Improve the Grasping Performance by Analyzing Target Objects with Computer Vision and Deep Learning Algorithm

by
Junhyeok Jeong

A THESIS

submitted to

Oregon State University

Honors College

in partial fulfillment of
the requirements for the
degree of

Honors Baccalaureate of Science in Computer Science
(Honors Associate)

Presented May 13, 2021
Commencement June 2021

AN ABSTRACT OF THE THESIS OF

Junhyeok Jeong for the degree of Honors Baccalaureate of Science in Computer Science presented on May 13, 2021. Title:
Improve the Grasping Performance by Analyzing Target Objects with Computer Vision and Deep Learning Algorithm

Abstract

approved:_____

Cindy Grimm

Recently there has been a large amount of research into basic human movement as a baseline for robotic motion. Robotic grasping is a challenging problem for a number of reasons. One of which is that it is difficult to accurately know the interaction between the hand and the object, especially for tasks where the arm is moving and as a result a fixed camera is not convenient. To alleviate the difficulty, there are lots of studies that focus on planning ideal pre-grasp pose by mimicking human actions on robots [1].

This thesis developed and validated computer vision and deep learning framework to provide object detection, object size, and location based on the camera and ultrasonic sensor. In this thesis, there are two main methodology. First, the deep learning framework, are called 'YOLOv3'. It allows us to detect objects with the real-world image training dataset. Second, this thesis presents the utilization of OpenCV for computer vision because it allows measuring the size of detected objects and distance from the camera to the object.

After tests and trials, we found some problems like the increment error range by camera calibration and the optimization of the fixed camera position. However, our methods can accurately determine the location of the object in XYZ space relative to the hand accurate within about 7% of error rate. Future work can extend towards using other deep learning frameworks and change segmentation methods with other vision sensors like depth camera and LIDAR. Although the transition method could require high computational cost, we expect it will improve the accuracy of object detection with more objects.

Key Words: Computer Vision, Deep Learning, Robotic Grasping

Corresponding e-mail address: jeongju@oregonstate.edu

Improve the Grasping Performance by Analyzing Target Objects with Computer Vision and Deep Learning Algorithm

by
Junhyeok Jeong

A THESIS

submitted to

Oregon State University

Honors College

in partial fulfillment of
the requirements for the
degree of

Honors Baccalaureate of Science in Computer Science
(Honors Associate)

Presented on May 13, 2021
Commencement June 2021

Honors Baccalaureate of Science in Computer Science project of Junhyeok Jeong presented on May 13, 2021.

APPROVED:

_____

Cindy Grimm, Mentor, representing the School of Mechanical, Industrial, and Manufacturing Engineering

_____

Minsuk Kahng, Committee Member, representing the School of Electrical Engineering and Computer Science

_____

Nigel Swenson, Committee Member, representing the School of Mechanical, Industrial, and Manufacturing Engineering

_____

Toni Doolen, Dean, Oregon State University Honors College

I understand that my project will become part of the permanent collection of Oregon State University, Honors College. My signature below authorizes release of my project to any reader upon request.

_____

Junhyeok Jeong, Author

Acknowledgements

This thesis has been an amazing learning opportunity during my undergraduate computer science degree at Oregon State University. I would like to appreciate everyone who has helped and supported me through this thesis project. First, I would like to appreciate Dr. Cindy Grimm, who introduced me the grasping research team and great robotics lab at the Graf Hall. She always gave me great advices and tools for my thesis project. I would especially like to appreciate Nigel Graham Swenson, Nuha Nishat, Yi Herng Ong, and Kartik Gupta because I have learnt a lot from all of you and really appreciate to help and give advices for my thesis project. I really appreciate Dr. Cindy Grimm and all students in the grasping lab again!

Contents

Acknowledgements 8

1 Introduction 10

2 Background 11

3 Object Detection 14

3.1 YOLO 15
3.2 Settings and Processes 17
3.2.1 Input Dataset and Camera Sensor 17
3.2.2 Model Configuration 18
3.3 Result 19

4 Computer Vision with OpenCV 22

4.1 Object Contours and Measurement 22
4.2 Settings 24
4.2.1 Measurement Sensor 24
4.2.2 FOV and Camera Matrix 24
4.3 Result 26

5 Overall Discussion 28

6 Future Work 29

7 Conclusion 30

References 32

1 Introduction

In recent years, the applications of robotics and artificial intelligence technologies have been increasing around the world because they both have great potential in numerous applications, and can both greatly improve the quality of human life. Robots are now being assigned to assist humans in a much broader range of tasks, such as medical procedures, elderly care, agriculture work, disaster response, and remediation of the environment [2]. Grasping has a huge role for all applications because lots of tasks require robots to grasp an object. Therefore, by using computer vision and deep learning technology, this thesis would like to answer the follow research questions:

1. Can the custom object detection model detect the desired object with a webcam and the object measurement works on various distances in the real time?
2. Can the object detection model connect with the object measurement module in python?
3. Can the object detection and object measurement abilities improve the performance of the grasping robot?

To accomplish our goal of deep learning framework, this thesis uses the most famous deep learning framework for object detection and segmentation, which is called ‘YOLOv3’. This framework was trained with various 3D printed basics shape images as an input dataset. For the object measurement, this thesis uses OpenCV which is a widely used computer vision library. The main purpose of using OpenCV is acquiring the detected object measurement with various sensors and providing user interface on the application. With a camera sensor, OpenCV lets the user calculate the size of detected object with segmentation and pixel data. Furthermore, ultrasonic sensor is connected with Arduino Uno board. It uses to get the distance data from ultrasonic sensor to the detected object. Section 2 covers background on computer vision and deep learning framework.

Sections 3&4 discuss our methods to determine object location and the results of our experiments. The sections 5, 6, and 7 covers overall discussion after experiments, future works, and conclusion.

