

**FISH PELLETT MEASUREMENT SYSTEM FOR FOOD INDUSTRY**

**BY**

**NGO KOK WEI**

**A REPORT**

**SUBMITTED TO**

**Universiti Tunku Abdul Rahman**

**in partial fulfillment of the requirements**

**for the degree of**

**BACHELOR OF INFORMATION TECHNOLOGY (HONOURS)**

**COMPUTER ENGINEERING**

**Faculty of Information and Communication Technology**

**(Kampar Campus)**

**FEBRUARY 2025**

**FISH PELLETT MEASUREMENT SYSTEM FOR FOOD INDUSTRY**

**BY**

**NGO KOK WEI**

**A REPORT**

**SUBMITTED TO**

**Universiti Tunku Abdul Rahman**

**in partial fulfillment of the requirements**

**for the degree of**

**BACHELOR OF INFORMATION TECHNOLOGY (HONOURS)**

**COMPUTER ENGINEERING**

**Faculty of Information and Communication Technology  
(Kampar Campus)**

**FEBRUARY 2025**

# **COPYRIGHT STATEMENT**

© 2025 Ngo Kok Wei. All rights reserved.

This Final Year Project report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Information Technology (Honours) Computer Engineering at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgement has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

## **ACKNOWLEDGEMENTS**

I would like to express my sincere thanks and appreciation to my supervisor, Dr Teoh Shen Khang and my moderator, Ts Dr Chang Jing Jing, who have given me this bright opportunity to engage in this project. Besides that, they have given me a lot of guidance to complete this project. When I was facing problems in this project, the advice from them always assisted me in overcoming the problems. A million thanks to you.

Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

## **ABSTRACT**

This project aims to address the need for precise measurement of fish pellets in the industrial sector by leveraging technology. This project will involve several fields such as computer vision, embedded systems and deep learning. Before that, manpower is required to measure the fish pellets individually. By using this traditional method, the accuracy and efficiency are low. Therefore, a new method using technology will be introduced to solve this problem. The core technology used in this project is computer vision and deep learning. Initially, a camera will be set up to capture the fish pellet. Then, the image will be processed in a trained model to detect the fish pellet. Then, an algorithm will be used to determine the fish pellet's diameter based on the result of the detection. To improve the consistency, Raspberry Pi will be chosen as the CPU of this project. The camera will be interfaced to it, and the trained model will be imported into it. A user-friendly GUI will also be provided to display the output information of the system. Python will be selected as the programming language in this project due to its extensive library support, such as OpenCV for computer vision and TensorFlow for deep learning. As a result, the GUI should perform the fish pellet detection and diameter calculation, which has a bounded box and label of the diameter value on each fish pellet as the output. In conclusion, this project will provide a more efficient solution than the traditional method.

Area of Study (Minimum 1 and Maximum 2): Internet of Things Vision

Keywords (Minimum 5 and Maximum 5): Computer Vision, Deep Learning, IoT, Image Processing, Algorithm Design

# TABLE OF CONTENTS

<b>TITLE PAGE</b>	<b>i</b>
<b>COPYRIGHT STATEMENT</b>	<b>ii</b>
<b>ACKNOWLEDGEMENTS</b>	<b>iii</b>
<b>ABSTRACT</b>	<b>iv</b>
<b>TABLE OF CONTENTS</b>	<b>v</b>
<b>LIST OF FIGURES</b>	<b>vii</b>
<b>LIST OF TABLES</b>	<b>ix</b>
<b>LIST OF ABBREVIATIONS</b>	<b>x</b>

<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 Problem Statement and Motivation	1
1.2 Objectives	2
1.3 Project Scope and Direction	2
1.4 Contributions	3
1.5 Report Organization	4

<b>CHAPTER 2 LITERATURE REVIEW</b>	<b>5</b>
2.1 History of Computer Vision	5
2.2 History of Deep Learning	5
2.3 Convolutional Neural Network	6
2.3.1 Review of Convolutional Neural Network	6
2.3.2 CNN Model Comparison	7
2.4 Object Detection Algorithm	8
2.4.1 Review of Object Detection Technology	8
2.4.2 Region-based Convolutional Neural Network	8
2.4.3 Fast R-CNN	9
2.4.4 Faster R-CNN	10
2.4.5 YOLO	11
2.4.6 SSD	12
2.4.7 Mask R-CNN	13
2.4.8 Performance of Object Detection Algorithm	14
2.5 Object Measurement Project	15

<b>CHAPTER 3 SYSTEM METHODOLOGY/APPROACH</b>	<b>18</b>
3.1 System Requirements	18
3.1.1 Hardware	18
3.2 Mask-RCNN Model Design	19
3.3 UI and Backend Design	20
3.4 Ratio Determination Algorithm Design	20
3.5 Diameter Algorithm Design	21
3.5.1 Circle Estimation	22

3.5.2 Ellipse Estimation	22
3.5.3 Convex Hull Approach	22
3.5.4 Average Diameter Approach	22
3.5.5 Discussion of Diameter Algorithm	23
3.5.6 Determination Algorithm	23
3.6 Project Timeline	25
<b>CHAPTER 4 SYSTEM DESIGN</b>	<b>26</b>
4.1 Overall System Design	26
4.2 Prototype Design	30
<b>CHAPTER 5 SYSTEM IMPLEMENTATION</b>	<b>32</b>
5.1 Prototype	33
5.2 UI Design	36
5.3 Difficulties and Challenges	
<b>CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION</b>	<b>37</b>
6.1 Result of Mask-RCNN Model	37
6.2 Result of Ratio Determination Algorithm	38
6.3 Discussion of Height and Width Measurement	39
6.4 Result and Discussion of Diameter Algorithm	42
6.5 Evaluation and Discussion of Time Efficiency	45
6.6 Objectives Evaluation	46
<b>CHAPTER 7 CONCLUSION AND RECOMMENDATION</b>	<b>48</b>
7.1 Conclusion	48
7.2 Recommendation	49
<b>REFERENCES</b>	<b>50</b>
<b>POSTER</b>	

