

CAAP Quarterly Report

Date of Report: June 26, 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 quarterly period ending: June 29, 2020

Business and Activity Section

(a) Contract Activity

Discussion about contract modifications or proposed modifications:

None

Discussion about materials purchased:

Intel® RealSense™ Depth Camera D435i

(b) Status Update of Past Quarter Activities

In this quarter, the research team works on Task 1.1, Task 2.1, and Task 2.2 to develop algorithms for automatic defect size estimation using unsupervised image segmentation and stereo vision, and develop supervised deep learning model for object detection and segmentation. Experiments were conducted to verify the effectiveness and accuracy of the developed algorithms.

Student Training Activities

- Kailing Liu (MS student) works on sensitivity analysis to study the effect of lighting condition and camera resolution on the quality of disparity map and accuracy of distance estimation using stereo vision algorithm. (Task 1). Kailing died in a car accident and a slight delay happened in Task 1. We are recruiting new students to work on some subtopics in Task 1.
- Rahul Rathnakumar (PhD student) works on integrating the unsupervised algorithm for image segmentation with stereo vision-based distance estimation for automatic identification of defect size (Task 1) and developing a fully supervised algorithm for RGB-D semantic segmentation (Task 2)

(c) 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.

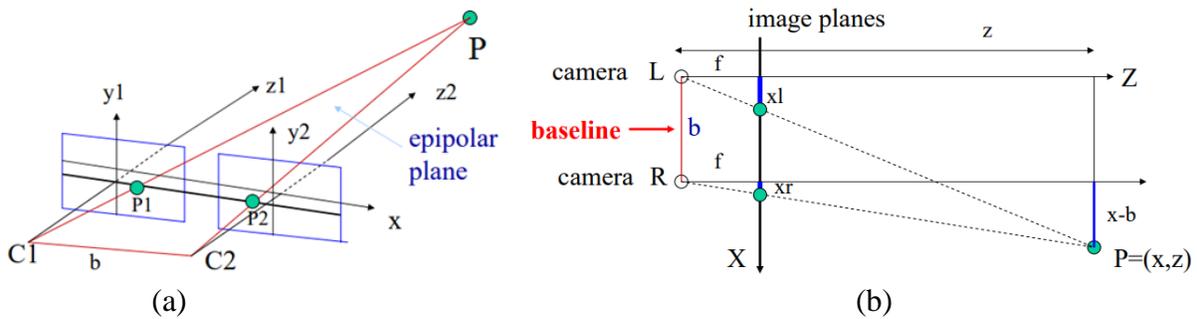


Figure 1: Principle of stereo vision: (a) epipolar plane; (b) triangulation

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 3rd quarter is to: (1) Improve the quality of disparity maps generated by the stereo vision algorithm by filling in the uncertain regions of the map; (2) Test the acquired image and disparity map using the unsupervised segmentation algorithm developed from the previous quarter; (3) Develop a fully supervised algorithm for RGB-D semantic segmentation.

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

2.1 Post-processing the disparity map: Filling in missing values using Gaussian Pyramids

The disparity map obtained from the previous quarter using the Semi-Global Matching (SGM) algorithm contained invalid regions with no disparity values which could compromise the segmentation results. As a first step, the disparity map was post-processed using the classical technique that makes use of Gaussian pyramids to downsample the original image followed by an upsampling the lower resolution representation.

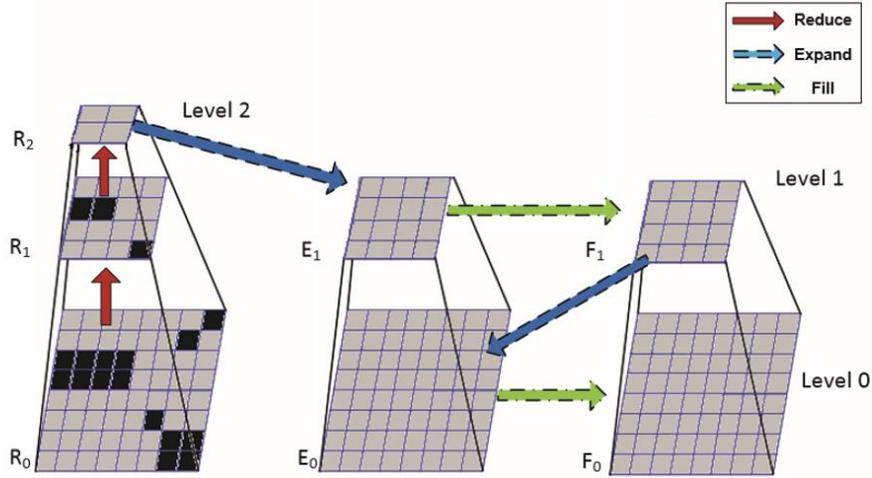


Figure 2: Filling missing values using a hierarchical pyramid approach [1]

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:

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

$$R_0 \in R^{M \times N}$$

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

A ratio $r = \frac{N_{\text{missing}}}{N_{\text{total}}}$ is computed for each scale, where N_{missing} is the number of missing values and N_{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:

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

$$R_k \in R^{M \times N}$$

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

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

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

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.

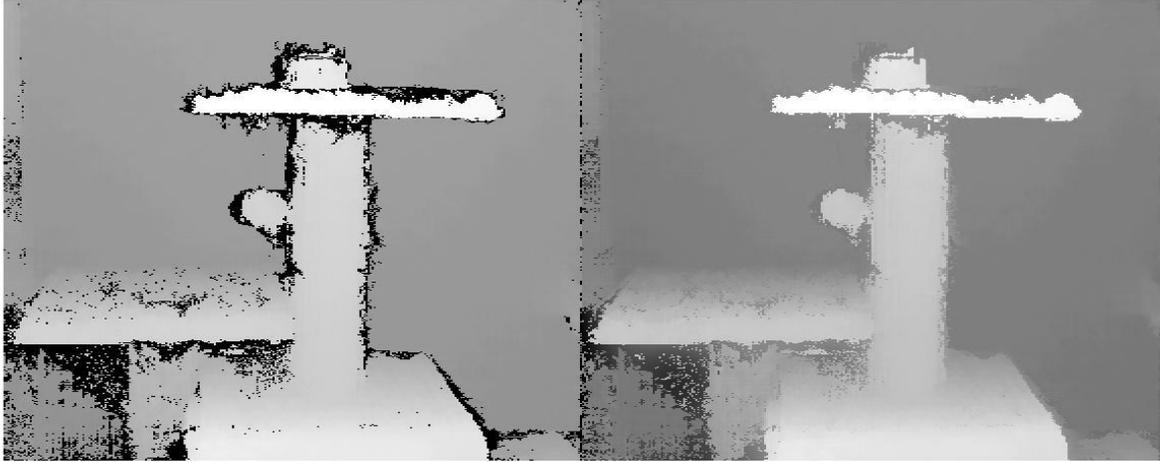


Figure 3: Difference between the raw disparity map from the SGM method and the filled out map using the Gaussian pyramid approach

2.2 Demonstration: Integration of unsupervised parameter-free segmentation with acquired image

In the previous quarter, an unsupervised parameter-free segmentation technique was implemented, and was evaluated on the Middlebury [2] dataset. In this section, we apply the same method to the images extracted from the stereo camera. The image acquired from the stereo-correspondence algorithm is by default aligned to the left camera. The disparity range parameter entered works in such a way that the correspondences for the first few columns (equal to the disparity range maximum) are blank and designated as NaN. We remove those columns from our computation and use the remainder of the image for segmentation. Figure 4 shows the fused representation of the disparity map acquired through semi global matching and the color image from the left camera. This shows that the disparity map is well aligned spatially to the left camera, as evidenced by the coinciding positions of the corners of the table, and the covering of the sphere hanging from the thread. However, there is a bleeding of object boundaries and noise from shadows in the disparity map.