2 Background

At first, this section gives a brief background of the computer vision and deep learning as main cores of this thesis. After that, this section provides some background about OpenCV and its current uses. Lastly, this section dives into background for YOLOv3 as the most widely using deep learning framework for image recognition.

Computer vision is the science of understanding or manipulating images and videos. It as a field has a long history because it has lots of applications, including autonomous driving, industrial inspection, and augmented reality. Although it has long history, it confronted lots of technical limitations and challenges like low camera quality, computational speed, and high cost. With the emergence of deep learning and high quality of GPU & CPU, the technical limitations of computer vision is reduced and has been useful for various applications [3]. The use of deep learning for computer vision can be categorized into classification, detection, segmentation, and images/videos generation. Classification refers to a predictive modeling problem where a class label is predicted for a given example of input data like image. Detection is a computer vision technique that allows us to identify and locate objects in an image or video. Segmentation is the process of partitioning a digital image into multiple segments like sets of pixels, also known as image objects. Image/videos generation is basically a creation process based on existed dataset. For example, Generative adversarial network(GAN) model detects the patterns of input dataset and then creates new images based on the patterns. Since the main purpose of this thesis is object detection and measurement for grasping robot arm, we focus on detection and segmentation methods.

This thesis uses OpenCV as a computer vision tool because the use of computer vision is necessary to interact with visual sensors and get data from the real-world. OpenCV is an opensource library for image and video analysis, originally introduced more than a decade ago by Intel to achieve advanced vision research and disseminate vision knowledge [6]. OpenCV provides over 2500 optimized algorithms for computer vision and it also has a large community to get help and tutorials.

Deep learning is a collection of techniques from artificial neural network (ANN) as a branch of machine learning [3]. An artificial neural network is the piece of a computing system designed to simulate the way the human brain analyzes and processes information. It helps to solve problems that would prove difficult by human or statistical standards. With neural networks, deep learning models have multi-level layers as convolutional neural networks and are very helpful in extracting complicated information from input images. The deep learning workflow extracts relevant features automatically from images because deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction.

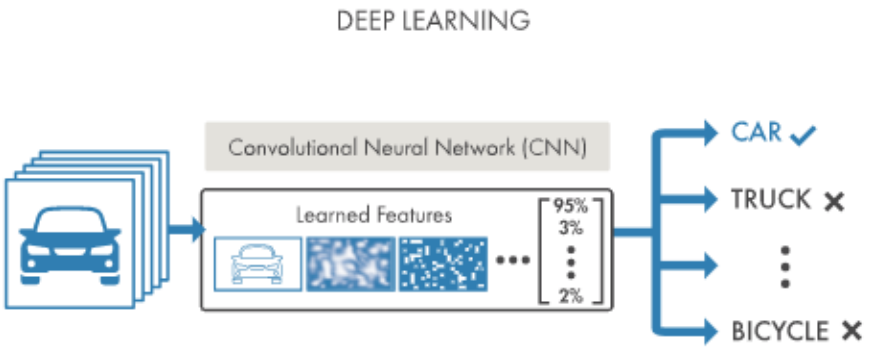


Figure 1. Simple Deep Learning Diagram [13]

Since modern computers are becoming faster and faster, the applications of deep learning are being feasible with good efficiency. With good computational speed and efficiency, various deep learning methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics [4].

When it comes to object detection for the grasping robot arm, this thesis chose YOLOv3 deep learning framework because this framework is easy to combine with OpenCV library and has fairly good performance. First, YOLO is a very commonly used model used for object detection because of its great accuracy and speed [7]. There are multiple versions of YOLO out there, it was decided to use YOLOv3, since it is the recent and stable version of the neural network.

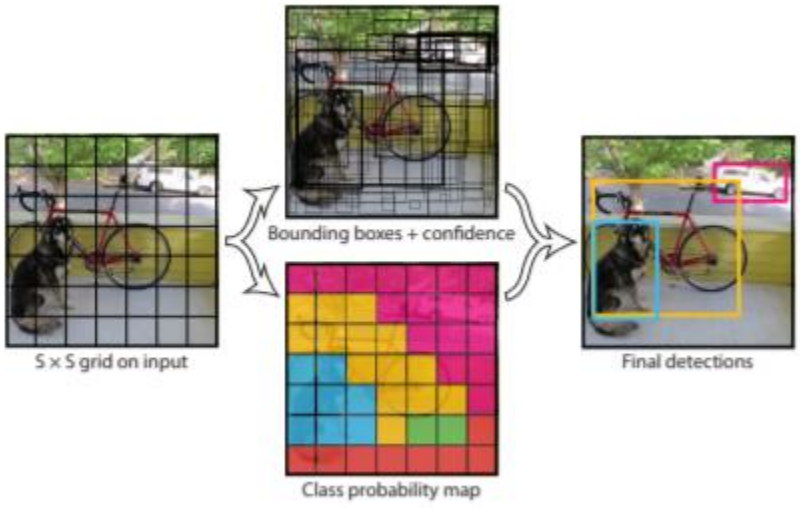


Figure 2. YOLOv3 basic idea [4]

YOLOv3 uses the K-means clustering method, which is an algorithm of unsupervised machine learning, to estimate the initial width and height of the predicted bounding boxes for specific object. With this method, the estimated width and height are sensitive to the initial cluster centers, and the processing of large-scale datasets is time-consuming. In order to address these problems, a new

cluster method for estimating the initial width and height of the predicted bounding boxes has been developed.

First, it randomly selects a couple of width and height values as one initial cluster center separate from the width and height of the ground truth boxes [5] and then it constructs a Markov chain, which is a stochastic model describing a sequence of possible events from a set of probabilities. In the construction of Markov chains, the intersection-over-union method is used to compute the distance between the selected initial clusters and each candidate point. Finally, this method can be used to continually update the cluster center with each new set of width and height values for the object bounding box, which are only a part of the data selected from the datasets [5]. Due to above two methods, YOLOv3 usually has fairly good performance and fast with low computational cost.

