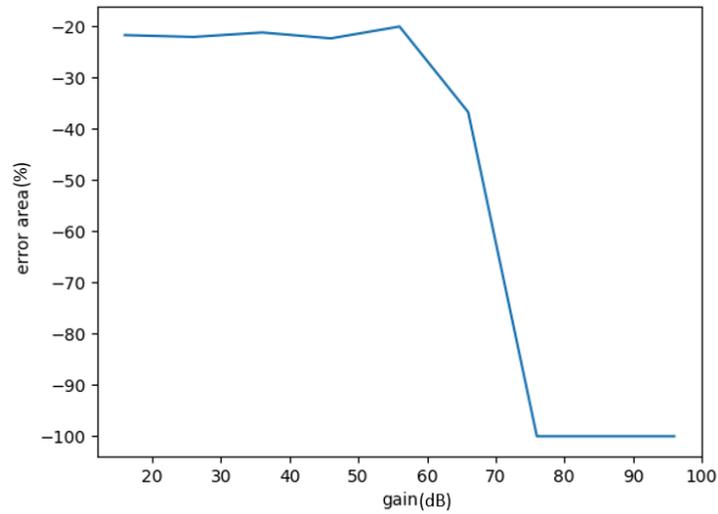


Figure 17: Contour area error variation with respect to camera gain

From Figure 17, it can be concluded that the depth map performs very well in the low Gain range (16-55). More gain introduces electronic noise, which is the reason for depth map distortion for gain above 55dB. Please note that any error beyond 100% was clamped for better readability.

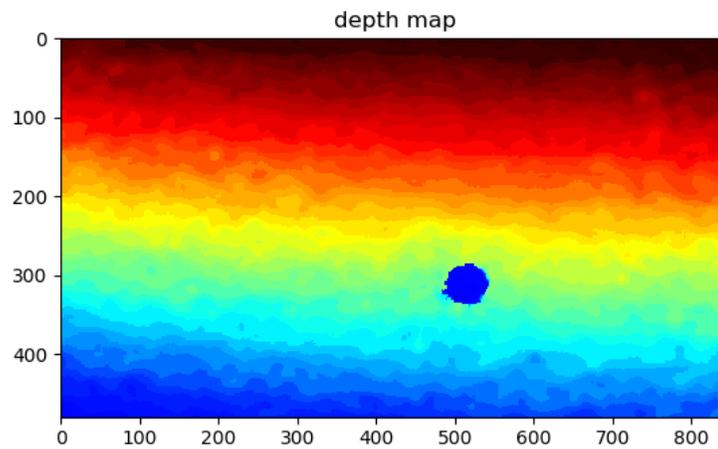


Figure 18: Depth frame for Gain in range (16-55)

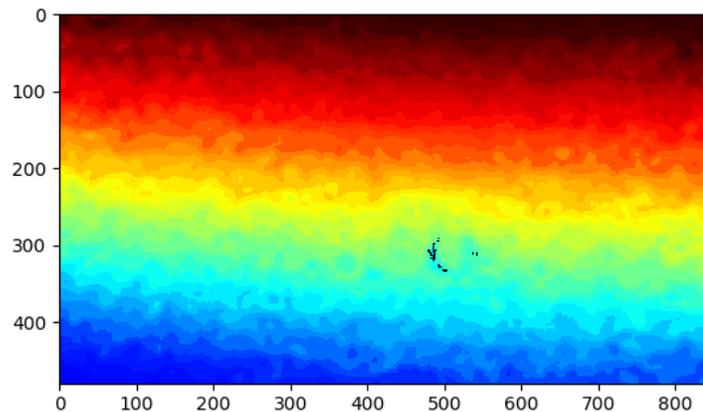


Figure 19: Distorted Depth frame for Gain beyond range (16-55)

Figure 18 shows the raw depth map after hole filling for Gain in range (16-55) and Figure 19 shows the raw depth map after hole filling for Gain outside the range (16-55) which shows the object is no longer detectable.

### Error in area estimation 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 Lux ( $W/m^2$ ) seconds. 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.

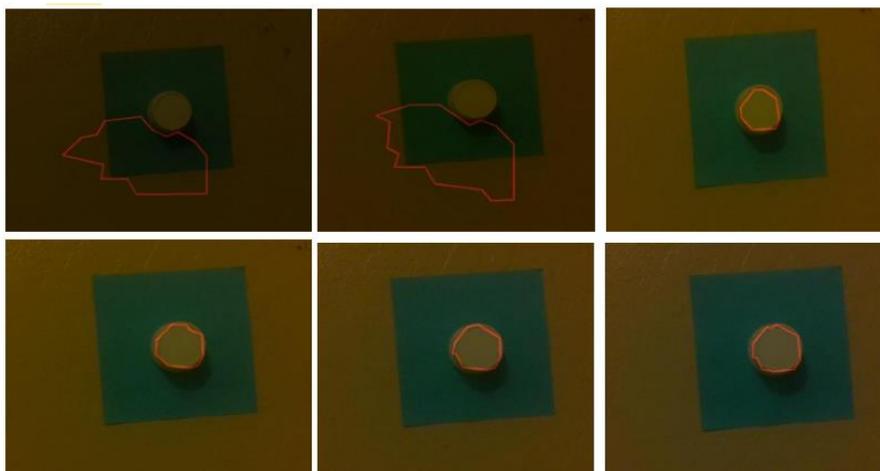


Figure 20: Representative contours for various values of Gains  
From top Left: Exposure= {0, 5000,10000,15000,20000,25000}