# LIST OF FIGURES

<b>Figure Number</b>	<b>Title</b>	<b>Page</b>
Figure 2.1	Example Architecture of CNN	7
Figure 2.2	Working Principle of R-CNN	9
Figure 2.3	Bounding Box Generation and Prediction	11
Figure 2.4	Example of Anchor Boxes	13
Figure 3.1	Model Training and Set Up.	19
Figure 3.2	Relationship Between Height and Width	21
Figure 3.3	Project Timeline for FYP 1	25
Figure 3.4	Project Timeline for FYP 2	25
Figure 4.1	Flow Chart of Raspberry Pi Capturing the Input	26
Figure 4.2	Flow Chart of Personal Computer	27
Figure 4.3	Flow Chart of Visualization Algorithm	28
Figure 4.4	Flow Chart of Raspberry Pi Capturing the Input	29
Figure 4.5	Flow Chart of Overall System	29
Figure 4.6	Top View of The System	30
Figure 4.7	Side View of The System	30
Figure 5.1	Raspberry Pi and The Measuring Station	32
Figure 5.2	Measuring Station	32
Figure 5.3	Top View of the Measuring Station	32
Figure 5.4	Camera Angle	32
Figure 5.5	Webpage of the Project	33
Figure 5.6	Measurement Methods	33
Figure 5.7	Aruco Detection	34
Figure 5.8	Object Match	34
Figure 5.9	Height of Camera	34
Figure 5.10	Height of Frame	34
Figure 5.11	Unit Selections	35
Figure 5.12	Calculation Preference Selections	35
Figure 6.1	Result of The Model	37



Figure 6.2	Result of Aruco Makers	38
Figure 6.3	Result of Object Matching Method	38
Figure 6.4	Result of Height of Camera Method	38
Figure 6.5	Result of Height of Frame Method	38
Figure 6.6	Algorithm Calculation	39
Figure 6.7	Width of Small Fish Pellet	39
Figure 6.8	Height of Small Fish Pellet	39
Figure 6.9	Width of Big Fish Pellet	39
Figure 6.10	Height of Big Fish Pellet	39
Figure 6.11	Comparison of Pixels Between Big and Small Fish Pellets	40
Figure 6.12	Different Degrees of Rotation of The Same Pellets	41
Figure 6.13	Method 1 for Manual Measurement	41
Figure 6.14	Method 2 for Manual Measurement	41
Figure 6.15	Result of Method 1	42
Figure 6.16	Result of Method 2	42
Figure 6.17	Result of Circle Estimation	42
Figure 6.18	Result of Ellipse Estimation	42
Figure 6.19	Result of Convex Hull Approach	43
Figure 6.20	Result of Average Diameter Approach	43
Figure 6.21	Result of Estimating the Shape of Fish Pellet	44

## LIST OF TABLES

Table Number	Title	Page
Table 3.1	Specifications of Computer	18
Table 3.2	Comparison Between Each Diameter Estimation Method	23
Table 3.3	Specifications of Fish Pellets	24
Table 6.1	Comparison Between Algorithm Calculation and Real Value	40
Table 6.2	Comparison Between Each Approach	43
Table 6.3	Comparison Between Two Versions of the System	45
Table 6.4	Comparison of Time Used Between Different Flows of Data Transmission	46

## LIST OF ABBREVIATIONS

<i>AI</i>	Artificial Intelligence
<i>CNN</i>	Convolutional Neural Networks
<i>GPU</i>	Graphic Processing Units
<i>TCN</i>	Temporal Convolutional Networks
<i>R-CNN</i>	Region-based Convolutional Neural Network
<i>RPN</i>	Region Proposal Network
<i>SVM</i>	Support Vector Machine
<i>YOLO</i>	You Only Look Once
<i>SSD</i>	Single Shot Detector
<i>GUI</i>	Graphic User Interface
<i>VIA</i>	VGG Image Annotator
<i>VGG</i>	Visual Geometry Group
<i>MQTT</i>	Message Queuing Telemetry Transport
<i>FLANN</i>	Fast Library for Approximate Nearest Neighbors
<i>RAM</i>	Random Access Memory

## CHAPTER 1

### Introduction

This chapter will provide an overview of this project, including the problem statement, its significance, and the aims of the project. After reading this chapter, you should be able to understand the concept of this project.

#### 1.1 Problem Statement and Motivation

Several industries will face some problems in dealing with the measurement of their manufacturing products. For example, some products such as nuts will need to be measured to get their length. The industry might need this information to classify the nuts and calculate the deviation or error of the product. Currently, many industries are still using the traditional methods which is using manpower to measure the products. The disadvantages of this method include the inaccuracy of data and it will require more costs for manpower. Furthermore, the inaccuracy of measurement value will also cause loss as a wrong price might be set. They might also face complaints from customers due to this reason.

In the technical aspect, the measurement system can be done by using Graph Clustering and Variational Image Segmentation methods [1]. However, it will face a limitation due to its complexity and it is sensitive to noise and ambiguity. These might impact the overall performance of the measurement system.

Another solution to measure objects is to use deep learning techniques such as YOLO to identify the objects and then calculate the length value. These models have been used in several object detection projects such as traffic sign detection for autonomous driving [2], COVID-19 detection based on chest X-ray images [3] and more. On the other hand, it also faces a limitation with a lower accuracy in detecting the object. Therefore, the choice of a suitable model is highly related to the performance of the system.

## CHAPTER 1

This project will develop a measurement system for an industry that manufactures fish pellets. Hence, a suitable method is needed to perform this function. The best way in this project is to use the object detection method to detect the fish pellet first, and then calculate its length and width.

### 1.2 Objectives

The main objective of this paper is to develop an Automated Fish Pellet Measurement System using computer vision and deep learning techniques to address the inefficiencies and inaccuracies associated with manual measurement processes in the industrial sector. Details should be explained below:

- To present an improved and efficient method for determining fish pellet diameter to address the current industry issue.
- To develop a deep learning model to predict the area of the fish pellet.
- To develop an algorithm to calculate the height and width, or diameter of the fish pellet by extracting the information from the detection.
- To integrate the computer vision, deep learning model and algorithm with the Raspberry Pi hardware.
- To develop a user-friendly GUI and measurement station for convenient observation.
- To validate the performance and accuracy of the measuring system.

### 1.3 Project Scope and Direction

At the culmination of this project, the deliverables will encompass a comprehensive solution tailored to meet the needs of the industrial sector:

- Software Application: A fully developed software application capable of capturing, processing, and analyzing video footage of the fish pellet. This software will

## CHAPTER 1

incorporate advanced computer vision algorithms and deep learning models to accurately measure the diameter of each fish pellet.

- **Hardware Integration:** Integration of Raspberry Pi-based hardware components to ensure consistent and reliable processing of the system.
- **Graphical User Interface (GUI):** Development of a user-friendly GUI to facilitate easy monitoring and management of the system. The GUI will provide an intuitive visualization of output information, including bounded box and diameter labels for each detected fish pellet, enhancing usability and accessibility.
- **Training Data and Model:** Provision of training data used for training the deep learning model, along with the trained model itself. This ensures transparency and reproducibility while enabling users to fine-tune the model as per specific requirements.