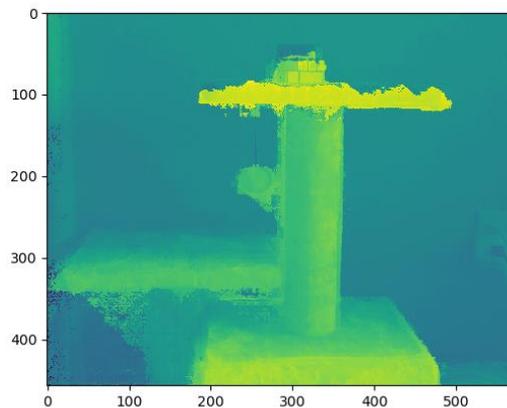


Figure 4: Fused representation – Disparity Map (overlay) with the baseline RGB image from the left camera

Figure 5 shows the results of applying the algorithm proposed in the previous quarter for various values of

disparity map and RGB weighting ($\lambda = \{1,5,10\}$). The key observation was that weighting the RGB or disparity values did not affect the general qualitative appearance or the number of clusters produced by the algorithm significantly between the two options, but within each weighting factor for the disparity map or the RGB (ie varying values of λ), a change in λ changes the segmentation results in an unpredictable manner. This is because the algorithm works in a parameter-free, unsupervised fashion where the objective is to perform a bottom-up hierarchical agglomeration of clusters with first neighborhood distance relations, which is qualitatively like the principle behind k-nearest neighbors. Clearly, the shadow noise in the disparity map affects some of the results. The object position is segmented out, but is attached to the rest of the foreground, such as the table, which makes it a drawback of the algorithm to clearly extract each object in a complex scene. In the next quarter, we will collect data from the pipeline scene and apply it to the algorithm to observe its performance on simpler defect detection problems and demonstrate the point of qualitative scene complexity when the algorithm fails.

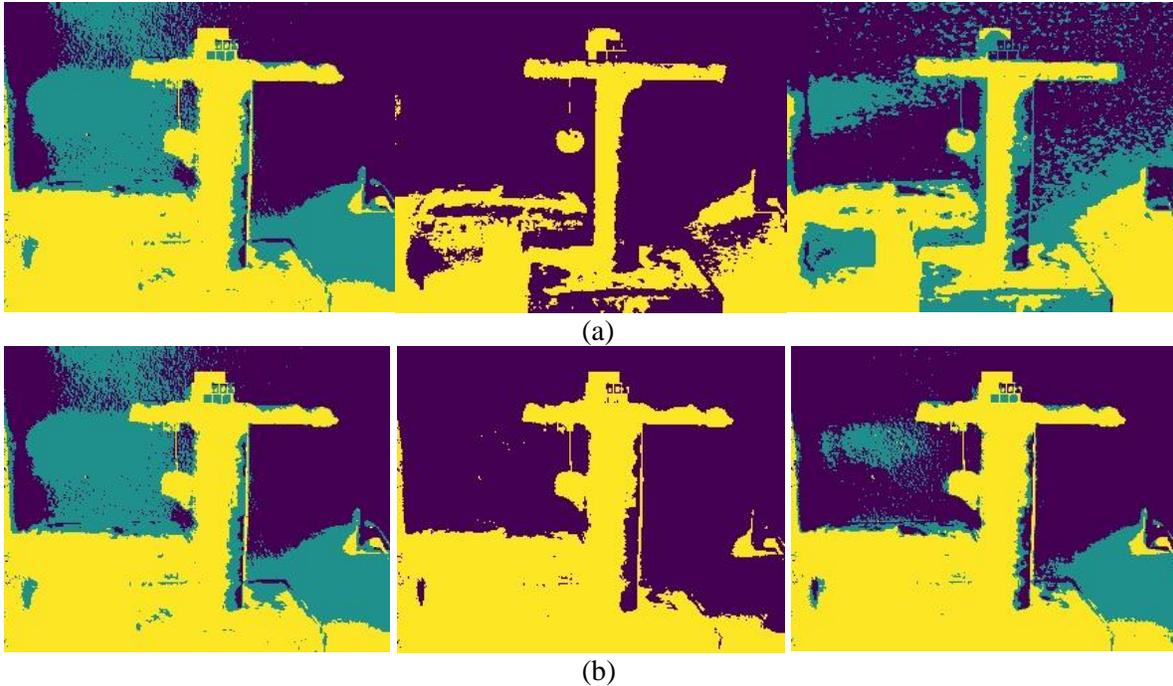


Figure 5: Results obtained from the segmentation algorithm from quarter 1 by: (a) varying λ for RGB image ($\lambda = \{1,5,10\}$) (b) varying λ for disparity map ($\lambda = \{1,5,10\}$)

Distance Estimation from Segmentation:

The next task was to estimate the edge length of the cube using the segmentation result. Ideally, the segmentation algorithm should produce an output that isolates the cube from the rest of the image. But since the previous section demonstrates that the cube is not separated from the scene, we manually pick points on the cube and match them to corresponding points in the right camera for a triangulation estimation of the edge length, as shown in figure 6. The pixel coordinates on the left camera are [282 39] and [329 39]. The corresponding coordinates on the right camera are [239 40] and [287 41] respectively. Upon applying the distance estimation code from the previous quarter, we find that the length of the top edge of the cube for the selected points is 6.32cm, and the original length of the top edge is 5.62cm, giving us an error of 12.45%.

Using the segmentation results and the triangulation method has a number of error sources that can affect the distance computation. To demonstrate this, Figure 7(a) shows a mask of just the cube extracted using MATLAB's local graph cut method in the Image Segmentation Toolbox. We use the mask to extract Harris corner features, and we apply the detected corner points directly into the left camera image as shown in Figure 7 (a). We also find corner points using the Harris detector in the right camera image. The corner

points closest to the cube edge are selected and the distance estimation is performed again using triangulation. Here we end up getting an edge length of 11.43cm, which overshoots the true edge length with an error of 104%. The points that are chosen for the left camera are [285 42] and [327 41], and the points that are chosen for the right camera are [241 43] and [287 44]. Shifting the first pixel coordinate of the left camera from [285 42] to [282 42] and leaving all the other coordinates intact gives us an edge length of 6.54cm. While this result is better than the previous one, it is obvious that the triangulation method is very sensitive to the pixel coordinates. The reprojection error for the first set of points is 1.73 pixels, and the reprojection error for the second set of points is 0.79 pixels. A visualization of the uncertainty in the distance estimation is presented in figure 7 (b). This was produced by calculating the edge length 100 times using a random integer generator from a uniform distribution as shown below:

P1 (left camera): $[U \sim [275 \ 285], U \sim [35 \ 45]]$
P1 (right camera): $[U \sim [235 \ 245], U \sim [35 \ 45]]$
P2 (left camera): $[U \sim [325 \ 335], U \sim [35 \ 45]]$
P2 (right camera): $[U \sim [285 \ 295], U \sim [35 \ 45]]$

The mean edge length obtained was 18.53 cm, and the standard deviation is 10.73 cm. A more robust algorithm is needed to account for segmentation boundary errors, and the errors from triangulations for distance estimation and will be worked upon in the future.