Figure 20 shows the contours extracted from the depth maps when the exposure was changed from 0-25000. The depth map on the first two images shows a contour that does not match the object. This is because the exposure of the camera was too low to produce the shape of the object, as shown in the last 4 images in figure 20. On the other hand, exposures greater than 50,000 make the image overexposed, which also leads to poor detection and area computation. The final plot of the variation of error with respect to the exposure is shown below:

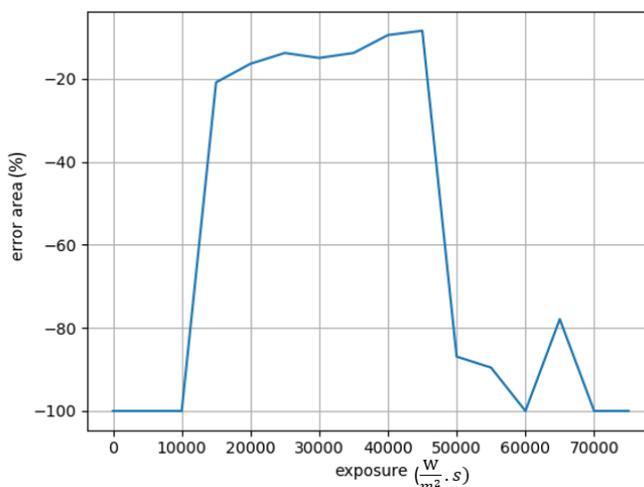


Figure 21: Contour area error variation for varying Exposure

From Figure 21, it can be concluded that the depth map gives a good estimation of the area when the exposure lies in range (10000-45000). Exposure depends on the light incident on the camera. More light creates a brighter image for both left and right camera which loses the depth information. Similarly, for less light, the image is too dark which again leads to loss of depth information. So, it works only in a particular range depending on the lighting conditions. Figure 22 shows the depth map obtained when the exposure is beyond the range (10000-45000).

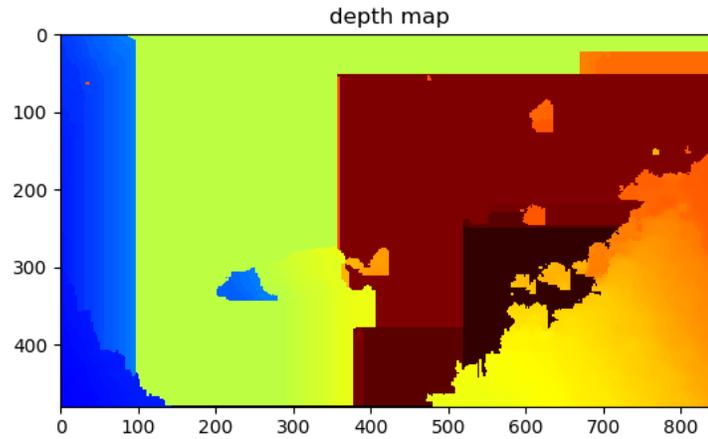


Figure 22: Distorted raw Depth Frame for exposure<10000 and exposure>45000

### Error in area estimation with respect to change in Laser Power

Laser Power is the power provided to the IR projector. The watt (symbol: W) is the unit in Fig. 24. Figure 23 shows the contours extracted from the depth maps when the exposure was changed from 0-50.

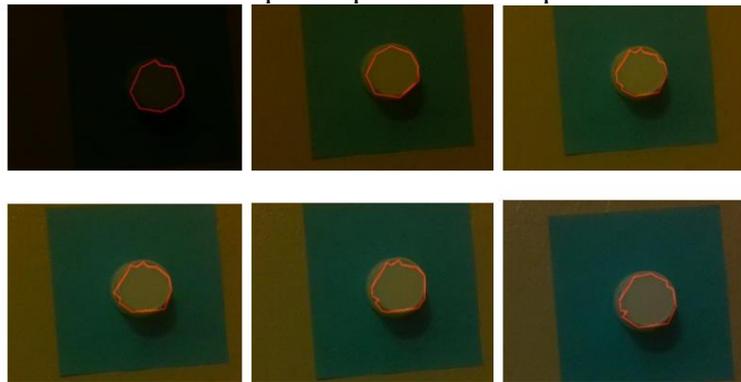


Figure 23: Representative contours for various values of Laser Power (From top Left: Laser = {0,10,20,30,40,50})

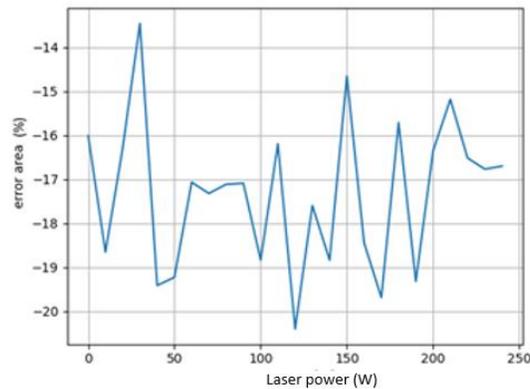


Figure 24: Contour area error variation for varying Laser Power

Figure 24 shows that the contour error variation is insignificant for the entire range of laser power. Laser power is mainly required for long range viewing and for viewing in the low texture and low brightness environments. Since the object was at constant distance and sufficient lighting conditions, the effect of laser power on area estimation is not significant.