The project scope encompasses the development of both software and hardware components, with a focus on delivering a robust, scalable, and user-friendly solution. By automating the fish pellet measurement process, this project aims to significantly enhance efficiency, accuracy, and productivity within the industrial sector, thereby providing tangible benefits to stakeholders.

### **1.4 Contributions**

This project will provide a solution to the food industry which can produce some advantages for them. As an outcome, the accuracy and efficiency of fish pellet measurement will be increased, and the cost of manpower will be reduced. Furthermore, exposure to the current technology in the industry will be achieved by bringing the latest technology inside. This will become a contribution to the achievement of Industry Revolution 4.0. Additionally, close contact between the industry and the university can also be maintained. In summary, this project will introduce a better way to measure the fish pellet and contribute to several divisions.

### **1.5 Report Organization**

The details of this research are presented in the following chapters. Chapter 2 provides a review of relevant background information, while Chapter 3 outlines the application of the proposed methodology in this project. Chapter 4 will show the flow and the prototype diagrams. Then, Chapter 5 reports the work done and the difficulties met. Chapter 6 shows the results obtained, discussion and improvement done. Furthermore, Chapter 7 will conclude this project and provide some recommendations.

## CHAPTER 2

### Literature Review

#### 2.1 History of Computer Vision

In the 1960s, scientists began making programs to look at pictures, but they could only do basic stuff because computers weren't very powerful back then [4]. In the 1970s, they got better at it and made programs that could understand images a bit more. By the 1980s and 1990s, they started using machine learning to teach computers to recognize things in pictures. Then, in 2001, they came up with a cool way to find faces in photos called the Viola-Jones algorithm [4]. In the 2000s and 2010s, things took off with deep learning, especially using something called convolutional neural networks (CNNs), which let computers do even fancier stuff like recognizing objects well [4].

#### 2.2 History of Deep Learning

Deep learning is a type of smart computer learning that goes way back to 1943 when Walter Pitts and Warren McCulloch made a computer model inspired by how brains work. Even though there were sometimes when people didn't believe in it, called Artificial Intelligence winters, it kept growing. In the 1960s, Henry J. Kelley and Stuart Dreyfus made things better by figuring out how to train deep models better. Then in the 1970s, Kunihiro Fukushima made convolutional networks, which are good at recognizing patterns in pictures. In 1997, Sepp Hochreiter and Juergen Schmidhuber made something called LSTM to help remember stuff better in these networks. Between 2000 and 2010, computer chips got better, which fixed some problems like the Vanishing Gradient Problem. Fei-Fei Li made a huge difference in 2009 by making ImageNet, a big collection of labelled pictures for computers to learn from. By 2011, computers got even faster, especially the



## CHAPTER 2

ones with graphics cards, which made training convolutional networks much quicker. This led to big improvements like AlexNet [5][6].

### **2.3 Convolutional Neural Network**

#### **2.3.1 Review of Convolutional Neural Network**

Convolutional Neural Networks (CNNs) have been around since the 1980s. They were inspired by how our eyes work, especially the research of David Hubel and Torsten Wiesel in the 1950s [7]. Dr Kunihiro Fukushima made a model called the neocognitron in the 1980s, which was like the starting point for CNNs because it copied how our eyes process things [8]. In the 1990s, Yann LeCun made LeNet-5, which was a big step forward because it used layers to understand pictures better, unlike older computer programs [9].

CNNs became popular in the 2000s because they could work super fast on special computer chips called GPUs. This made them useful for more than just looking at pictures [10]. They started being used in things like understanding sounds and predicting financial stuff [9]. For example, DeepMind made something called WaveNet that showed how well CNNs could understand sounds using fancy techniques like dilated convolutions and residual connections [9].

CNNs kept getting better, especially after AlexNet won a big contest in 2012 by being good at recognizing things in pictures [7]. This made people excited, and they made even more improvements, like Temporal Convolutional Networks (TCNs) in 2018, which are good at understanding sequences of things, like words in a sentence or sounds in speech [9]. CNNs are still being improved because they're good at finding complicated patterns in data, which makes them useful for lots of different things.

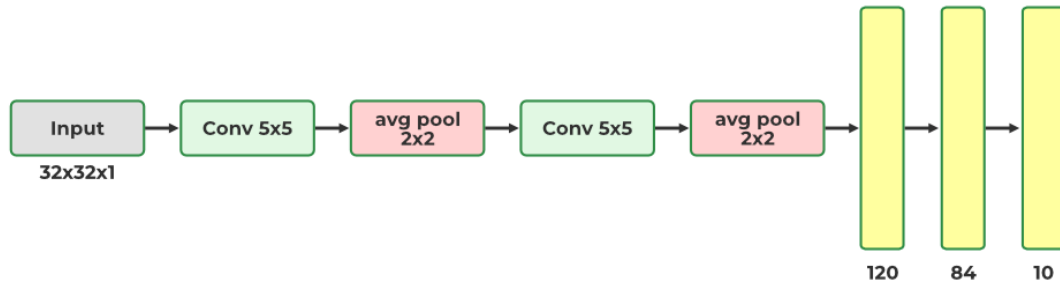


Figure 2.1 Example Architecture of CNN

Figure 2.1 shows an example architecture of a CNN model. It involves two convolutional layers (green), two average pooling layers (red), and three fully connected layers (yellow). In practice, the model will be more complex to improve its accuracy.

### 2.3.2 CNN Model Comparison

By comparing different CNN models and how well they work, it's clear that things like the design of the model and the optimizer used are important. Here are some important things we've learned from comparing them:

- **Accuracy:** Some CNN models, like DenseNet121 and VGG-11M, can be super accurate, with scores ranging from 98.97% to 99.80% [11][12]. Others, like ResNet-50 with the SGD optimizer, can still do well, reaching around 96.35% accuracy [13].
- **Efficiency:** The VGG-11M model is known for being very efficient, while still being accurate, with a top accuracy of 99.80% [12]. It's also important to pick the right model and have a good mix of examples in the training set to get the best results [12].
- **Model Performance:** When comparing different models, like fully connected networks, simple CNNs, and pre-trained networks like MobileNet, it has obvious differences in accuracy. For example, MobileNet did the best with an accuracy of 89.17%, beating out other models like CNN and fully connected networks [14].

- **Architectural Design:** The architecture of a CNN will affect how well it works. Changing things like how many layers it has or how the convolutional layers are set up can make it better at recognizing things and improve its overall performance [12].

In the end, picking the right CNN design, optimizer, and dataset is crucial for getting the best results. Each model has its strengths and weaknesses, so it's important to choose the one that fits best with different applications.