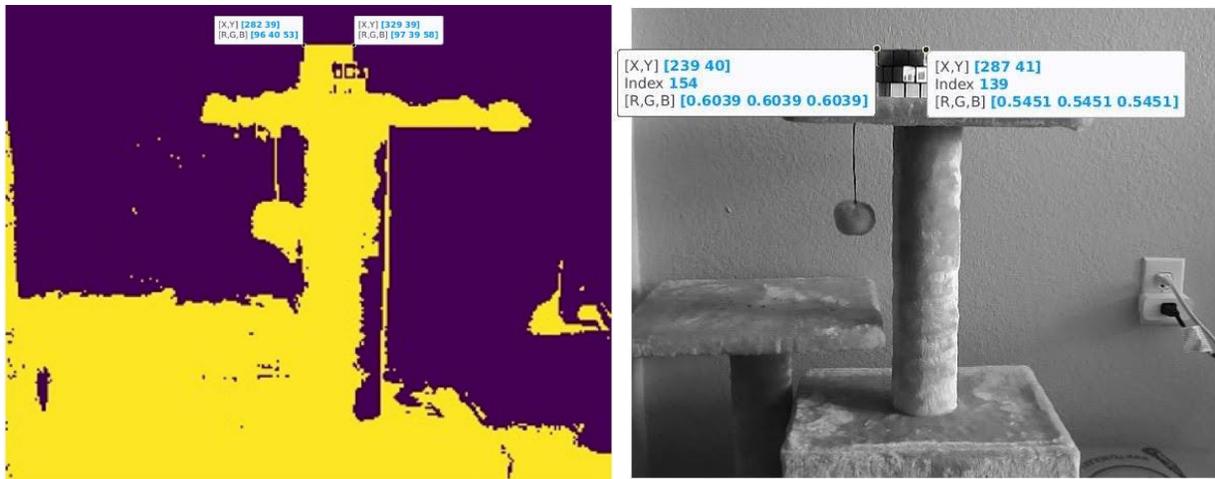
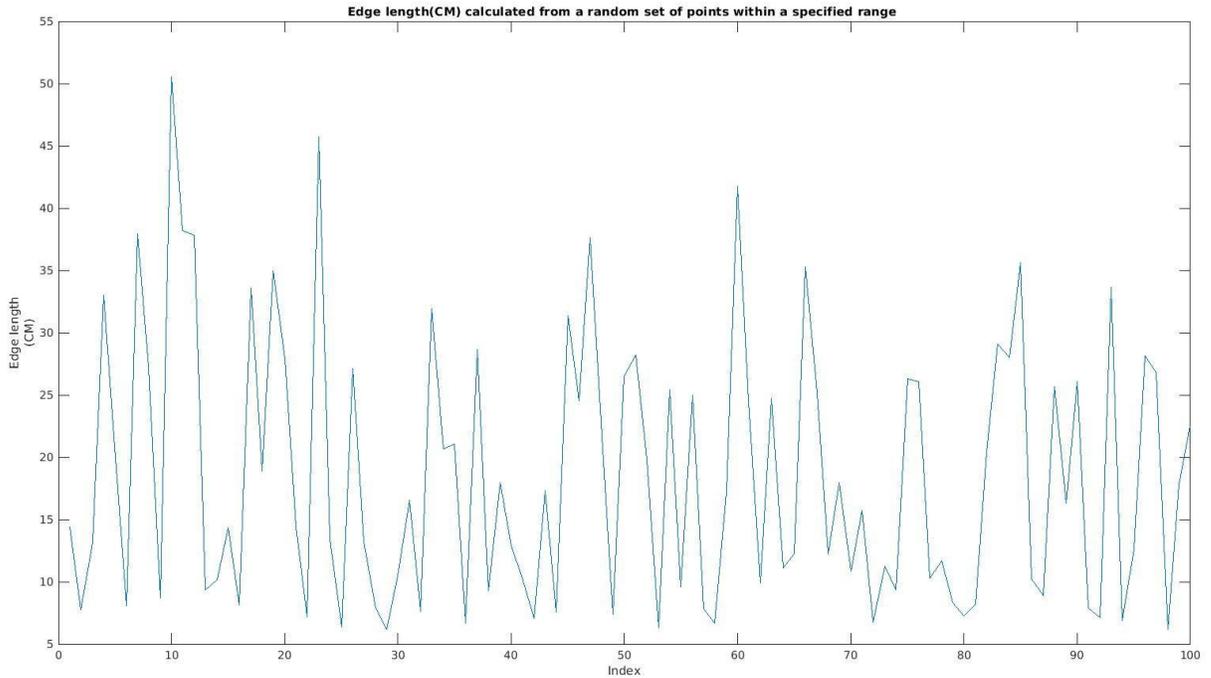


Figure 6: Segmentation chosen for the edge length calculation with the coordinates marked manually



(a)



(b)

Figure 7: (a) (Left to right) Extracted mask for the cube, Harris Corner detection on the mask transferred to the left camera image, Harris corner detection performed on the right camera image. The left and right camera images have multiple corners, and the edge corners of the top edge were chosen manually for this demonstration. (b) Uncertainty in the edge length for a random set of points within a specified range, near the actual top corners of the cube.

3. Task 2: Algorithm development for AI-based damage detection

3.1 Fully-Convolutional Network Architecture

In this section, a fully-supervised machine learning framework for semantic scene segmentation is described. Convolutional Neural Networks (CNN) yield a hierarchy of learnable features that are obtained via training on a dataset. 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.

Convolutional networks have been used for object detection and image segmentation, where patch-based, region-of-interest based methods contain multiple steps such as region proposal search, as in the R-CNN, Mask R-CNN, and Fast R-CNN techniques, or it contains a fixed set of regions to output directly a set of bounding boxes in an end-to-end fashion with requirements on anchor boxes, as in YOLO v3.

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 due to the fact that 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 had the VGG-16 backbone network. The VGG-16 network

had 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 8. 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.