3 Object Detection

Although humans can also detect specific objects in a scene, heavy condition, which is like annoying or lots of object patterns, can decrease human's ability to distinguish specific object. Deep learning is making major advances in solving problems that have resisted the best attempts of the artificial intelligence community for many years [3]. Therefore, due to this possibility, robotics research can expect that the ability of robot can be improved by deep learning and computer vision as being robot's eyes. In this section, we discuss about two object detection frameworks as known as YOLO. The background information of both frameworks is described in Chapter 3.1. Furthermore, Chapter 3.2 explains how both frameworks used on this thesis. Lastly, Chapter 3.3 discuss about the result of validation and training from the used model.

3.1 YOLO

Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Table 1. YOLOv3 runs significantly faster than other detection methods with comparable performance [10]

Since deep learning has advanced with good quality of hardware (GPU&CPU) and vision sensors, lots of researchers focus on developing object detection with various algorithms and the formation of neural networks. YOLO is one of the most popular object detection frameworks recently. The main reason is that this framework is easy to use, fast, and fairy good accuracy like Table 1. As it is the latest branch of YOLO, this thesis decided to use YOLOv3 for object detection of the grasping robot.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Table 2. Darknet-53 [10]

YOLO object detection branches have been modified to get better performance and speed by changing the structure of networks. YOLOv3 also has different structure like below when it compares to last branches and uses multi-scale predictions to get higher accuracy of the detection. In contrast to the architecture of YOLOv2 which uses 19-layer network supplemented with 11 more layers for object detection (darknet-19), YOLOv3 uses darknet-53 to perform feature extraction. Darknet-53 has 53 layer pretrained network from Imagenet and 53 more layers for object detection like Table 2.

Basically, all YOLO branches have utilized bounding box predictions by using dimension clusters as anchor boxes [11]. The network predicts 4 coordinates for each bounding box, t_x, t_y, t_w, t_h. If the cell is offset from the top left corner of the image by (c_x, c_y) and the bounding box prior has width and height p_w, p_h, then the predictions correspond to:

$$b_x = \sigma(t_x) + c_x \quad [10]$$
$$b_y = \sigma(t_y) + c_y \quad [10]$$
$$b_w = p_w * e^{t_w} \quad [10]$$
$$b_h = p_h * e^{t_h} \quad [10]$$

In the training of the model, the framework uses sum of squared error loss to get prediction performance. For each bounding box from YOLOv3, it gets the objectness score by using logistic regression because lots of bounding boxes are overlapped by YOLOv3. To get high accuracy, the score should be 1 if the bounding box prior overlaps a ground truth object by more than any other bounding box prior. By extracting the best scores from bounding boxes, it can make less cost on computation of the loss function.

When it comes to the class classification, the bounding boxes of YOLOv3 have own class label. It simply uses independent logistic classifiers for multi-label classification. Therefore, YOLOv3 uses binary cross-entropy loss for the class predictions [10].

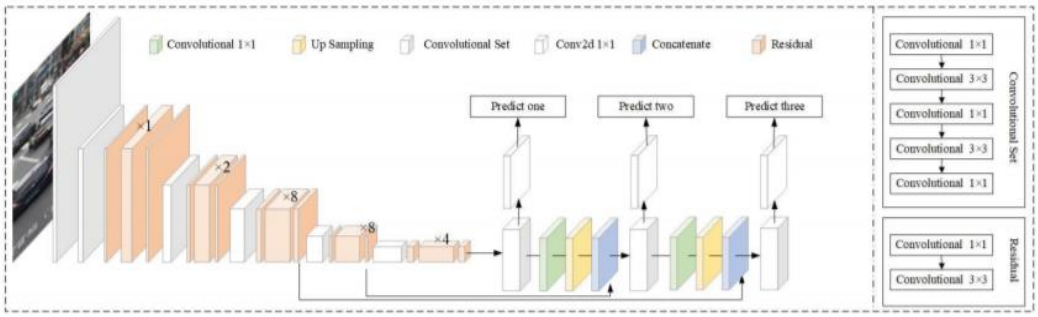


Figure 2. Structure detail of YOLOv3 [12]

To sum up, YOLOv3 processes like this flow:

Step 1. Input the image and scale the image to the standard size.

Step 2. Divide the input image into 13*13, 26*26 52*52 grids of three scales. If the center point of an object falls in the grid unit, the grid unit predicts the object.

Step 3. Using a k-means clustering to determine our bounding box priors on each grid unit. There are three clusters on each grid unit. Because of the three scales, there are a total of 9 clusters per grid unit.

Step 4. Input the image into the network for feature extraction. The model first produces a feature map of 13*13 on a small scale.

Step5. The 13*13 small-scale feature map is first subjected to convolutional set and 2 times upsampling, and then connected to the 26*26 feature map and output the prediction result.

Step 6. the 26*26 feature map output by step5 is subjected to convolutional set and 2 times upsampling, then connected to the 52*52 feature map and output the prediction result.

Step 7. Fusion of features from three scale predictive outputs, Afterward, using a probability score as a threshold to filter out most anchors with low scores. Then using Non-Maximum Suppression (NMS) [32] for post-processing, leaving more accurate boxes.

3.2 Settings and Processes

3.2.1 Input Dataset and Camera Sensor

Since this thesis decided to use YOLOv3 as object detection technique for grasping robot, there are lots of options to set testing environment. First, as the input dataset, we used the images of 3D-printed shapes with various angles and lightning shown in figures 3 and 4.