### **2.4 Object Detection Algorithm**

#### **2.4.1 Review of Object Detection Technology**

Object detection algorithms are important because they help find and figure out where things are in pictures or videos. They've gotten a lot better over time, and now some fancy ones like Faster R-CNN, Mask R-CNN, YOLO, and RetinaNet which are the best [15][16]. These algorithms work by drawing boxes around objects to show where they are, and they can find lots of different things, which makes them useful for all sorts of stuff [15].

#### **2.4.2 Region-based Convolutional Neural Network**

Region-based Convolutional Neural Network (R-CNN) is an important model for finding things in pictures. It uses fancy CNN to suggest where objects might be and to figure out what they are [17][18]. It works by breaking the picture into pieces and then suggesting where objects might be using a method called Selective Search. These suggestions are then checked by the CNN to see if they're objects, and the model also fixes any mistakes in where it thinks the objects are [18][19].

One of the cool things about R-CNN is that it's good at finding objects accurately, even when they're different sizes or shapes. It can also be changed to do other tasks like figuring out exactly where objects are or following them in a video [19].

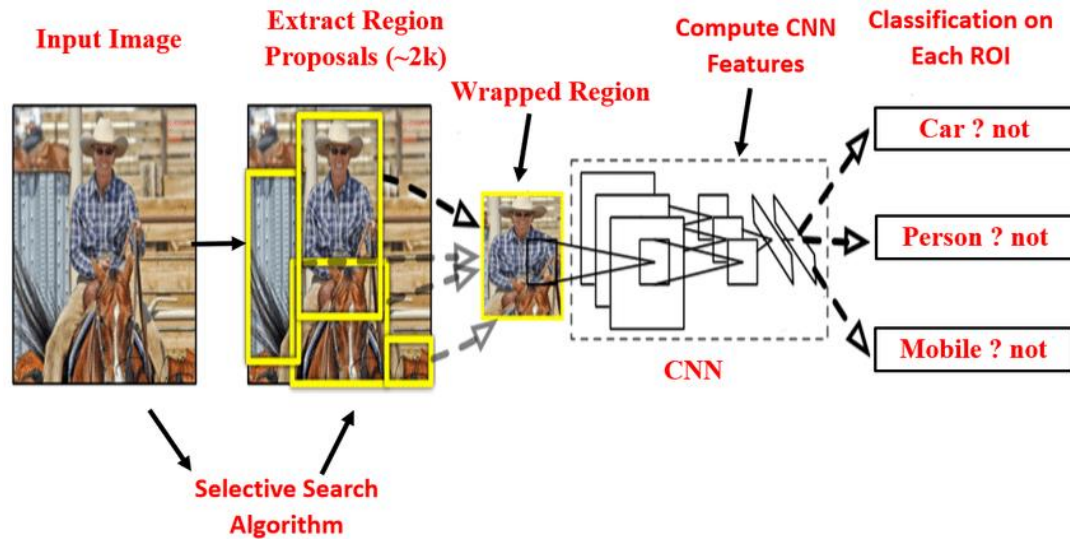


Figure 2.2 Working Principle of R-CNN

Figure 2.2 shows the working principle of R-CNN. First, it will use a Selective Search Algorithm to propose the location of objects. Then, it will pass the region to CNN for image processing and prediction.

### 2.4.3 Fast R-CNN

Fast R-CNN is a big step forward in finding things in pictures, building on the earlier R-CNN model. It's made to be faster and more accurate at figuring out where objects are in images [20][21]. Unlike R-CNN, Fast R-CNN solves the problem of being slow by adding a Region Proposal Network (RPN) that suggests where objects might be directed to the detection part of the model [22].

Fast R-CNN works in two steps: first, it suggests where objects might be in the picture, and then it figures out what they are. In the first step, the RPN suggests regions, which are then checked to see if they're objects. Then the features of these regions are used to decide what the objects are using something like a support vector machine (SVM) [22]. By adding the RPN, Fast R-CNN makes the process of suggesting regions faster and the whole model more efficient compared to before [22].

## CHAPTER 2

One of the cool things about Fast R-CNN is that it can handle finding objects in pictures efficiently. It's good at sharing work between different parts of the picture, which makes it faster and better at figuring out what objects are [22]. This makes Fast R-CNN useful for lots of things like self-driving cars, security cameras, recognizing faces, and more [22].

In summary, Fast R-CNN is a big improvement in finding objects in pictures by making it faster with the help of the Region Proposal Network. This makes it quicker and better at finding and figuring out objects in images, which is useful for lots of computer vision tasks [20][22].

### 2.4.4 Faster R-CNN

Faster R-CNN is a big step forward in finding things in pictures. It's based on earlier models like R-CNN and Fast R-CNN, but it's much better. It came out in 2015 and makes the process of finding things faster and more accurate by adding something called a Region Proposal Network (RPN) right into the model [22][20]. This makes it faster and more accurate because it doesn't need to do a separate step to suggest where objects might be.

The main difference between Faster R-CNN and Fast R-CNN is how they find these suggestions. In Fast R-CNN, it does this before checking with the model. But in Faster R-CNN, it's all part of the same process, so it's faster and better at finding objects in pictures [21].

Faster R-CNN works by first looking at the picture and making a map of what it sees. Then, it uses this map to suggest where objects might be. These suggestions are then checked to see if they're objects, and the model also figures out exactly where they are. This all happens in one go, which makes it much quicker and more accurate [22][20].

In conclusion, Faster R-CNN is a great tool for finding things in pictures because it combines everything into one process, making it faster and more accurate. It's really useful for lots of different things in computer vision [22][20].

### 2.4.5 YOLO

YOLO, which stands for You Only Look Once, is a famous model for finding things in pictures. It's known for being fast and accurate at spotting objects in images. Created by Joseph Redmon, Ali Farhadi, and Santosh Divvala, YOLO changed how object detection is done. Unlike older models like R-CNN which need to do separate steps to find objects, YOLO does everything in one go with a single pass of CNN [23][24].

One thing about YOLO is that it finds objects in just one step, making it super-fast and accurate. It divides the picture into a grid and each part of the grid is responsible for finding objects within it. YOLO predicts boxes around objects and how sure it is that those boxes are correct, which makes the process quicker and more efficient [25].

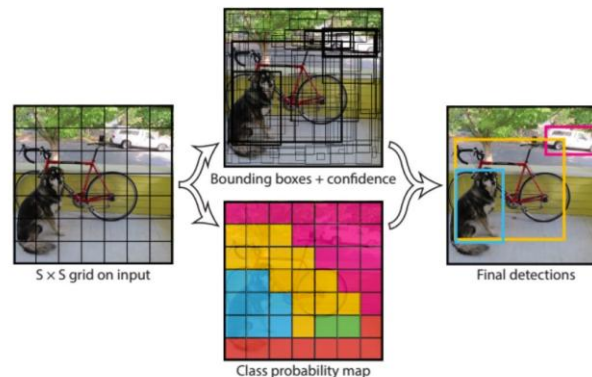


Figure 2.3 Bounding Box Generation and Prediction

Figure 2.3 shows the generation of bounding boxes. YOLO will divide the image into  $S \times S$  grid and predict several bounding boxes including their confidence and class probabilities in each grid cell.