### Error in area estimation with respect to change in DS Second Peak Threshold

DS Second Peak Threshold is a parameter used in the stereo-matching algorithm for depth calculation. Deep-Sea (DS) is an ASIC (Application-Specific Integrated Circuit) which is used for depth computation.

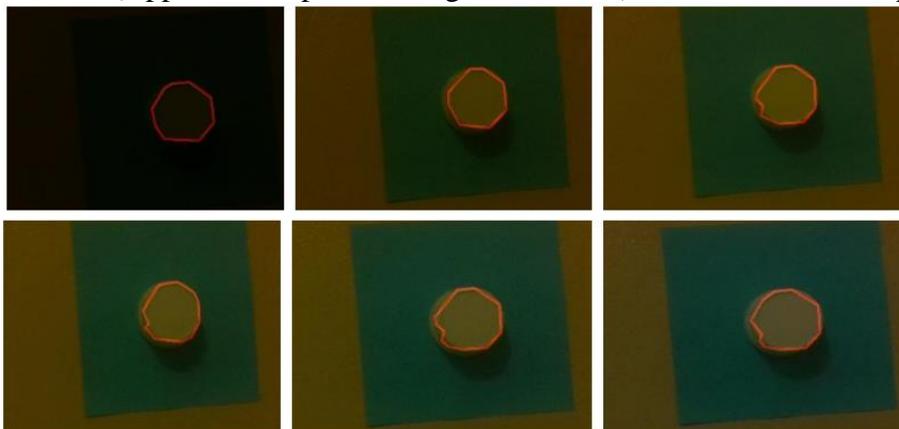


Figure 25: Representative contours for various values of DS Second Peak Threshold  
From top Left: DS Second Peak Threshold= {0,50,100,150,200,250}

Figure 25 shows the contours extracted from the depth maps when the DS Second Peak Threshold was changed from 0-250.

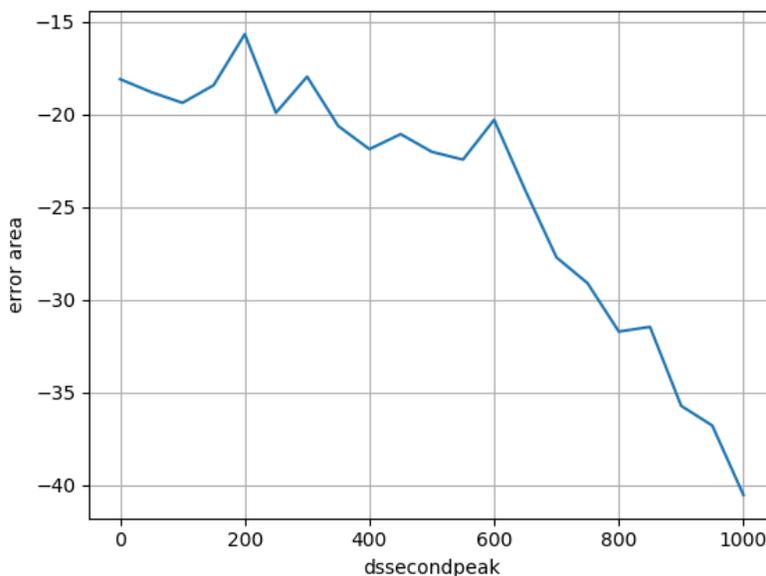


Figure 26: Contour area error variation for varying DS second peak threshold

The DS second peak threshold tells how different the first peak and second peak matching between left and right camera need to be. As this threshold increases, more matches are regarded as aliasing and thus depth result is 0 in most pixels. Thus, as shown in Figure 26, it works well only till DS second peak threshold value of 600. Figure 27 shows that the depth map gets distorted for a high DS second peak threshold.

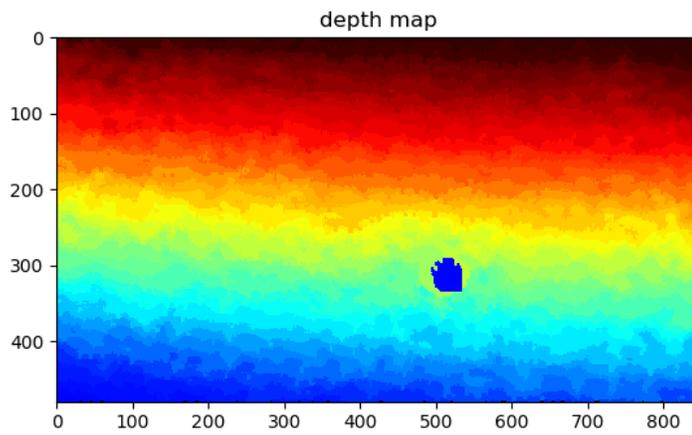


Figure 27: Aligned Depth Frame for DS second peak beyond 600

### Error in area estimation with respect to change in DS Neighbor Threshold

The DS Neighbor Threshold is a parameter used in the computation of the stereo matching algorithm for the depth map. Figure 28 shows the contours extracted from the depth maps when the DS Neighbor Threshold varies from 0-250.