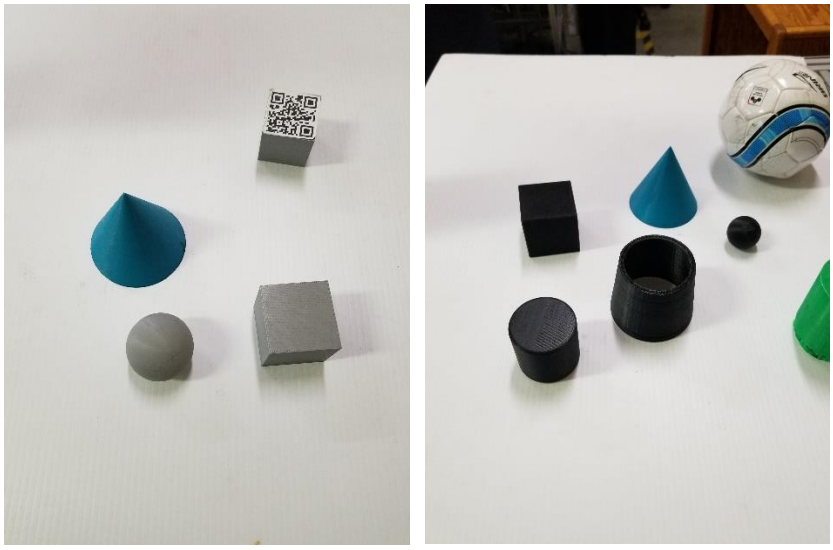


Figure 3&4. Input Images for training YOLOv3 model [14]

We used whiteboard background for the dataset because it improves the accuracy of drawing bounding boxes during training the model. Furthermore, when it comes to the camera sensors, this thesis used Logitech C615 and C920 webcam models. The main reason is that both models are very common webcams, have wide field of view (78°) and has good quality of HD resolution (1080p) for real-time object detection. Most of all, both webcams interact well with OpenCV library.

3.2.2 Model Configuration

Before training our model we needed to convert our raw images from our dataset to annotated images. To annotate the images, we used YOLO_mark image annotation tool. YOLO_mark provides good user interface for marking bounding boxes and recording box coordinates on text file for each input image. The text

file includes the coordinates data like this: <class num> <x_center> <y_center> <width> <height>

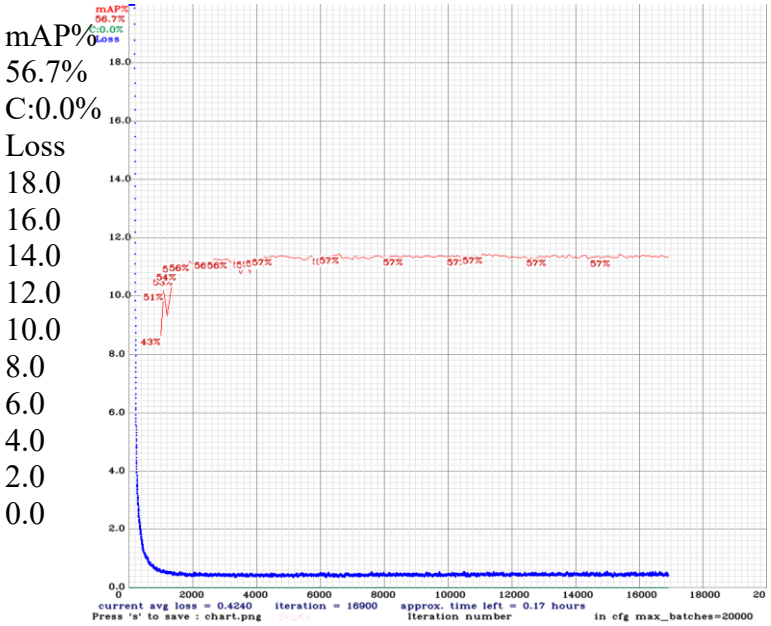
After the annotated dataset generated, the model configuration required to change for custom dataset because YOLOv3 model has default configuration for COCO dataset. Therefore, the model settings should be modified to fit own custom dataset and to get better performance on object detection test. Since our dataset has 4 classes (cone, sphere, cuboid, and cylinder), we used the following settings:

- the number of class = 4
- the number of epochs = 6000~8000
- Network size = (width = 416, height = 416)
- batch size = 32
- filter size = 27
- Learning rate = 0.01 ~ 0.001

Most of all, YOLOv3 provides two options: original YOLOv3 and tiny YOLOv3. This is depending on the number of layers for pre-trained weight. The original one uses 74 convolutional layers, so it has high accuracy but quite slow like less than 10-fps on real-time object detection. Tiny YOLOv3 uses 11 convolutional layers. Since it has 1/7 layers when it compares to the original one, the accuracy is slightly reduced but it is really fast like 30-fps on real-time object detection. In this thesis, we set up both models and trained with our custom dataset.

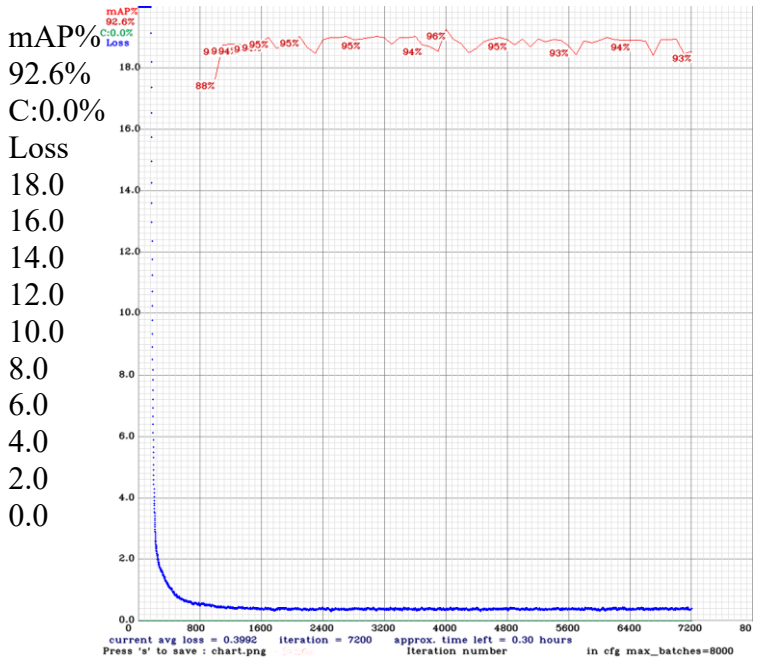
3.3 Result

With various model configurations from *3.2.2*, the custom object detection model can be trained and tested in the real-world with a Logitech C615 webcam. At first, model training with 65 images showed 57% accuracy rate on average like figure 3. With new 200 images, the accuracy could reach on 93% training accuracy like figure 6.