YOLO's setup involves resizing the picture and then passing it through a series of layers that help it understand what's in the picture. It also uses techniques like batch normalization and dropout to make sure it learns properly without getting too good at just one thing. YOLO also uses different-sized boxes to match different objects, making it better at finding different types of things [24][26].

## CHAPTER 2

Over time, YOLO has gotten even better with newer versions like YOLOv8, which improve accuracy, speed, and how much the model can learn. These versions use new techniques like Feature Pyramid Networks and better ways of figuring out what size boxes to use. This makes YOLO a top choice for finding objects in pictures quickly and accurately [24][26].

In short, YOLO is an amazing model for finding objects in pictures. It's fast and accurate because it does everything in one step, using smart techniques like grid-based processing and different-sized boxes. This makes it perfect for lots of different computer vision tasks that need to find and understand objects in images [23][25].

### 2.4.6 SSD

The Single Shot Detector (SSD) is a top-notch model for finding objects in pictures, known for being both quick and accurate. SSD works by using a two-part network: one part looks at the picture and the other part figures out what's in it [27][28]. The first part of the network usually uses a fancy computer program called VGG16 that's been trained to understand pictures. Then, the second part of the network adds some more layers to help find objects in the picture [27].

SSD introduced a new thing called anchor boxes. These are like pre-set boxes that help figure out where objects might be in the picture. They come in different sizes and shapes to match different objects, which helps SSD find things of all sizes [27][28]. By using these anchor boxes, SSD can quickly find objects and figure out what they are.

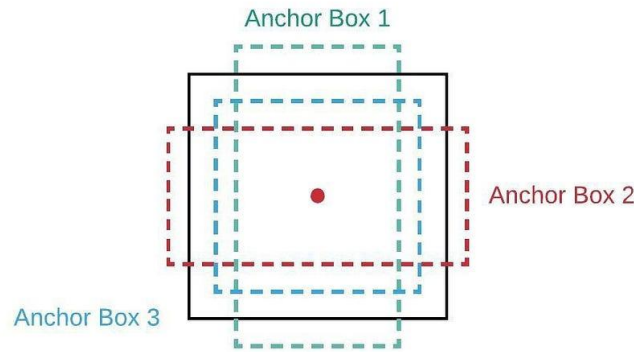


Figure 2.4 Example of Anchor Boxes

Figure 2.4 shows the anchor boxes generated by SSD. SSD will divide the image using a grid and have each grid cell be responsible for detecting objects in that region of the image. In each grid, it will generate several anchor boxes with different shapes and sizes. In this figure, three anchor boxes are generated.

Here's how SSD works: first, it looks at the picture and makes a map of what's in it. Then, it uses these anchor boxes to guess where objects might be. After that, it checks each guess to see if it's an object and what kind of object it is. Finally, it cleans up any mistakes by getting rid of overlapping guesses, and giving a final list of objects and what they are [28].

Overall, SSD is great at finding objects quickly and accurately. They're useful for lots of things like self-driving cars, security cameras, and robots. Their simple approach to finding objects makes them handy for all sorts of computer vision tasks that need to find and understand things in pictures [28].

### 2.4.7 Mask R-CNN

Mask R-CNN is a model for finding and outlining objects in pictures. It's good at both finding objects and figuring out their exact boundaries, a process called instance segmentation [29][30]. It builds upon the Faster R-CNN model by adding a special "mask head" branch that helps outline objects precisely [29][30].



What makes Mask R-CNN special is its ability to not only find objects but also to outline them pixel by pixel. It does this by adding an extra branch to the model that works alongside the regular parts of the model. This branch predicts masks for each proposed region, giving detailed outlines for accurate object localization and classification [30].

To make sure the outlines match the objects perfectly, Mask R-CNN uses something called ROIAlign. This helps preserve the exact locations of features, ensuring precise segmentation, especially for small objects. Additionally, Mask R-CNN separates mask and class prediction, making sure each class gets its mask without any mix-up [30].

Mask R-CNN has lots of advantages. It's easy to train, works well for different tasks, and doesn't slow down Faster R-CNN too much. It's also flexible and can be used for tasks beyond just finding objects and outlining them. Its ability to precisely outline objects makes it great for tasks where accuracy is crucial in computer vision [31].

### 2.4.8 Performance of Object Detection Algorithm

Different architectures of object detection models will cause different performances. Their performances can be shown in several attributes. Here are some comparisons of the model:

1. Accuracy:
  - Studies show that Faster R-CNN tends to have lower loss values compared to Mask R-CNN, suggesting it might be more accurate in certain situations [32].
  - Mask R-CNN is known for being good at lots of tasks and often outperforms other models, showing high levels of accuracy [31].
2. Training Speed:
  - YOLOv3 does pretty well in training, but it might struggle with objects of different sizes and shapes compared to Faster R-CNN, which handles these variations better [33].

- Faster R-CNN is great because it's good at dealing with changes in objects and trains efficiently, making it a strong choice for fast and robust training [33].

### 3. Efficiency:

- Mask R-CNN is praised for being easy to train, performing well across tasks, and not slowing down Faster R-CNN much. It's also flexible and can be used for tasks beyond just finding objects and outlining them [31].
- SSD is efficient in finding objects because it uses special boxes and a simple network, making it useful for quick tasks that need precise object location [34].

In conclusion, each model has its strengths. Mask R-CNN is accurate and easy to train, while Faster R-CNN is good at handling different-sized objects efficiently. YOLOv3 is competitive but might have trouble with certain types of objects.

## 2.5 Object Measurement Project

Object measurement projects employing object detection algorithms have made significant strides, especially with the adoption of deep learning methods. A thorough examination of recent literature on object detection underscores the crucial role of deep learning in this area [35]. Object detection involves identifying objects in images, along with determining their location and classification, making it essential for vision-based software systems.

In the field of machine learning for moving object detection, security applications are particularly important. While previous studies have explored various techniques for detecting and tracking objects in video surveillance, there is still a need for comprehensive literature reviews on specific topics within this domain [36]. For example, research has investigated methods like background subtraction for spotting moving objects and comparing 3D interest point descriptors for baggage inspection at airports [36]. However, many of these studies have narrower focuses compared to systematic literature reviews covering a broader range of relevant research papers in machine learning for object detection security [36].