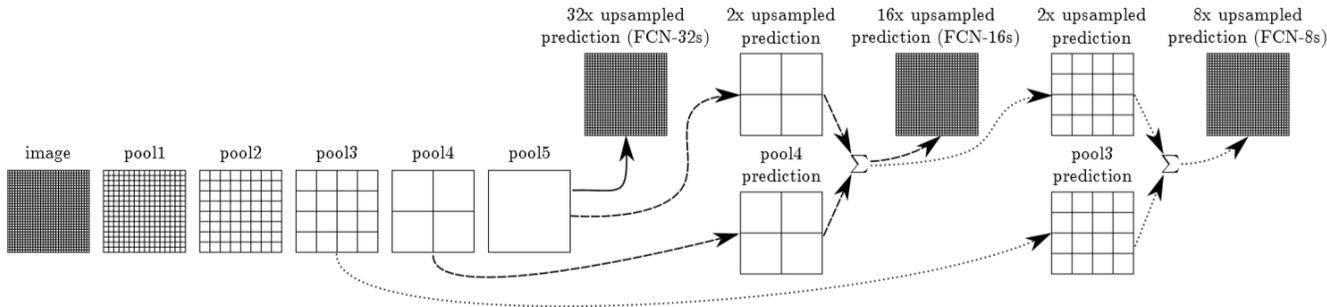


Figure 8: Skip connections in fully convolutional networks [3]

3.2 Dataset

The dataset used for all the experiments is the NYU RGB-D dataset. The dataset consists of a triplet of a raw color image, depth map, and a ground truth label. The images were acquired from a Microsoft Kinect depth camera, and consists of 1449 RGBD pairs, with an associated label image and 40 distinct categories. This dataset will have a higher scene complexity as compared to the pipeline scenes and is currently being used to develop a general scene segmentation algorithm. It is asserted that a data-driven algorithm that can generalize well to a complex scene can do better on a simpler dataset.

The dataset was split, with 1093 images and 356 images for training and validation, respectively. Cross-validation was not performed for these experiments and will be carried out in the future.

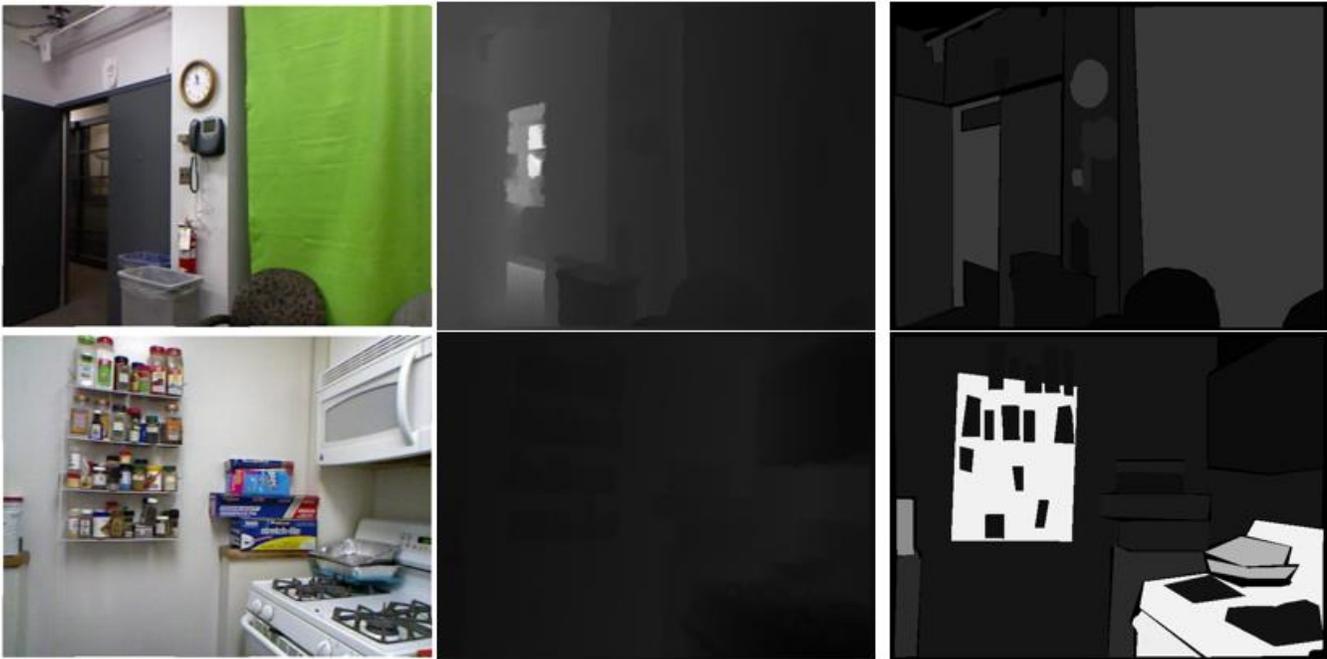


Figure 9: Demonstrative examples of the NYU v2 RGB-D dataset (Left: RGB Image Middle: Depth Map Right: Labels). The depth map and labels shown here use the 8-bit representation with the real values of the matrix without any colormap transforms.

3.3 Fusion Blocks

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.

This section describes fusion in the context of (ii), where we use intermediate features obtained using a FCN. The general architecture is shown in figure 10.

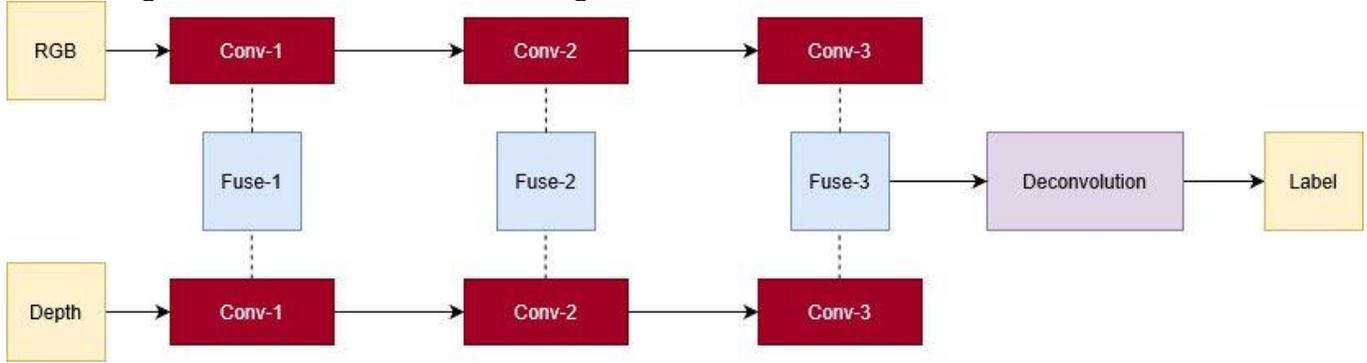


Figure 10: General architecture for a fully convolutional fusion network

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:

$$F = M_{RGB} + M_D$$

This is simple element wise addition of features from the convolutional layers. These outputs are then combined with the corresponding deconvolutional layer output. The schematics are shown in figure 11.