Current avg loss = 0.4240 iteration = 16900 approx. time left = 0.17 hours
iteration number max_batches =20000

Figure 5. Training accuracy chart with 65 images [18]



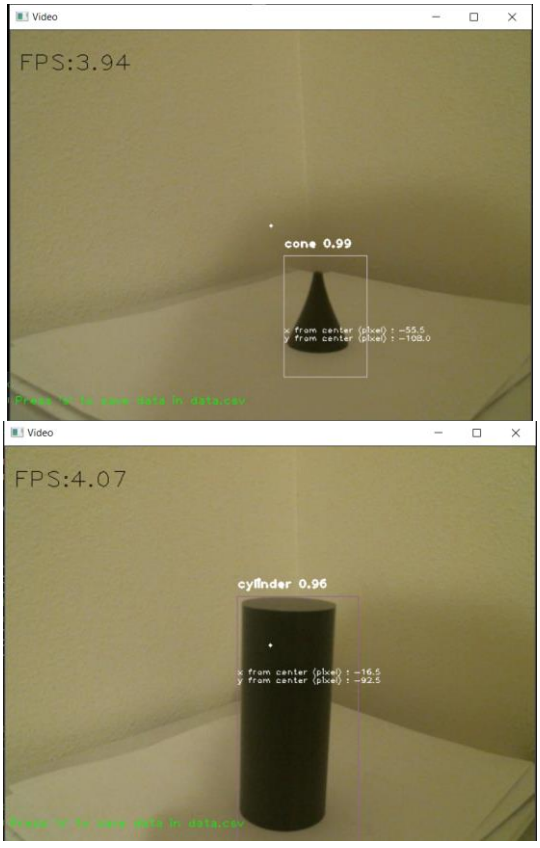
Current avg loss = 0.3992 iteration = 7200 approx. time left = 0.30 hours
iteration number max_batches = 8000

Figure 6. Training accuracy chart with 200 images [19]

Since the custom model is set with 4 classes which are 3D-printed basic shapes (cone, sphere, cylinder, cuboid), we set from 65 of input images to

increase 200 of input images. After we trained the model, the object detection was tested in the real-world. The detection of all trained classes worked well like figure 7,8, and 9. However, the test led to new problems in the object detection model. The first problem is the size of input dataset because 200 images are still small numbers to detect objects with various angles and lightings. Therefore, it needs to get more data with various angles and lightings as the future work. Secondly, sometimes the area of bounding box is unstable like cuboid in the figure 9.

Object detection algorithm needs to not only accurately classify and localize important objects, it also needs to be incredibly fast at prediction time to meet the real-time demands of video processing. However, due to COVID-19, object detection couldn't test in the lab where has a good performance pc. With low performance laptop, the average FPS was 4~5. This could make distortion on video frame and led to draw unstable bounding boxes.



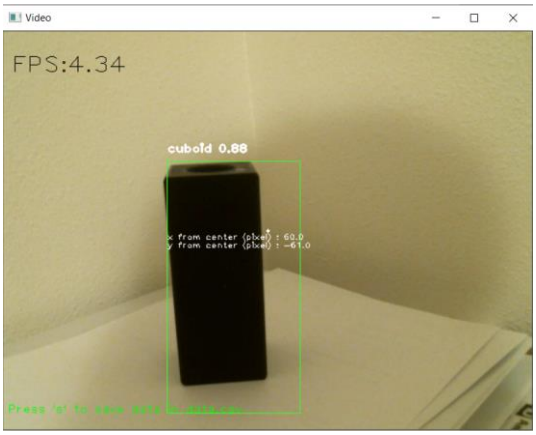


Figure 7,8,9. Object Detection Test with Basic Shapes [20]

4 Computer Vision with OpenCV

When it comes to the computer vision, MATLAB and OpenCV are the best options. As one of core techniques for this thesis, the OpenCV Python library was utilized to measure object size and distance from camera to object. OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms [8].

OpenCV is optimized well to test with webcam and to get data from Arduino boards. This thesis used ultrasonic sensor with Arduino UNO to get the distance between camera and detected object. Therefore, OpenCV was the best option for the object measurement. In addition, OpenCV with Python is easy to use when it compares to OpenCV with C++ because OpenCV with Python API has concise documents for implementation. Lastly, OpenCV with Python provides good visualization and user interface tools.

4.1 Object Contours and Measurement

The object measurement is the second phase of test in this thesis, so it requires the bounding box information from object detection phase. After the object detection model found desired object, bounding box pixel data from the whole image frame is cut out. Then, the sliced pixels are changed to grayscale and added gaussian blur to get better results. In the final step, the OpenCV's contour functions are utilized for drawing contours of the detected objects and calculating contour area. Meanwhile, the various threshold functions were tested like below:

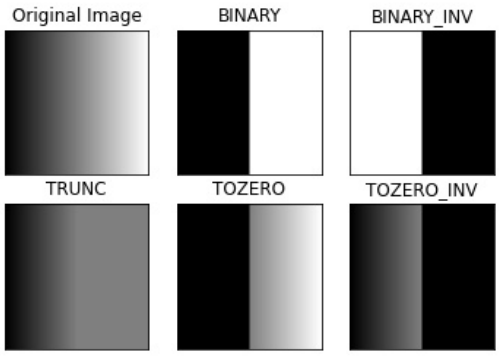


Figure 10. The result of threshold functions in OpenCV [9]

The above threshold functions have their own advantages to draw contours of the detected object. For example, TRUNC function is good for barely lighted images because it gets a threshold between thick dark color and light dark color. Since our test environment uses white background board, BINARY type function is a great option.