## CHAPTER 2

Furthermore, extensive research has been conducted on the challenges associated with detecting and tracking objects in dynamic environments. Real-time computing tasks in video surveillance systems are crucial for enhancing security measures by accurately identifying and tracking objects such as humans, cars, and potential threats [36].

Besides, recent advancements in computer vision have significantly enhanced dimensional measurement techniques within industrial and manufacturing settings. One notable application is the use of computer vision for monitoring pellet moisture in iron ore processing. Zhou and Liu [37] developed a grey-scale imaging system that correlates image intensity with moisture levels in iron ore green pellets. Although their study focused primarily on moisture detection, the researchers observed a clear association between intensity variations and changes in pellet size during drying. The system employed polynomial regression on average intensity values, achieving rapid processing speeds of approximately 0.361 seconds per image. This indicates promising potential for real-time monitoring applications in industrial environments.

Beyond moisture detection, computer vision techniques have shown substantial utility in measuring physical dimensions such as length and diameter in industrial components. A 2023 study introduced a metrology approach based on Harris-corner detection to evaluate the dimensions of screws, bolts, and rectangular metal pieces using smartphone cameras [38]. By detecting corners through pixel intensity variations and employing RANSAC (Random Sample Consensus) for line fitting, the method achieved 99% agreement with conventional Vernier calliper measurements. The low-cost implementation and real-time processing capabilities reduce the need for skilled manual labour, thereby improving quality control efficiency.

Similarly, another 2023 study applied deep learning and signal processing techniques to measure the lay length in metallic wire ropes [39]. This system used Mask R-CNN for segmenting individual strands of the rope into Voronoi-like patterns to suppress image noise. Phase correlation analysis was then performed on these segments to evaluate the spatial periodicity, which allowed accurate lay length computation. This approach significantly outperformed manual methods, achieving an average error of just 1.0672 mm and a processing time of only 0.1045 seconds.

## CHAPTER 2

Across these studies, several common technological themes emerge. Robust edge detection techniques, such as Harris-corner detection [38] and improved Canny edge detectors [40], form the foundation of many measurement methods. These are often paired with model fitting algorithms like RANSAC [38] or RHT-LSM (Randomized Hough Transform with Least Squares Matching) [40], which enhance accuracy in identifying geometric features. Additionally, there is an increasing emphasis on hardware optimization; for instance, the use of smartphone cameras [38] and grey-scale imaging systems [37], [39] lowers the barrier to adoption in industrial contexts. The integration of deep learning frameworks, such as Mask R-CNN in wire rope inspection [39], exemplifies the field's shift towards more sophisticated and high-accuracy solutions.

These advancements collectively highlight the growing maturity of computer vision as a reliable tool for dimensional analysis in manufacturing. The convergence of classical image processing, model fitting techniques, and modern deep learning models enables not only improved accuracy and processing speed but also reduced reliance on costly hardware or manual labour.

## CHAPTER 3

### System Methodology/Approach

This chapter will give an overview of the whole project, including the software and hardware used, the methodology and approach of the project.

#### 3.1 System Requirements

##### 3.1.1 Hardware

The hardware involved in this project is the computer and the Raspberry Pi. A computer was issued for mask RCNN set-up, data labelling, model training, program design and prediction. The model of Raspberry Pi is Raspberry Pi 4 Model B 8GB, and it is mainly used for convenience used for getting the input at the measuring station.

Description	Specifications
Motherboard	Asus Prime B760M-K D4
Processor	Intel Core i5-12400F
Operating System	Windows 11
Graphic	NVIDIA GeForce RTX 4060 Ti
Memory	2 * 16GB DDR4 RAM
Storage	1TB SSD

Table 3.1 Specifications of the Computer

### 3.2 Mask-RCNN Model Design

The model design in this project will be separated into five parts, which are data acquisition, data preprocessing, model setup and model training.

In this project, a dataset is required to train the model. The number and the standard deviation of the dataset will affect the performance of the model. To achieve the outcome, the dataset will be collected by capturing the image of fish pellets. Images with different numbers of fish pellets in different positions will be a preference for this project.

Next, data preprocessing will be applied to the datasets. In the first step of this process, the VGG Image Annotator (VIA) will be used to label all the fish pellets in the image and produce a JSON file. The JSON file will include all the information about the position of fish pellets in each image file. After that, a piece of Python code is needed to execute the result in the JSON file.

The next thing is to build a model that can achieve the goals. First and foremost, choosing a model is essential for the project. After reviewing the models for object detection, Mask R-CNN with ResNet architecture should be the best model for this project. After this step, the training process will be started by fitting the training and validation data into this model. The training process might take a lot of time. The training parameters, batch size and epoch will initially be set as 2 and 100, respectively. After this process, the model will be exported.

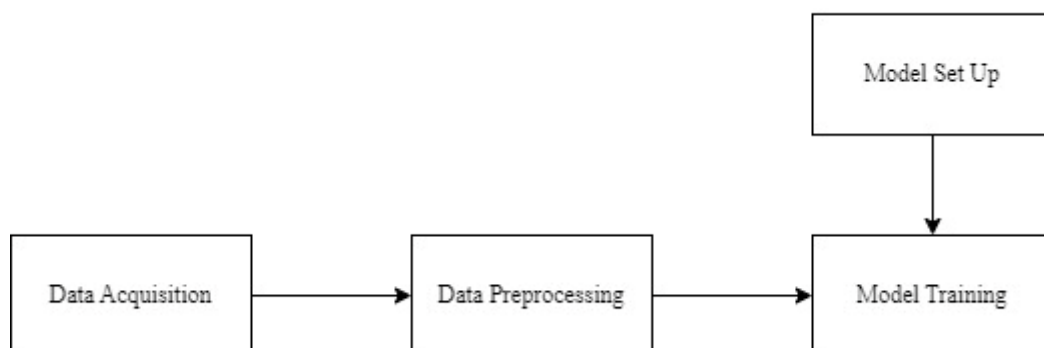


Figure 3.1 Model Training and Set Up

Figure 3.1 shows the steps to train and set up the model. In summary, gathering the training dataset and preprocessing it is necessary. Simultaneously, constructing the model and fitting the dataset for training should be undertaken. The output for this process should be a trained model with expected accuracy.

### 3.3 UI and Backend Design

After the Mask-RCNN is trained, a UI is needed to make the user experience convenient for performing every operation, including taking pictures, selecting the ratio determination algorithm, etc. The backend used in this project is the Flask framework, which can be written in Python, and it will communicate with HTML and JavaScript.

After the user selects all the input and presses the predict button, the image and ratio measurement method will be passed to the Flask backend, and it will be transmitted to the PC for prediction, measurement, and visualization.