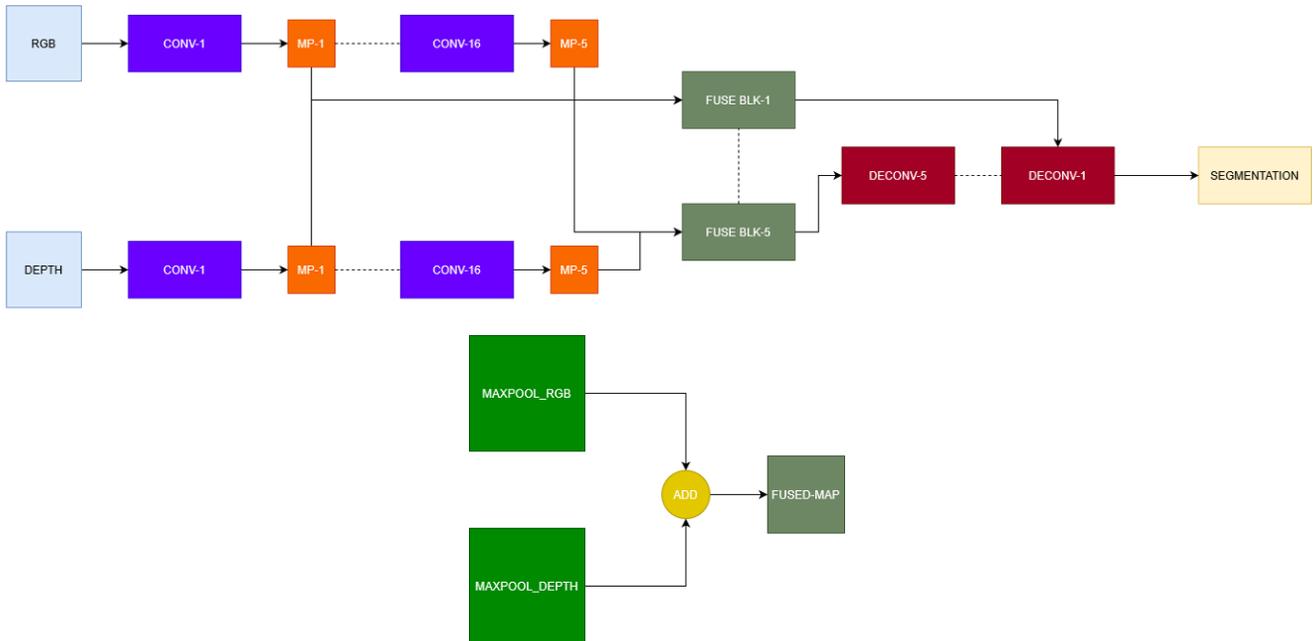


Figure 11: (Top) Fusion architecture-1 for FCN (Bottom) Fusion block design for architecture-1

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:

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

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

Where K is a 1x1 convolutional filter set.

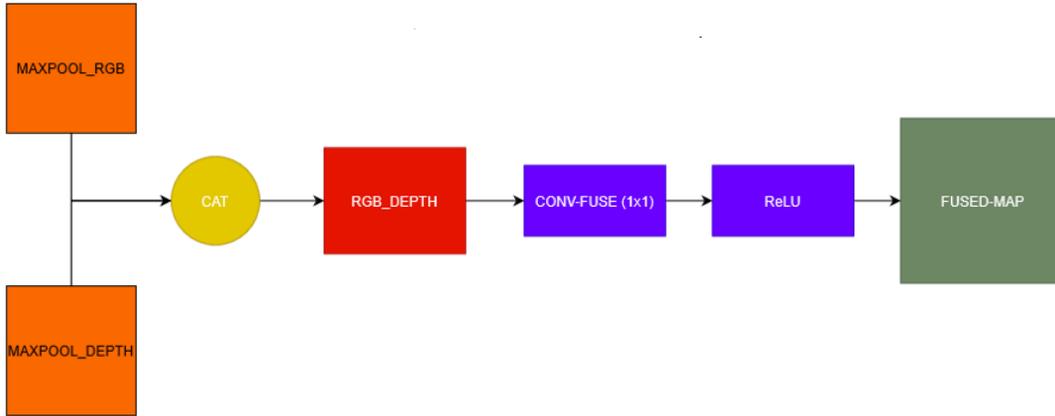


Figure 12: (Top) Fusion architecture-2 for FCN (Bottom) Fusion block design for architecture-2

The above formulation is represented in figure 12. 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.

3.4 Imbalanced data – Compensatory Loss functions

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. This problem is exacerbated in multi-class settings and scenes with clutter and small objects. 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. The data imbalance problem can be dealt with by a combination of the sampling strategy for the data and the loss function itself. The loss function models the expected value between the predicted output and the real output, and in the case of supervised learning, is quite straight forward to define for the segmentation task. The sampling strategies could involve systematic oversampling of the minority class or under sampling of the majority class. Effectively, this ends up reducing the imbalance in the dataset by introducing ‘fake’ samples of the minority class or removing extra examples in the majority class. Different loss functions have different objective and changing the loss function according to the task at hand was found to be helpful for the segmentation task. Loss functions can be adapted to be sensitive to learn the minority classes better by having a higher cost for a bad prediction of the minority class as compared to the majority class. Punishing the algorithm for making an error for a minority class would lead to better weights in the network to detect these classes, but it would be at the expense of the majority class. This is relevant for the pipeline dataset because we want our algorithm to be more sensitive to errors in the detection of damage as compared to the background.

The NYU dataset presents itself as a highly imbalanced, long tailed dataset with 40 categories, as shown in figure 13. The background class has 233% more pixels than the next dominant category (Class #21). Minority classes are more than 6 orders of magnitude less frequent, making detection a particularly sensitive task to the high end of the distribution without any modifications to the regular loss function and data sampling approaches. We briefly discuss two approaches taken to the problem before moving on to the results.

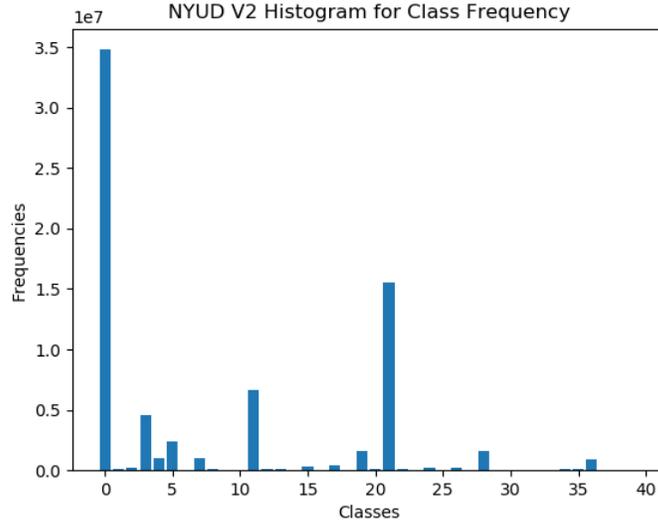


Figure 13: Class Histogram for the complete NYU v2 RGB-D dataset

Class balanced sigmoid cross entropy using effective volume of samples:

The first approach to address long tailed data distribution involves a theoretical framework that measures data overlap and performs reweighting of the classes based on their frequency using the effective number of samples in that class [4]. Originally made for classification, we adapt the framework for semantic segmentation. A classical approach to reweight the data would be to have an inverse relationship between the frequency of the data and the corresponding weighting term in the loss function. The argument presented is for the use of effective number of samples is inspired by the random covering problem, where the goal is to cover a large set by a sequence of i.i.d. random small sets. Let K be a fixed set and let $B = \{B_1, B_2 \dots\}$ be a sequence of iid random smaller sets. Let N be the number of small sets to cover K completely. The greatest lower bound of the collection of small sets such that B contains K is defined as:

$$N = \inf \{n: \cup_1^n B_i \supset K\}$$

The argument for using an effective number of samples is centered around the assertion that more samples from a class does not necessarily imply more useful information, as there can be considerable overlap between the samples. We can assume that a newly sampled datapoint is either entirely inside the set of previously sampled data or it is not, with probabilities p and $1 - p$ respectively. The value of p increases as the number of datapoints increases. Due to the intrinsic similarities among real world data, as the number of samples grows, it is possible that a new sample is a near duplicate of one of the previous samples. Data augmentations are essentially near-duplications of the same data point with minor variations. The more augmented a dataset is, the smaller the value of N to cover the entire class. Let us denote the expected volume of the number of samples as E_n , where n is the number of samples:

$$E_n = \frac{1 - \beta^n}{1 - \beta} = \sum_{j=1}^N \beta^{j-1}$$

$$\beta = \frac{N - 1}{N}$$

The j^{th} sampled contributes β^{j-1} to the effective number of samples. In the asymptotic case, $E_n = 1$ if $\beta = 0$ and $E_n \rightarrow n$ as $\beta \rightarrow 1$. This means that when N is large, the effective number of samples is the same as the number of samples n . In this scenario, we think the number of unique prototypes N is large, thus there is no data overlap and every sample is unique. On the other hand, if $N = 1$, this means that the effective contribution of the class is essentially just one sample. The case for the background class is usually one where there is a large overlap, and minority classes approach an effective number equal to the original number of samples. This formulation is a generalization of the inverse class frequency approach, as $\beta = 0$ corresponds to no weighting, and $\beta = 1$ corresponds to the use of inverse class frequency.

The above formulation can be added to the loss function an inversely proportional scaling factor to the effective number of samples. Given a sample from class c_i that contains n_i samples in total, we can scale the loss corresponding to the class as:

$$L_{CB} = \frac{1}{E_{n_y}} L(p, y)$$

p – detected class y – ground truth label

Applying the class balancing term to the sigmoid cross entropy loss:

$$CE_{sigmoid}(z, y) = - \sum_{i=1}^c \log \left(\frac{1}{1 + \exp(-z_i^t)} \right)$$

$$CE_{CB}(z, y) = - \frac{1 - \beta}{1 - \beta^{n_y}} \sum_{i=1}^c \log \left(\frac{1}{1 + \exp(-z_i^t)} \right)$$

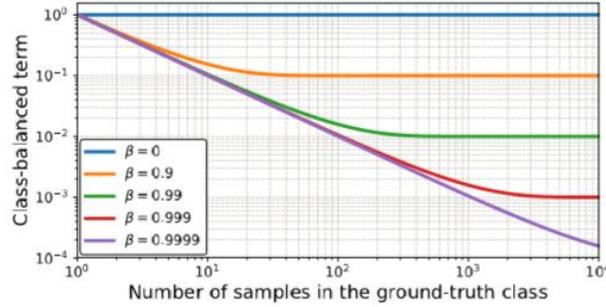


Figure 14: Variation of the class-balancing term, the inverse of the expected volume, as the number of samples in the ground truth class increases, in logarithmic scale.

Piecewise linear convex surrogate for the IoU loss using Lovasz extension of submodular set functions:

The performance metrics used for evaluating the segmentation models is the accuracy and the IoU score (Jaccard score). The Jaccard score is a better indicator of segmentation quality because it incorporates information about misclassified pixels (false positives and false negatives) into the formulation. Optimizing the network for the Jaccard score directly is the subject of this section, Given a vector of ground truth labels y^* and a vector of predictions y the Jaccard score of class c is:

$$J_c(y^*, y) = \frac{|\{y^* = c\} \cap \{y = c\}|}{|\{y^* = c\} \cup \{y = c\}|}$$

Corresponding to the above expression, which lies in $[0,1]$ with 1 being the best metric. For the loss, therefore we can write $\Delta J_c(y^*, y) = 1 - J_c(y^*, y)$

In order to optimize the IoU score, a smooth extension to this discrete loss function is proposed. This can be accomplished by interpolating the values of the Jaccard index to yield a convex surrogate loss. We approximate the loss function of a prediction function $f: X \rightarrow Y$ by an empirical sum of losses over a finite sample. The approximation of a discrete loss function is done using a convex surrogate to the discrete loss function, that estimates an upper bound of the loss. For a segmentation output y and ground truth y^* we define the set of mispredicted pixels for class c as:

$$M_c(y^*, y) = \{y^* = c, y \neq c\} \cup \{y^* \neq c, y = c\}$$

For a fixed ground truth y^* the loss function can be written as a function of M_c :

$$\Delta J_c: M_c \in \{0,1\}^p \mapsto \frac{|M_c|}{(|\{y^* = c\} \cup M_c|)}$$

We need to compute the convex closure of the above equation in R^p for a continuous version of the loss. The above equation is shown to be submodular in [5]. The convex closure of the above equation is obtained as the Lovasz extension.

Definition: The Lovasz extension of a set function $\Delta: \{0,1\}^p \rightarrow R$ such that $\Delta(0) = 0$ is defined by:

$$\bar{\Delta} : m \in R^p \mapsto \sum_{i=1}^p m_i g_i(m)$$

$$\text{with } g_i(m) = \Delta(\{\pi_1, \dots, \pi_i\}) - \Delta(\{\pi_1, \dots, \pi_{i-1}\})$$

π – permutation ordering the components of m in decreasing order

Let Δ be a set function encoding a submodular loss such as the Jaccard loss. By submodularity, $\bar{\Delta}$ is the tight convex closure of Δ . $\bar{\Delta}$ is piecewise linear and interpolates the values of Δ in R^p not included in the original discrete points and has the same values as Δ on $\{0,1\}^p$. If m is a vector of all pixel errors, then the convex closure $\bar{\Delta}$ is a sum weighting these errors according to the interpolated discrete loss.

For the multi-class segmentation problem, a surrogate based on the softmax loss is chosen. We map the output scores of the model to a softmax layer just as in softmax cross-entropy. We use the resulting class probabilities $f_i(c) \in [0,1]$ to construct a vector of pixel errors $m(c)$ for class $c \in C$:

$$m_i(c) = \begin{cases} 1 - f_i(c), & \text{if } c = y_i^* \\ f_i(c), & \text{otherwise} \end{cases}$$

$$m(c) \in [0,1]^p$$

We then construct the surrogate loss using $m(c)$:

$$l(f(c)) = \bar{\Delta} J_c(m(c))$$

The algorithm to compute the function g given m is shown in algorithm 1 and runs in $O(p \log p)$. The class averaged loss is then given by:

$$L(f) = \frac{1}{|C|} \sum_c \bar{\Delta} J_c(m(c))$$

3.5 Results and Discussion

Multiple experiments were conducted for semantic segmentation using the baseline network VGG-16, and fusion block design, data sampling and loss functions were varied. The depth map is expected to add new information to the network and improve performance metrics, subject to the fusion criterion. The objective is to demonstrate a measurable improvement to the performance metrics, accuracy and IoU. Parameters other than the fusion technique, and loss function were kept fixed. For the RGB-only case, we increased the batch size to 32, because the memory usage from depth map was freed up. Table 1 shows a summary of training parameters in the demonstration.