The main reason is that drawing contours with original image frame has low accuracy because the boundary between different color pixels is not clear. Therefore, low accurate contours of the detected object cause low accuracy of object size measurement. Since the testing environment used white background, BINARY, RETR_EXTERNAL, and CHAIN_APPROX_NONE are used for threshold parameters. With OpenCV user interface, the threshold values can be changed with scale bars in the real time.

4.2 Settings

4.2.1 Measurement Sensor

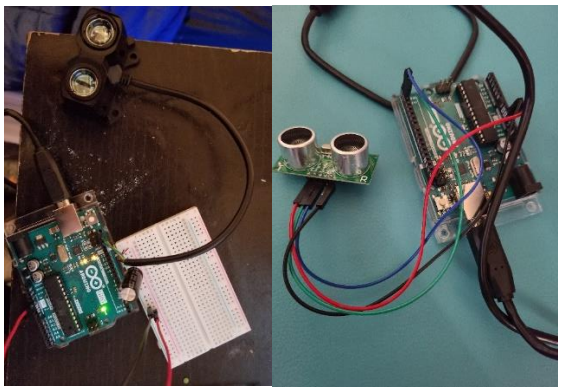


Figure 11&12. LIDAR-Lite v3HP(right) and US-100(left) ultrasonic sensor [15]

For measuring the distance between camera and detected object, we considered two options. First, we tested with LIDAR-Lite v3HP (Garmin) which is high-speed optical distance measurement sensor. Although LIDAR-Lite v3HP has high accuracy up to 40-meters and fast response time (1kHz measurement speed) [16], we figured out that it is good for over 50cm distance. Since this thesis requires to measure distance less than 50cm to grasp detected object with robot arm, we gave up using LIDAR-Lite v3HP model. After comparison between two sensors, we decided to use US-100 ultrasonic sensor because US-100 model has fairly good accuracy from 2cm to 450cm measurement and compatible with Arduino board.

4.2.2 FOV and Camera Matrix

A camera is mapping between the 3D world and a 2D image. In other words, the camera does 3D projection to write 3D information on 2D frame. Due to 3D projection, every object in the single frame looks different depending on camera angle. Therefore, we wrote camera matrix code to get better object size measurement in the bounding box from object detection and tested it in real-time.

The main purpose of the matrix code is that calculate the coordinates of a 3D point and center of camera lens based on focal length and field of view. Then, get normalized pixel data on 2D image plane. With the pixel data, it allows to get the distances of x-axis and y-axis from the center of camera.

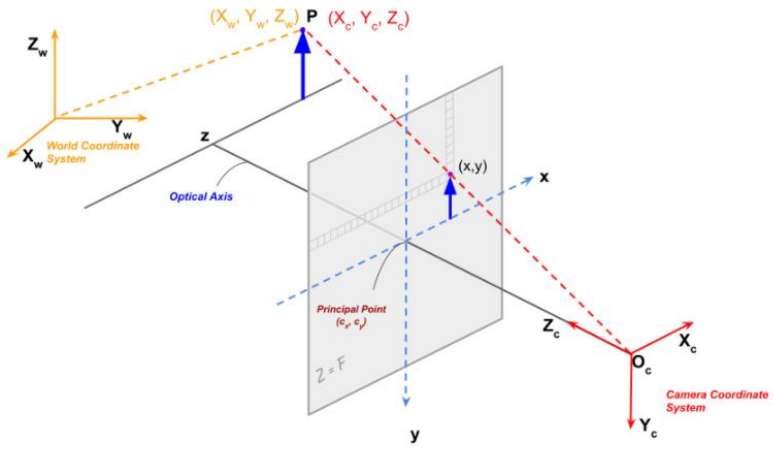


Figure 4. The projection of point P onto the image plane is shown [17]

After we got x-axis and y-axis values from camera matrix application, we need to get object's height and width. At first, this thesis used object contours with threshold parameters, pixel data, and camera matrix, and ultrasonic sensor data to measure detected object size in the real time. Since the distance from the camera to the detected object has good accuracy with ultrasonic sensor, the equation of Field of View (FOV) for camera lens can be considered. The equation of FOV can get from the basic thin lens formula like below:

$$\frac{1}{d_o} + \frac{1}{d_i} = \frac{1}{f} \qquad [22]$$

- d_o is the distance (measured along the axis) from the object to the center of the lens
- d_i is the distance (measured along the axis) from the image to the center of the lens

- f is the focal length of the lens = 3.67 mm (Logitech C920 and C615 Webcam)

And then the basics lens optics equations can be used for calculating the real object height like this:

$$object\ height\ on\ sensor\ (mm) = \frac{sensor\ height\ (mm)\ \times object\ height\ (pixels)}{sensor\ height\ (pixels)} \quad [23]$$

$$Real\ Object\ Height\ (mm) = \frac{Distance\ to\ Object\ \times Object\ height\ on\ sensor\ (mm)}{focal\ length\ (mm)} \quad [23]$$