### 3.4 Ratio Determination Algorithm Design

When the image and ratio measurement method are sent to the PC, it will first calculate the ratio between the real length and the pixel. Therefore, four algorithms will be designed to calculate it, and the methods are shown below:

- **ArUco Maker:** ArUco markers are 2D binary patterns designed for rapid detection by computer vision systems. It can be applied in this project as there is a built-in library for the detection. It will return the number of pixels(length) of the ArUco maker.
- **Object Matching:** This method uses a FLANN-based matcher, which is also built-in into the library in Python. It will require users to input a template image and its real shortest side length, and then match the real thing in the frame by detecting its key points and descriptors. It will return the number of pixels(length) of the shortest side.
- **Height of Camera:** This method will require users to input the height of the camera and use the algorithm below to calculate the height of the frame.

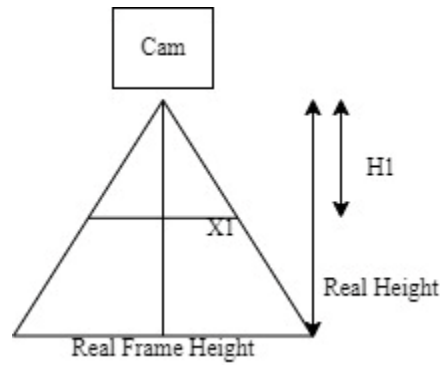


Figure 3.2 Relationship Between Height and Width

$$Real\ Frame\ Height = \frac{X_1}{H_1} \times Real\ Height$$

From the equation, the  $X_1$  and  $H_1$  are tested before and fixed; the users only need to input their real height to know the frame height.

- Height of the Frame: This method requires users to input the height of the frame, which is also called the length of the frame.

All these methods will return the pixel value. By using the equation below, the ratio between the real length and the pixel can be determined:

$$Ratio = \frac{Real\ Length}{Pixel}$$

After determining the real ratio, the image will be processed by the Mask-RCNN to predict the boundaries of the fish pellets. After this, algorithms to calculate the diameter or height and length will be applied to predicted boundaries and work with the ratio calculated to determine the value in cm or mm.

### 3.5 Diameter Algorithm Design

While the shape of every fish pellet might not be the same, it will lead to different heights and widths when the degree of rotation of the fish pellet is changed. Therefore, this algorithm is designed to determine the diameter of the fish pellet. The approaches below are the algorithms to estimate the diameter of fish pellets by using the area or boundary predicted from the Mask-RCNN.



### 3.5.1 Circle Estimation

In this approach, the fish pellets will be treated as a circle and circle estimation is applied to them. In practice, the area in pixels can be calculated by the predicted result and substituted in the equation below to get the diameter:

$$Diameter = 2r = 2 \times \sqrt{\frac{Area}{\pi}}$$

### 3.5.2 Ellipse Estimation

To improve the accuracy, as the fish pellet is not a perfect circle, the ellipse estimation can be applied to the algorithm. The flow of the ellipse estimation is:

1. First, the contour (area) of the fish pellets will be passed to the prebuilt function in OpenCV to find the best-fitting ellipse and return the major axis and minor axis.
2. Calculate the estimated diameter by using the major axis and minor axis.

$$Diameter = \frac{Major\ axis + Minor\ axis}{2}$$

### 3.5.3 Convex Hull Approach

However, if the shape is extremely irregular and cannot be estimated by the methods above, a convex hull approach will be better to estimate the diameter. The flow of this algorithm is:

1. Find the points of the boundary from the contour.
2. Calculate the maximum Euclidean distance between every pair of points and select the maximum value.

### 3.5.4 Average Diameter Approach

As some of the fish pellets are rectangular or other shapes, but they are not extremely irregular, the three methods above might not be suitable for estimating the diameter. In general, the average diameter approach is designed to overcome this shape. The flow of the algorithm is:

1. Find the center point of the boundary.
2. Find the points of the boundary from the contour.

3. Calculate the distance of the center to every point and take the average distance as the radius, and multiply by 2 to get the diameter.

### 3.5.5 Discussion of Diameter Algorithm

From the four methods listed above, a comparison of the four methods can be made below:

Method	Pros	Cons	Best for
Circle Estimation	Simple, fast, and gives a rough approximation	Overestimates for elongated shapes	General circular objects
Ellipse Estimation	Accounts for major and minor axis	Slightly sensitive to noise	Elongated and oval shapes
Convex Hull Approach	Works well for irregular shapes	Overestimates for rounded edges	Concave and irregular shapes
Average Diameter Approach	More robust for slight irregularities	A bit more computationally expensive	Balanced method for all shapes

Table 3.2 Comparison Between Each Diameter Estimation Method

The table above shows the advantages and disadvantages of each method, and the best case to use them. Therefore, the determination of the most suitable method will depend on the shape of the fish pellet.

### 3.5.6 Determination Algorithm

As the shape of fish pellets will be different, to achieve a more accurate result, an algorithm will be designed to determine which approach should be used for different fish pellets.

To determine if the fish pellet is a circle, a circularity equation can be applied to the fish pellet:

$$Circularity = \frac{4\pi A}{P^2}$$

in which A is the area and P is the shape's perimeter. The valid range of circularity is from 0 to 1. The shape is almost a perfect circle if the circularity is close to 1. Therefore, the threshold will be set at 0.85 as the shape is nearly a circle when the circularity is larger than 0.85.

If the shape is not a circle, then the algorithm will check if the shape is most likely an ellipse with the following equation:

$$\text{Aspect Ratio} = \frac{\text{Major Axis Length}}{\text{Minor Axis Length}}$$

The valid range of aspect ratio is larger than or equal to 1. When the aspect ratio is larger than 1.2, the shape is strongly elongated, more like an ellipse or rectangle.

If these two conditions are not met, the algorithm will automatically calculate the diameter by using the average diameter method.

As the Convex Hull Approach is mostly for extremely irregular shapes, it will not be used in the current project.

In conclusion, if the circularity is larger than 0.85, the circle estimation will be used. If it is not a circle, and if the aspect ratio is greater than 1.2, ellipse estimation will be used. Otherwise, the average diameter method will be used.

After the diameter is calculated, the bounding box and mask will be drawn on the image. Moreover, the colour of the mask and bounding box will also be determined as some specific range will be set. When the height and width/diameter of the detected fish pellets are within the range, the colour of the bounding box and mask will be green, otherwise, it will be red.

ID	SKU 79010	SKU 79030	SKU 39070
Diameter	3.0-3.6mm	6.8-8.2mm	12.5-13.5mm
Length	3.0-3.6mm	6.8-8.2mm	12.5-13.5mm

Table 3.3 Specifications of Fish Pellets

After drawing the necessary item on the image, it will send the resulting image to the Raspberry Pi through MQTT. Once the Raspberry Pi receives the image, it will post the image on the webpage using Flask.