Table 1: Summary of the parameters used in the key experiments done for the study

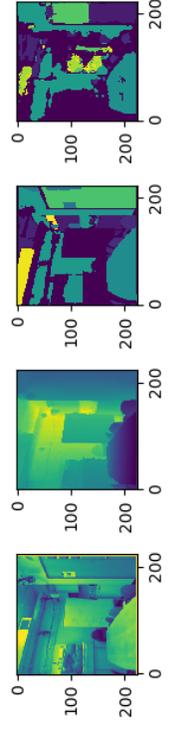
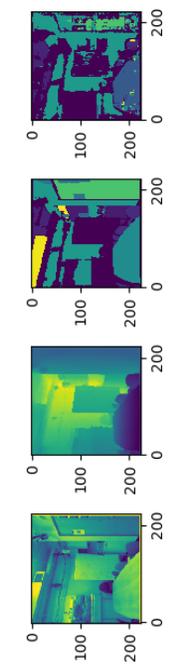
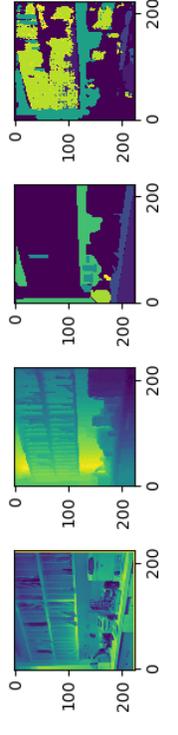
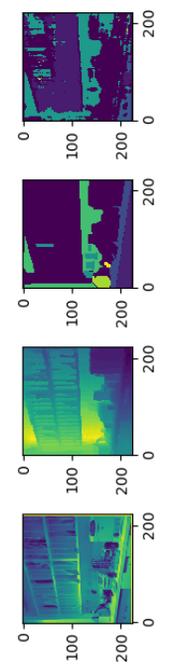
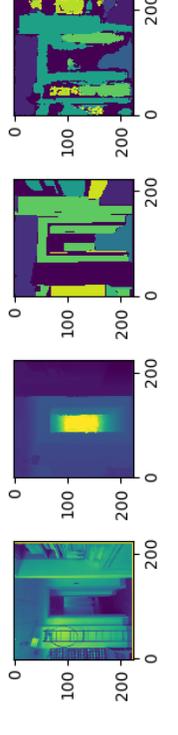
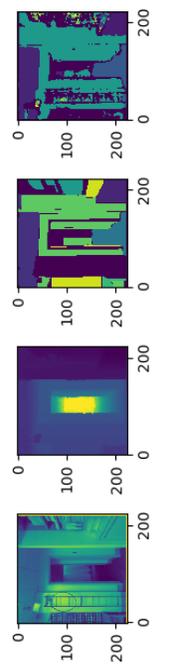
Network	Fusion	Batch size	Epochs	Learn rate	Step size	Loss	Input/Output Size	Gamma	β_{CB}
VGG-16	None	32	100	0.0001	20	Cross Entropy	(224x224)	0.5	N/A
VGG-16	Element-Wise addition	16	100	0.0001	20	CB-Sigmoid Cross Entropy	(224x224)	0.5	0.9999
VGG-16	Non-linear Fusion	16	100	0.0001	20	CB-Sigmoid Cross Entropy	(224x224)	0.5	0.9999
VGG-16	Element-wise addition	16	100	0.0001	20	Lovasz Softmax	(224x224)	0.5	N/A
VGG-16	Non-linear Fusion	16	100	0.0001	20	Lovasz Softmax	(224x224)	0.5	N/A

Table 2: Results from the experiments with both the training and validation set metrics

Description	Loss minimum (Train/Val) (approx.)	Peak Accuracy (Train/Val)	Peak IoU (Train/Val)
RGB Only – Cross Entropy	0.14	0.62	0.47
	0.15	0.68	0.38
RGBD – Elementwise Add (CB-Sigmoid Loss)	0.04	0.65	0.39
	0.05	0.68	0.44
RGBD – NL Fusion (CB-Sigmoid Loss)	0.06	0.67	0.54
	0.05	0.65	0.42
RGBD – Elementwise Add (Lovasz Loss)	0.34	0.86	0.52
	0.48	0.78	0.39
RGBD – NL Fusion (Lovasz Loss)	0.3	0.79	0.53
	0.49	0.85	0.4

Table 2 shows a summary of the results obtained from our experiments. As expected, the accuracy of the network showed a general increasing trend as the depth information was added. Making the switch from a simple element wise addition to a non-linear fusion did not increase accuracy in the case of the sigmoid class-balanced loss, and increased accuracy by 6% when the Lovasz loss was used. However, the effect of changing the loss function formulation showed that the Lovasz loss performed worse than the Class balanced loss in terms of the validation Jaccard score. This was not an expected result, because the Lovasz loss optimized for the IoU directly. More investigations on the computation of the Lovasz loss will be done in the next quarter to explain this result.

Figure 15 shows three demonstrative examples of the detections in the validation set. Evidently, the class balanced sigmoid loss tends to capture the presence of small objects better than both the Lovasz loss and the cross-entropy loss. The detections with the simple element wise addition model and the class balanced loss were the least noisy in general and produced the highest IoU score in the validation set. Extensive experimentation with various parameters, ablation studies and cross validation would provide us a clearer picture of the observations shown here. As part of the future work, we are planning to incorporate an automated, comprehensive, systematic evaluation procedure for semantic segmentation models to better inform the significance of the results and the uncertainties in the performance metric measures. It is crucial to appreciate the latter factor because the training of the neural network consists of tuning multiple hyperparameters, the use of Stochastic Optimization, and the random initialization of weights in the deconvolution filters. The VGG-16 network is relatively small compared to networks such as ResNet-50, and the effect of increasing network complexity on information fusion will be interesting to observe. The VGG-16 model serves as a good baseline for our initial studies as the complexity of the network does not tax the GPU available on the local system, and the fusion of max-pooling outputs leads to a variety of interesting fusion design choices that can be experimented upon with this baseline model.