4.3 Result

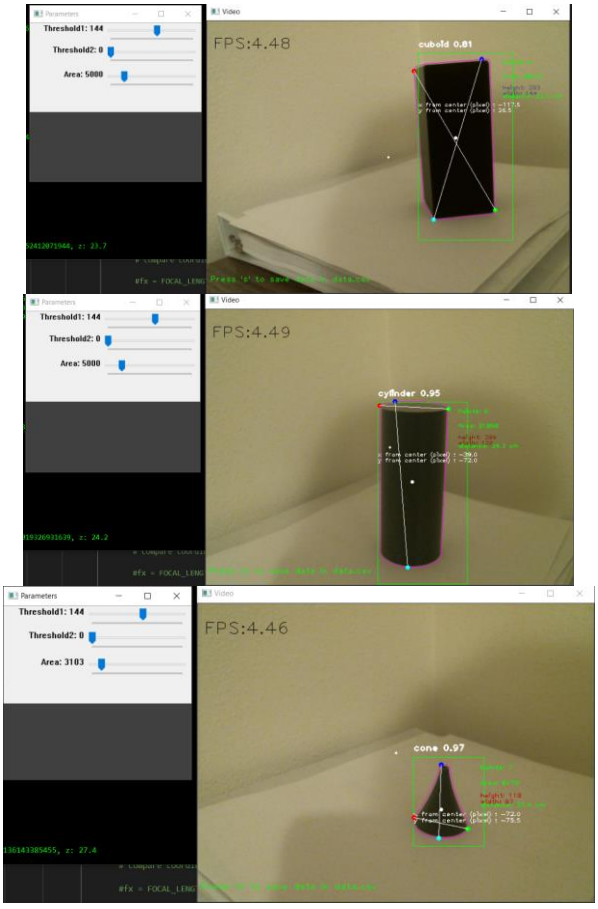


Image 6,7,8. Object Detection and Measurement Test with Basic Shapes [21]

After set up the camera and ultrasonic sensors from *4.2*, the object measurement can be tested in the real-world. Since the object measurement aims for getting distance from camera to object, distance from camera center to object center, and object size with OpenCV contours and camera matrix, the comparisons between measured and real sizes are the main way to get the result. The error rate is calculated by below formula and then get the average error rates of each shape measured 30 times for X-axis, Y-axis, Z-axis (distance from camera to object), object height, object width, and various distance (10 ~ 20 cm).

$$Error\ Rate\ (\%) = \frac{|\ Calculated\ Value - Exact\ Value\ |}{|\ Exact\ Value\ |} \times 100\%$$

Error rate (%)+/- Std Dev(%)	**X-axis**	**Y-axis**	**Z-axis**	**Height**	**Width**
Cylinder1	8.5% +/- 0.8%	6.7% +/- 1.0%	1% +/- 0.2%	33.4% +/- 9.8%	37.1% +/- 12.8%
Cylinder2	7.6% +/- 0.6%	3.5% +/- 0.4%	0.7% +/- 0.2%	32.3% +/- 10.3%	32.5% +/- 13.0%
Cuboid	8.2% +/- 1.0%	7.1% +/- 1.1%	0.3% +/- 0.1%	34.2% +/- 8.8%	36.4% +/- 11.4%
Cone	7.2% +/- 0.9%	2.5% +/- 0.5%	0.5% +/- 0.1%	31.2% +/- 9.6%	34.2% +/- 12.3%

From the real-world test, we could check the application of camera matrix worked well because X-axis and Y-axis showed about 8% error rates averagely. We assume that the main reason of error rate is from the unstable center of object. The center of object kept changing depending on the object contours. Therefore, this error rate could be decreased by high accurate object contours. Z-axis, which is the distance from camera to the detected object, had less than 1% error rate. This error rate is acceptable because US-100 sensor uses ultrasound reflection to

measure distance. In other words, if a reflect surface is not even, then it could measure slight off from exact distance.

However, the object's height and width were definitely off from exact values with over 30% error rates. We assume that these high error rates are caused by frame drop by non-gpu testing (laptop), unstable contours, non-fixed focal length of the webcam, and auto-focus ability of the webcam. The frame drop by a bad performance machine causes the data processing speed doesn't sync on the frame, so some frames could calculate the measurement with the previous frame's pixel data. Furthermore, the unstable contours, non-fixed focal length, and auto-focus ability of the webcam make it difficult to get exact pixel data of the detected object. For the unstable contours, it causes huge error rates on the results. Most of all, the testing conditions didn't match to get the best results with FOV equations.

5 Overall Discussion

All attempts and tests showed potential for merging video data with ultrasonic sensors for grasping applications. The limited validation environment and error rate of measurement lead further insight into the object detection model and the importance of the test environment. The first research question was: *Can the custom object detection model detect the desired object with a webcam and the object measurement works on various distances in the real time?* The custom object detection model in this thesis can detect only trained objects (Chapter 3.3). However, the real-world test finds new problem on customed dataset and test environment like the performance of using computer for the model, lighting, and camera quality and angles. From the experiments, the object measurement can work on various distance from 15cm to about 60cm. However, the field of view of webcam and auto-focusing mode makes difficult to capture the object in the camera frame within 15cm. Therefore, as the future work, better camera sensor is needed to work on more various distances.

The flexibility of custom model was proved by the second research question: *Can the object detection model connect with the object measurement*

module in python? The connectivity between object detection model and object measurement modules in python sketch is core technique for this thesis because the workload in the real-time is phase-structure with condition statements. When the webcam is running, the first phase is object detection part. If the desired object is detected, then python script goes to object measurement phase in the real-time.

The task environment generalizability was evaluated by the third research question: *Can the object detection and object measurement abilities improve the performance of the grasping robot?* The primary research goal with various tests was demonstrating the reliability of the object detection model and the object measurement module in the real time. As the test results, the object detection model is capable of generating fairly good accuracy to detect trained objects in the camera frame. Also, the object measurement module has low error rates on the calculation of the distances from center of camera to the center of the detected object in the real time (X-axis, Y-axis). Ultrasonic sensor works well to get the distance from camera to the detected object, but data receiving time from Arduino board to base-computer is little bit slow. However, the main problem from tests is high error rate on object height and width measurement from the average error rate chart (Chapter 4.3). The object contours (Chapter 4.1) and FOV equation for height & width (Chapter 4.2.2) may explain their high error rates.

6 Future Work

Through this thesis project, we have found that computer vision and deep learning technology can be widely useful, not only for the supporting grasping robot arm because the object detection and measurement has combined various image processing techniques with deep learning. For example, object detection can be useful for image annotation, activity recognition, face detection, face recognition, video object co-segmentation. Due to the specialty and flexibility of

computer vision and deep learning, it could be used for agriculture, industrial fields, and public safety. However, the real-world tests of the custom object detection and object measurement showed problems and what to do list as the future works. First, the object detection model needs more input dataset to detect in any test scenario because the test proved that lighting and camera angles affect detection ability. Next, the performance of the base-computer affects the real-time frame and detection ability. Therefore, we consider using Nvidia's Jetson model which is an embedded AI computing device with gpu in the future.