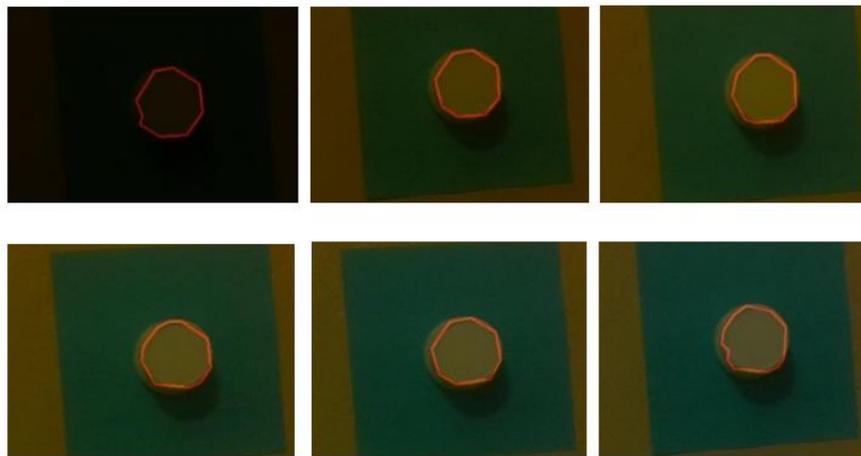


Figure 28: Representative contours for various values of DS Neighbor Threshold  
From top Left: DS Neighbor Threshold= {0,50,100,150,200,250}

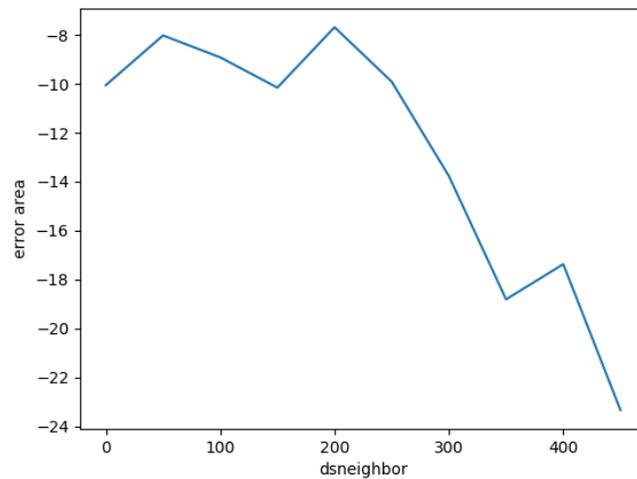


Figure 29: Contour area error variation for varying DS Neighbor Threshold

The plot in Figure 29 shows that the depth map gives good estimate of area when the DS Neighbor Threshold lies in range of 0-400. The increasing neighbor threshold ends up producing many missing values in the depth map computation, rendering a poor quality depth map as shown in Figure 30.

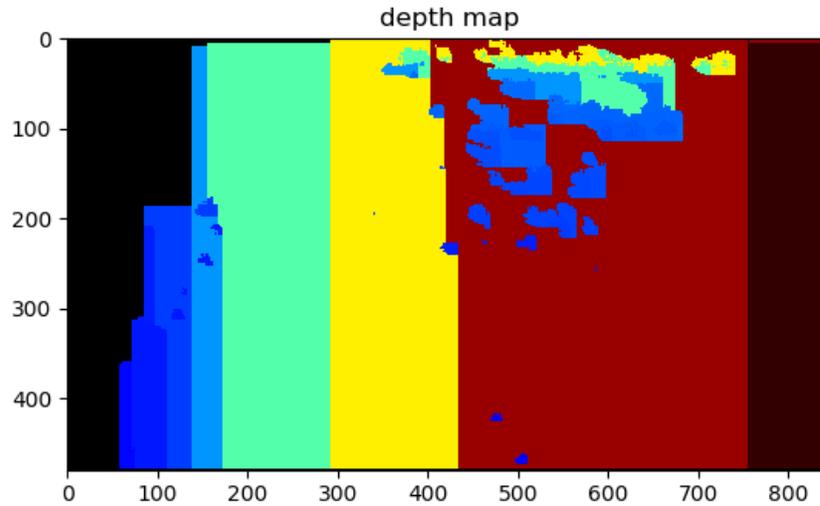


Figure 30: Distorted Depth map for DS neighbor threshold beyond 400

### Error in area estimation with respect to change in Disparity Shift

Disparity Shift changes the minimum and maximum depth that can be registered from the camera. It is measured in pixels. Figure 31 shows the contours extracted from the depth maps when the Disparity Shift was changed from 0-50.