RGBD – NL Fusion (CB-Sigmoid Loss)	RGBD – Elementwise Add (CB-Sigmoid Loss)	Description
 <p>Four depth maps for Example 1 using NL Fusion. The maps show depth values on a 200x200 grid. The first map shows a scene with a table and chairs. The second map shows a scene with a table and chairs. The third map shows a scene with a table and chairs. The fourth map shows a scene with a table and chairs.</p>	 <p>Four depth maps for Example 1 using Elementwise Add. The maps show depth values on a 200x200 grid. The first map shows a scene with a table and chairs. The second map shows a scene with a table and chairs. The third map shows a scene with a table and chairs. The fourth map shows a scene with a table and chairs.</p>	<p>Example 1</p>
 <p>Four depth maps for Example 2 using NL Fusion. The maps show depth values on a 200x200 grid. The first map shows a scene with a table and chairs. The second map shows a scene with a table and chairs. The third map shows a scene with a table and chairs. The fourth map shows a scene with a table and chairs.</p>	 <p>Four depth maps for Example 2 using Elementwise Add. The maps show depth values on a 200x200 grid. The first map shows a scene with a table and chairs. The second map shows a scene with a table and chairs. The third map shows a scene with a table and chairs. The fourth map shows a scene with a table and chairs.</p>	<p>Example 2</p>
 <p>Four depth maps for Example 3 using NL Fusion. The maps show depth values on a 200x200 grid. The first map shows a scene with a table and chairs. The second map shows a scene with a table and chairs. The third map shows a scene with a table and chairs. The fourth map shows a scene with a table and chairs.</p>	 <p>Four depth maps for Example 3 using Elementwise Add. The maps show depth values on a 200x200 grid. The first map shows a scene with a table and chairs. The second map shows a scene with a table and chairs. The third map shows a scene with a table and chairs. The fourth map shows a scene with a table and chairs.</p>	<p>Example 3</p>

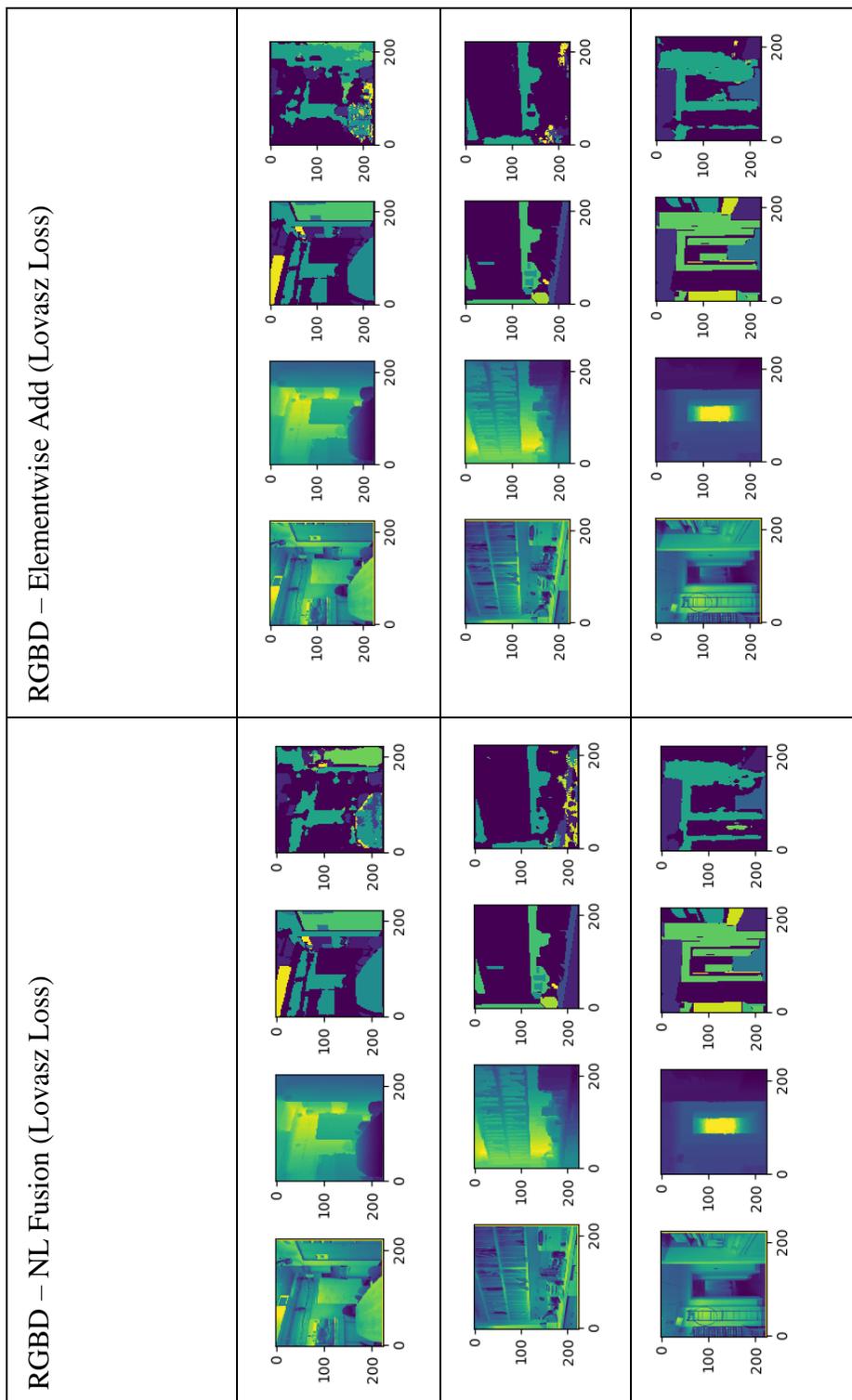


Figure 15: Demonstrative examples of the detections in the validation set for varying fusion block designs and class reweighting schemes

4. Summary and Future Work

4.1 Task 1

In this quarter, we improved upon the disparity map by using a hierarchical hole filling algorithm and integrated this result with the segmentation method from quarter 1. In the next quarter, we will conduct experiment in pipelines to simulate realistic environment for pipeline inspection. Furthermore, we will continue to work on hardware design and integration while investigating hardware requirements particularly on lighting quality and camera resolution to ensure accurate detection and characterization of pipeline

defects. Lab experiments in pipeline environment will also be conducted to verify the developed systems and algorithms in more realistic settings.

4.2 Task 2

In this quarter, we developed a fully-supervised fusion algorithm using deep learning to perform semantic segmentation in the NYU v2 RGB-D dataset. We experimented on two fusion block architectures and two different loss functions to study the effect of changing the fusion method and to counteract class imbalance, respectively. Future progress would be focused on improving segmentation capabilities by proposing improved fusion blocks and loss functions and verifying the proposed method using pipeline image dataset.

References

- [1] M. Solh and G. Alregib, "Hierarchical hole-filling for depth-based view synthesis in FTV and 3D video," *IEEE Journal on Selected Topics in Signal Processing*, 2012, doi: 10.1109/JSTSP.2012.2204723.
- [2] D. Scharstein and C. Pal, "Learning conditional random fields for stereo," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007, doi: 10.1109/CVPR.2007.383191.
- [3] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015, doi: 10.1109/CVPR.2015.7298965.
- [4] Y. Cui, M. Jia, T. Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019, doi: 10.1109/CVPR.2019.00949.
- [5] J. Yu and M. B. Blaschko, "Learning submodular losses with the Lovász hinge," in *32nd International Conference on Machine Learning, ICML 2015*, 2015.