Also, the object measurement module needs more work in the future. Although the camera matrix was working well in the test, drawing contours on detected objects has quality issues. The object contours give impacts on next steps like the calculation of camera matrix and pixel data information. We saw that X-axis and Y-axis are close to the exact values, but object height and width have over 30% error rates. As the future works, we should collect more images for each class with various angles and lightings to get better bounding boxes and we should consider changing the camera sensor and test environment. Lastly, with those supportive technologies, the robot arm can be tested as an automatic gardener to prove usability and flexibility of the supportive technologies. For example, the object detection ability would be helpful to distinguish between weed and flower. After then, the sensor will measure distance and then pick a specific flower with a robot arm.

7 Conclusion

This honors thesis explores and validates a custom object detection model from YOLOv3 algorithm and object measurement with camera & ultrasonic sensors and OpenCV library to be supportive abilities for robotic grasping arm KINOVA lab at Oregon State University. The primary research objective was combining the sensors with their housing for the grasping arm, developing a custom object detection model and object measurement module with camera and

ultrasonic sensors, and validating them in the real world. The 3D-printed basic shapes and white background were used to be input dataset for training and testing the custom model. The validation demonstrated that the object detection model has good accuracy to detect trained classes, but it revealed two problems on dataset size and base-computer performance. In addition, the validation for object measurement module proved that the camera matrix for projecting 3D world plane to 2D image plane has good accuracy. However, the validation demonstrated that drawing contours and the calculation for detected object's height and width need to improve for decreasing high error rates.

Although COVID-19 slowed the progress of this thesis, I could conclude the research questions and made the future plan with great advice from graduate researchers and the thesis mentor in the lab. As the primary contribution of this thesis, it highlights the reliability and connectivity between object detection model and object measurement module in the real-time. Most of all, this thesis leads to future work and practical usages in the industry.

References

[1] Y. H. Ong, J. Morrow, Y. Qiu, K. Gupta, R. Balasubramanian and C. Grimm, "Near-contact grasping strategies from awkward poses: When simply closing your fingers is not enough*," 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 2019, pp. 646-651, doi: 10.1109/IROS40897.2019.8968468.

[2] E. Rimon and J. Burdick, “Introduction and Overview,” in The Mechanics of Robot Grasping, Cambridge: Cambridge University Press, 2019, pp. 1–14.

[3] S. Rajalingappaa, “Deep Learning for Computer Vision : Expert Techniques to Train Advanced Neural Networks Using TensorFlow and Keras,” Paths International Ltd, 2018, pp. 6

[4] Y. Lecun, Y. Bengjo, and G. Hinton, “Deep learning,” Nature 521, 2015, pp. 436–444, doi: https://doi-org.ezproxy.proxy.library.oregonstate.edu/10.1038/nature14539[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” arXiv.org, 09-May-2016. [Online]. Available: https://arxiv.org/abs/1506.02640

[5] Z. Liquan and L. Shuaiyanm, “Object Detection Algorithm Based on Improved YOLOv3.” Electronics (Basel), vol. 9, no. 3, 2020, pp. 537.

[6] I. Culjak, D. Abram, T. Pribanic, H. Dzapo and M. Cifrek, "A brief introduction to OpenCV," 2012 Proceedings of the 35th International Convention MIPRO, Opatija, 2012, pp. 1725-1730.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” arXiv.org, 09-May-2016. [Online]. Available: https://arxiv.org/abs/1506.02640

[8] OpenCV, “About OpenCV”, OpenCV.org. [Online]. Available: https://opencv.org/about/

[9] OpenCV, “Image Thresholding”, OpenCV.org. [Online]. Available: https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html

[10] J. Redmon, and A. Farhadi, “YOLOv3: An Incremental Improvement ,” arXiv.org, 08-April-2018. [Online]. Available: https://arxiv.org/pdf/1804.02767.pdf

[11] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, pages 6517–6525. IEEE, 2017.

[12] Mao, Qi-Chao & Sun, Hong-Mei & Liu, Yan-Bo & Jia, Rui-Sheng. (2019). Mini-YOLOv3: Real-Time Object Detector for Embedded Applications. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2941547.

[13] The MathWorks, Inc., “Deep Learning”, mathworks.com. [Online] Available: https://www.mathworks.com/discovery/deep-learning.html

[14] J. Jeong, “Image of 3D-printed shapes”, 2020.

[15] J. Jeong, “Image of LIDAR-Lite v3HP sensor and US-100 ultrasonic sensor”, 2020.

[16] Garmin Ltd., “LIDAR-Lite v3HP specification”, buy.garmin.com. [Online] Available: https://buy.garmin.com/en-US/US/p/578152/pn/010-01722-10#specs

[17] S. Mallick, “Geometry of Image Formation”, Learn OpenCV, February 20, 2020. [Online] Available: https://learnopencv.com/geometry-of-image-formation/

[18] J. Jeong, “Training Accuracy Chart with 65 Images”, 2020.

[19] J. Jeong, “Training Accuracy Chart with 200 Images”, 2020.

[20] J. Jeong, “Object Detection Test with Basic Shapes”, 2020.

[21] J. Jeong, “Object Detection and Measurement Test with Basic Shapes”, 2020.

[22] C. Catharine H, "Resource Lesson: Thin Lens Equation". *PhysicsLab.org*. [Online] Available: http://dev.physicslab.org/Document.aspx?doctype=3&filename=GeometricOptics_ThinLensEquation.xml

[23] W. Fulton, “Math of Field of View (FOV) for a Camera and Lens”, 2018. [Online] Available: https://www.scantips.com/lights/fieldofviewmath.html