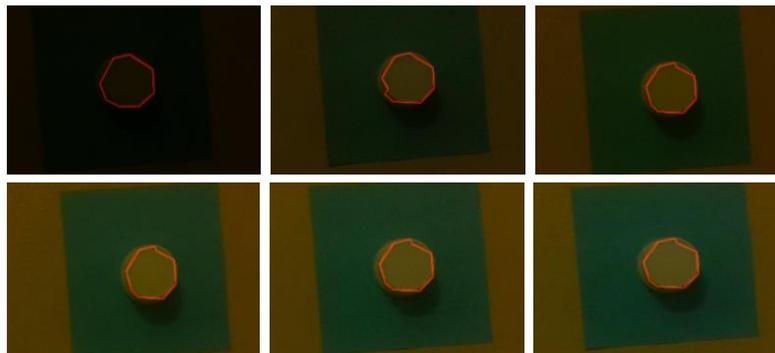


Figure 31: Representative contours for various values of Disparity Shift  
From top Left: Disparity Shift= {0,10,20,30,40,50}

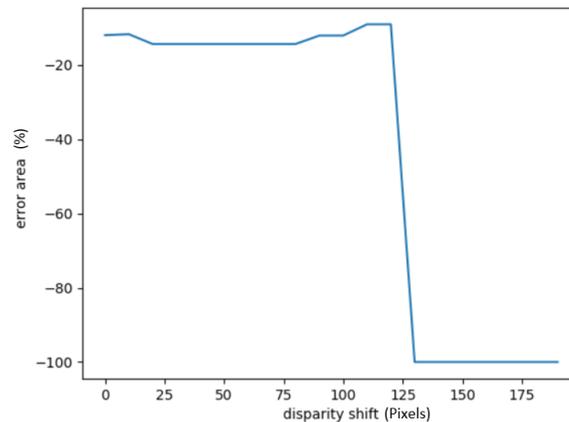


Figure 32: Contour area error variation for varying Disparity Shift

Figure 32 shows that the depth map gives a good estimate of object area for disparity shift range (0-120). Beyond 120, 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 as shown in Figure 33.

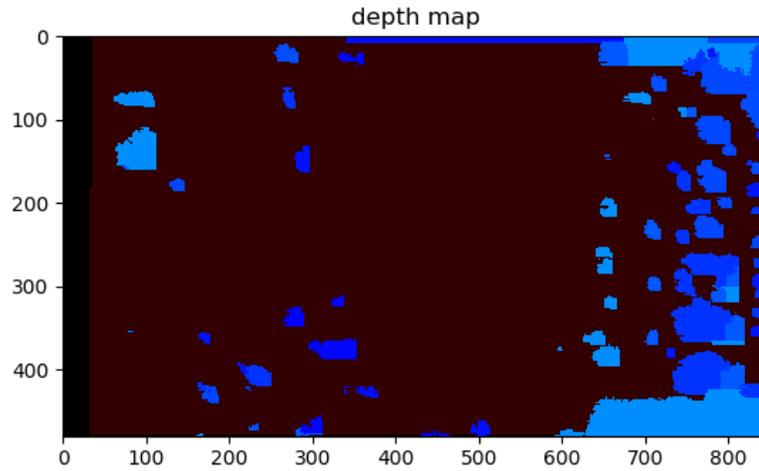


Figure 33: Distorted raw depth map beyond disparity shift of 120

### Error in area estimation with respect to change in Resolution

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 2 shows the variation of error in surface area when the resolution of the image varies.

Table 2: Variation of error in area computation with resolution

Resolution	Error in area
848x480	-13.29 %
640x480	-11.762%
1280x720	-11.954%

### Error in area estimation with respect to change in Object Surface Area

To change the surface area of the object, a custom object is created using a clay dough. The shape of the object is a cuboid. The length and width of a custom object is changed while the thickness is kept constant. In this experiment, objects of varying sizes are placed in front of the camera and the area of the object was found using the contour approximation method.

Table 3: Variation of error in area computation with surface area change

Length(cm)	Width(cm)	Thickness(cm)	Error%
1.9	1.5	1.3	8.804
1.4	1.4	1.3	-4.329
0.9	0.5	1.3	9.64

Table 3 shows how the error percentage changes as the surface area is changed while keeping thickness of the object constant. The experiment was conducted by gradually reducing the size of the object from (1.9x1.5) cm to (0.9x0.5) cm. The reduction in size of the object does not independently impact the accuracy of measurement given that the thickness of the object 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 limit of resolution determines per pixel area, and this will impact the size of the object that can be detected. Further

work will be done to finalize the results with respect to object size in the next quarter.

### Error in area estimation with respect to change in Object Thickness

Figure 34 shows the contours extracted from depth images for different sizes of object thickness.

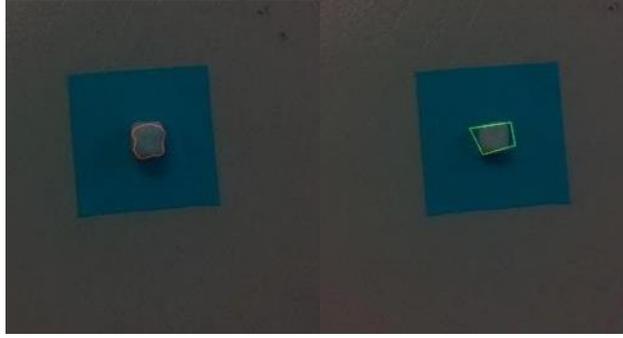


Figure 34: Contour capture at an object thickness of (Left) 1.8cm and (Right) 1cm

Table 4: Variation of error in area computation with change in object thickness

Length(cm)	Width(cm)	Thickness(cm)	Error%
1.5	1.4	1.8	4.4
1.5	1.4	1.	15.6%

As shown in Table 4, we kept the length and width of the object constant and varied the thickness of the object to find the error of area estimation. As expected, the error increases when we reduce the object thickness. 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 1.8cm roughly captures the morphology of the actual defect. However, as the object thickness is decreased to 1cm, the depth map does not preserve the shape of the object, which led to a much higher error.

### 3. Task 2: Integrating the Data Acquired from the Camera system with the Fully Convolutional Fusion Network

#### 3.1 Background review of the Fully Convolutional Fusion Network

First, we briefly describe the fully convolutional network architecture developed in the previous quarter along with a fusion rule algorithm for combining depth and RGB data. The Fully Convolutional Network (FCN) 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. FCN takes inspiration from the classification convolutional network architecture and removes the final fully connected layer and replaces it with an expanding, upsampling architecture. 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 had the VGG-16 backbone network. The VGG-16 network had 5 convolutional blocks, where each block ends with a max-pooling layer.

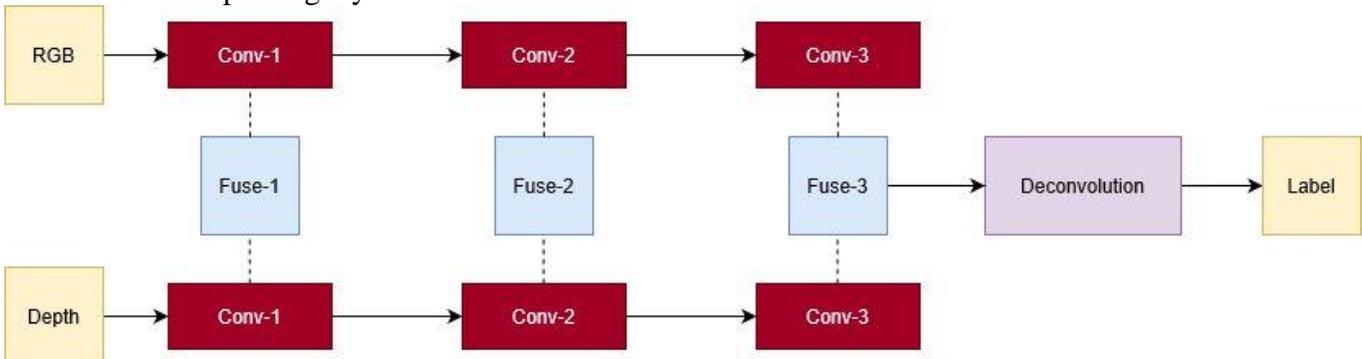


Figure 35: Overall Fully Convolutional Network Architecture

Fusing the two incoming sources of data, the color image and the depth map will be the focus of this section. Fusion can be accomplished in multiple ways:

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

Our approach currently implements method (ii) to fuse data at the feature level. We use a non-linear weighted combination of the max pooling layer outputs from each convolutional block as follows:

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

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

where K is a 1x1 convolutional filter set.

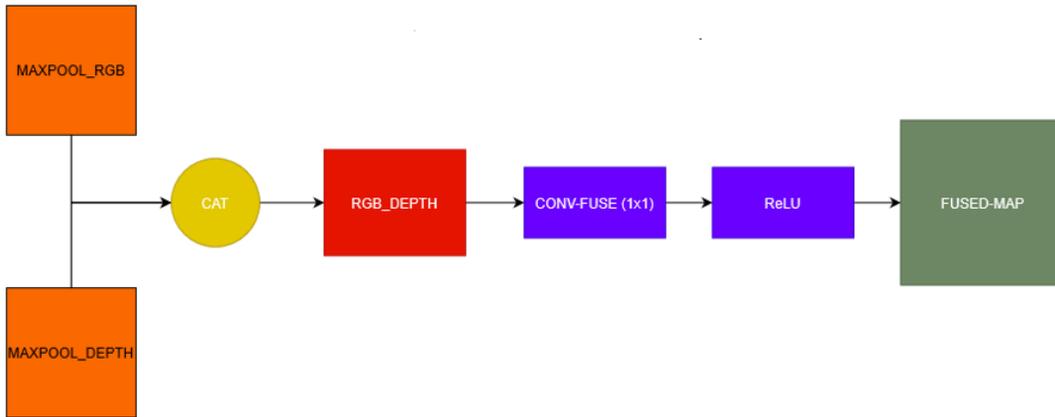


Figure 36: Fusion rule architecture

The initial max pooling outputs 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. The max pooling layer 1 in the network consists 64 channels each. 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.

Data imbalance has been a pervasive problem in the image segmentation problem. We observe that the background class is the most dominant class in the scene for most examples, and the object that needs to be detected only has a few samples compared to the total number of samples available. Objects with small sizes when segmented by a deep network often lose their morphological details, and the majority class ends up bleeding into the object boundary, leading to noisy, poor quality segmentation, as will be shown in the initial results. We use the Lovasz-Extension approach from the previous quarter to address this class imbalance.

### 3.2 Data acquisition for Fully Supervised Defect Detection

Preliminary work on data acquisition for this quarter involved using the pipe sample with simulated defects types. Two defect types were considered: (i) Rough corrosion (ii) Localized denting. The corrosion defect was simulated using aluminum foils. While the defect does not visually register with the depth map due to its low depth variance from the background, the neural network is expected to capture the minute gradient changes, as the point cloud visualization of these defects clearly showed the surface roughness being captured. The second defect type was made using clay dough, and the depth was visually distorted by this defect. The RGB-D pair was extracted for five samples, as shown below in Figure 37.

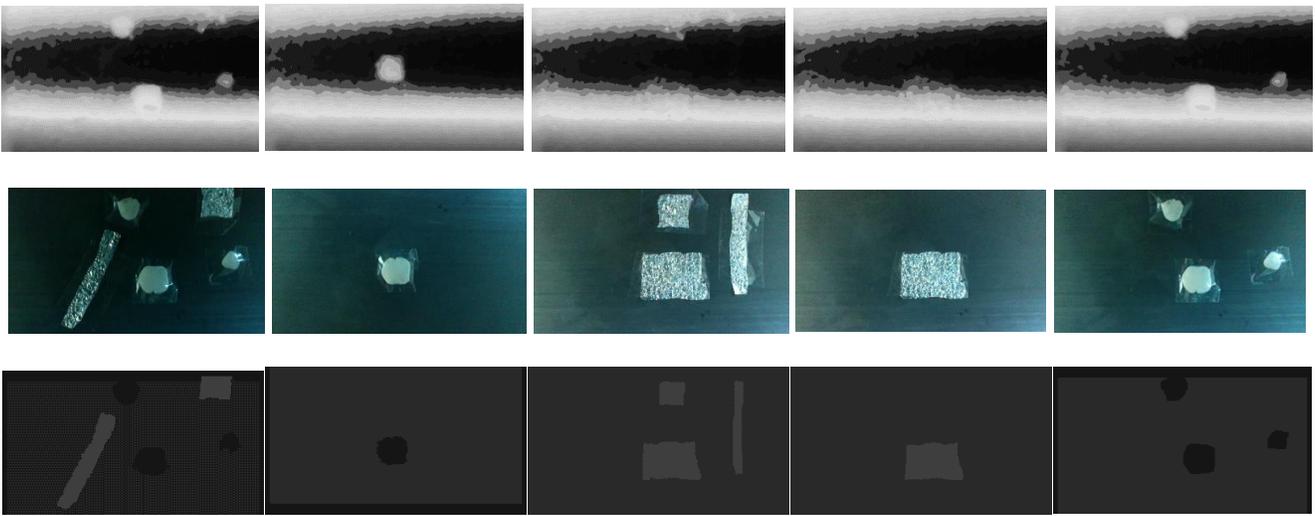


Figure 37: Captured samples for defect detection: (Top): Depth Map ; (Middle): RGB Image ; (Bottom): Label

The five acquired samples were processed using the depth map hole filling algorithm. MATLAB's image labeler was used to create ground truth for these samples. Once the ground truth was created, the data was augmented using the following operations: Random flipping, affine transformations using a combination of image shearing and scaling and were all resized to a size of (224,224) pixels. The transformations in the values of depth was not accounted for in this case, as we are not using the algorithm in its current state to measure object sizes.

### 3.3 Demonstrative results

This section describes the results obtained from applying the algorithm to the dataset for segmentation. The augmented data set was split into 400 training and 100 validation samples. The training progress and sample detection results are plotted in Figures 38 and 39, respectively. The validation and training IoU exceeded 90%, as the defect samples were distinct from the background and there was no object clutter or ambiguity. The shapes of the objects were therefore well preserved, and any error in measurement would largely be attributed to errors in the depth map. In the next quarter, more defect types will be introduced, and more complex scene examples will be tested, with the validation set drawn from a totally different set of original samples.

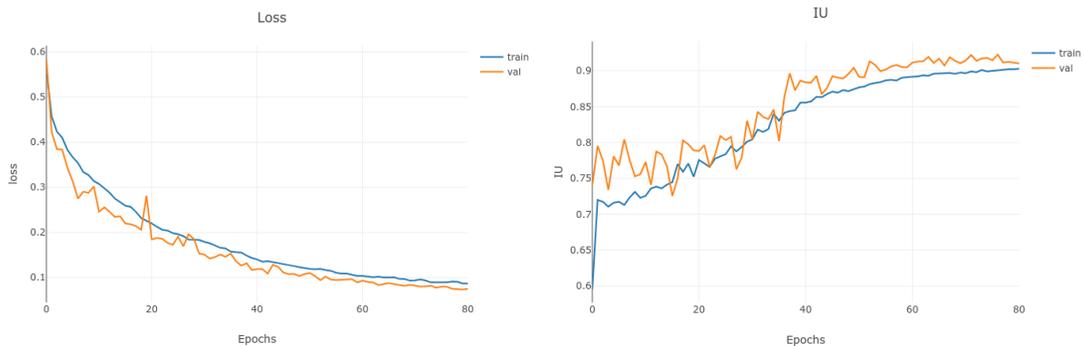
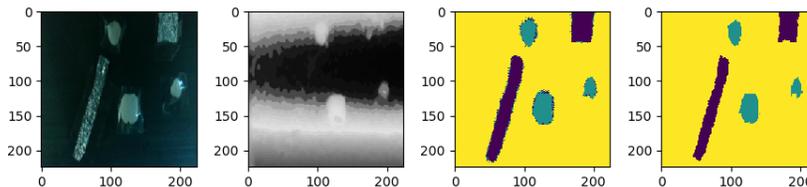


Figure 38: Loss and IU plots across training and validation epochs



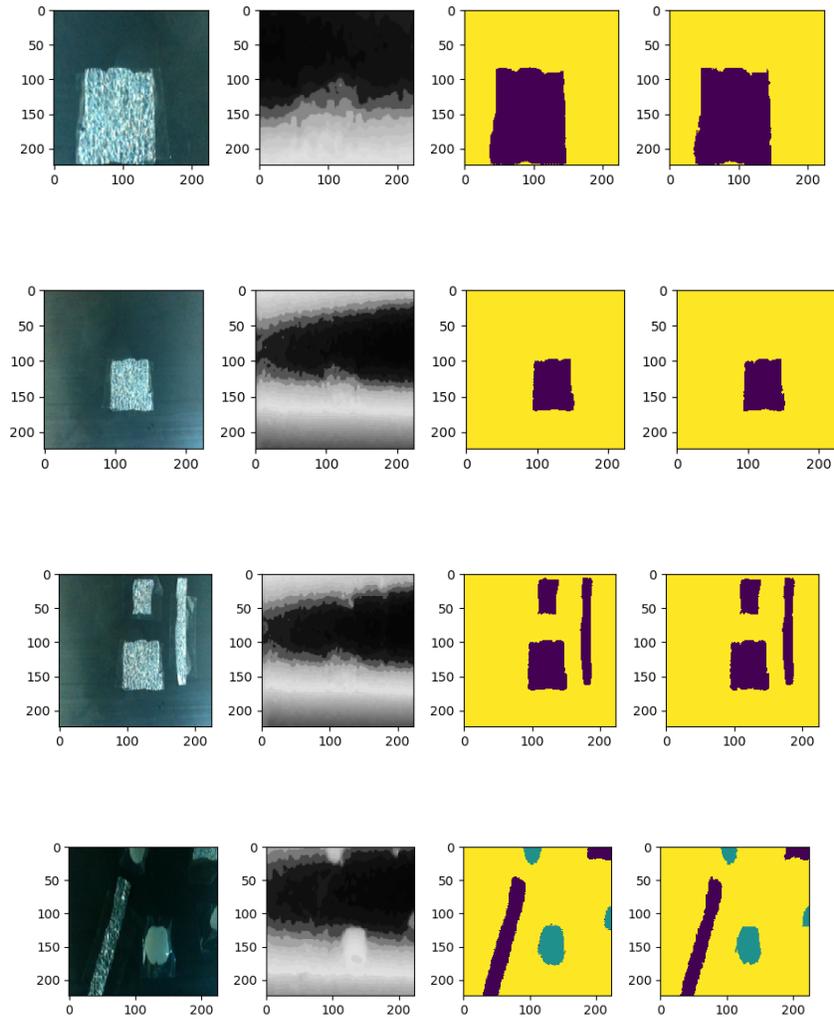


Figure 39: Demonstrative examples of segmentation results (From Left: RGB, Depth, Detection, Ground Truth)

#### 4. Summary and Future Work for the Fourth Quarter

##### 4.1 Task 1

In this quarter, an improvement upon the disparity map hole filling algorithm by using an optimization approach to produce smooth, visually plausible depth maps without missing values was completed. A modification to the design of the ILI tool was proposed. A demonstration of 3D reconstruction using a graph-based SLAM algorithm was implemented using RTAB-Map. Preliminary results on the sensitivity analysis to determine the potential detection and sizing accuracies for various object sizes and camera parameters were produced. In the next quarter, the sensitivity analysis results including estimations in both plane and depth dimensions will be finalized and the optimal camera parameters obtained from the experiments will be summarized.

##### 4.2 Task 2

In this quarter, we integrated the hole filling algorithm from Task 1 to collect new images from the acquired pipe sample with simulated defects. The acquired samples were then used to test on the fully supervised learning algorithm developed in the previous quarter. In the next quarter, more defect types will be introduced and the complexity of dataset will be increased. The resulting images will be tested alongside the currently developed learning algorithm, and more sophisticated fusion methods will be proposed to build upon the non-linear weighted combination fusion block proposed earlier.

## References

- [1] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," in *ACM Transactions on Graphics*, 2004, doi: 10.1145/1015706.1015780.
- [2] M. Labbé and F. Michaud, "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*, 2019, doi: 10.1002/rob.21831.
- [3] M. Labbe and F. Michaud, "Appearance-based loop closure detection for online large-scale and long-term operation," *IEEE Transactions on Robotics*, 2013, doi: 10.1109/TRO.2013.2242375.
- [4] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *VISAPP 2009 - Proceedings of the 4th International Conference on Computer Vision Theory and Applications*, 2009.
- [5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 2004, doi: 10.1023/B:VISI.0000029664.99615.94.