### 3.6 Project Timeline

This project has taken about 12 weeks to complete. The timeline is shown below:



Figure 3.3 Project Timeline for FYP 1

This is the Gantt Chart for FYP 1. The first week is spent doing data acquisition. From week two until week four, the data labelling is processed. After that, the model setup is done in week five. Model training and algorithm development started in week six and ended in week six and week seven, respectively. Then, GUI development is done in week eight, while hardware setup begins from week nine and ends in week ten. The prototype setup is done within week ten. Lastly, the final improvement is processed from week eleven to week twelve.

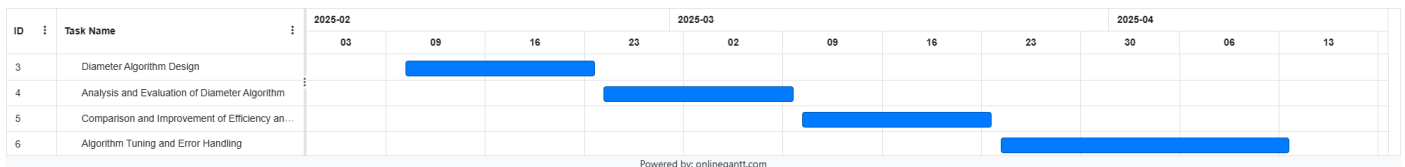


Figure 3.4 Project Timeline for FYP 2

This is the Gantt Chart for FYP2. In the first and second weeks, the diameter algorithm will be designed. In the third and fourth weeks, an analysis and evaluation will be done on diameter algorithms. In weeks five and six, the comparison and improvement of efficiency and stability will be made. From weeks seven to nine, tuning will be applied to the algorithms and testing for errors.

## CHAPTER 4

### System Design

In this chapter, the system design and overall flow diagram will be shown.

#### 4.1 Overall System Design

In the overall system, two devices, which are Raspberry Pi and a personal computer, will be used to build the whole system. Raspberry Pi will mainly be used to capture the image, display the output and show the GUI for the user to interact with. The personal computer will be used to receive the image and process it. After the prediction is done, the result will be extracted, and then the mask and bounding box will be drawn on the image to send back to the Raspberry Pi.

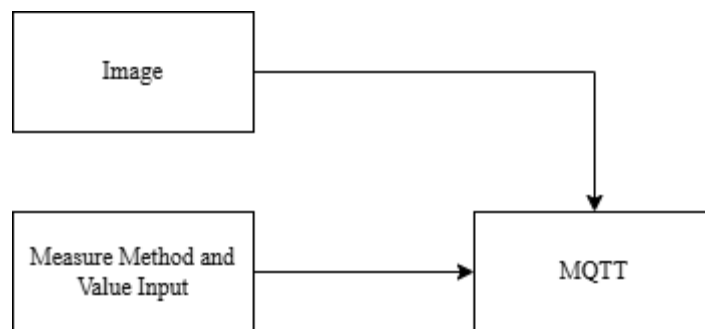


Figure 4.1 Flow Chart of Raspberry Pi Capturing the Input

At first, the user can choose the measurement method and input the value (if needed). Then, the user can capture the image for prediction. The measurement method and input, together with the image, will be passed into the personal computer by using MQTT.

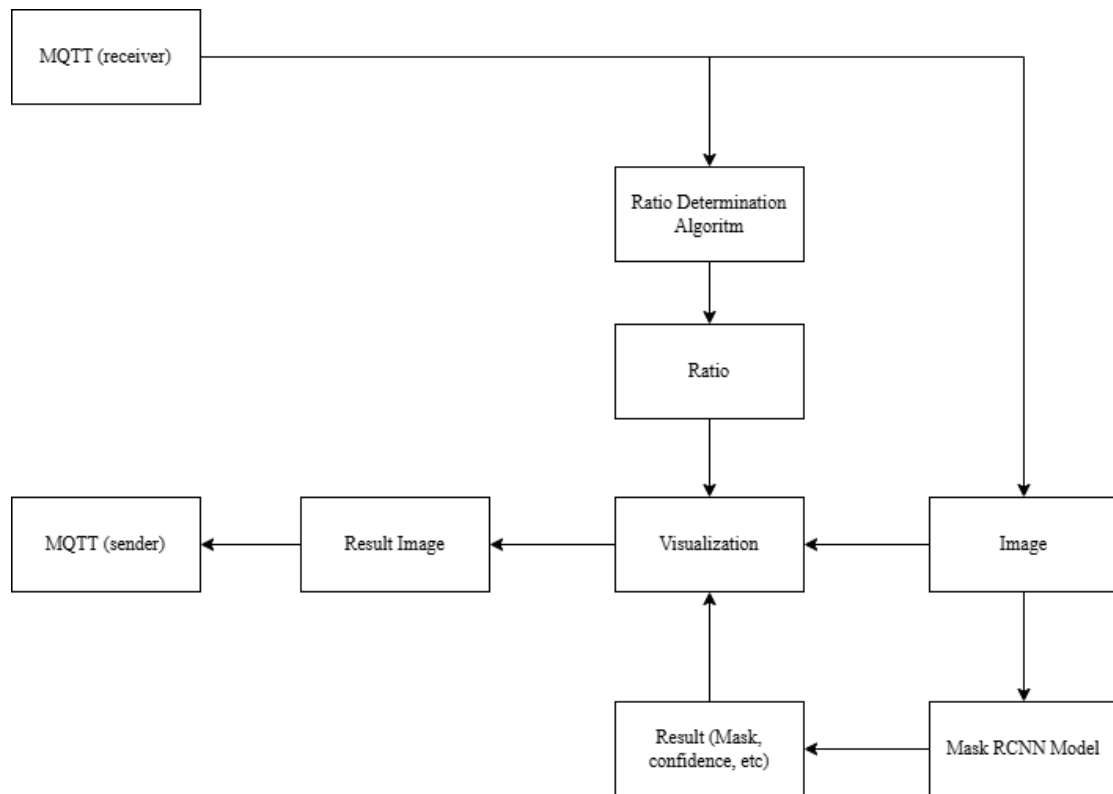


Figure 4.2 Flow Chart of Personal Computer

The MQTT server will pass the image and measurement method with input (if applicable) to the PC. The image will be passed into the Mask RCNN model to obtain information about the image, such as the mask of the fish pellet and the bounding box. The measurement method and input will be passed into the ratio determination algorithm to determine the ratio between the length in pixels and the real length in cm or mm.

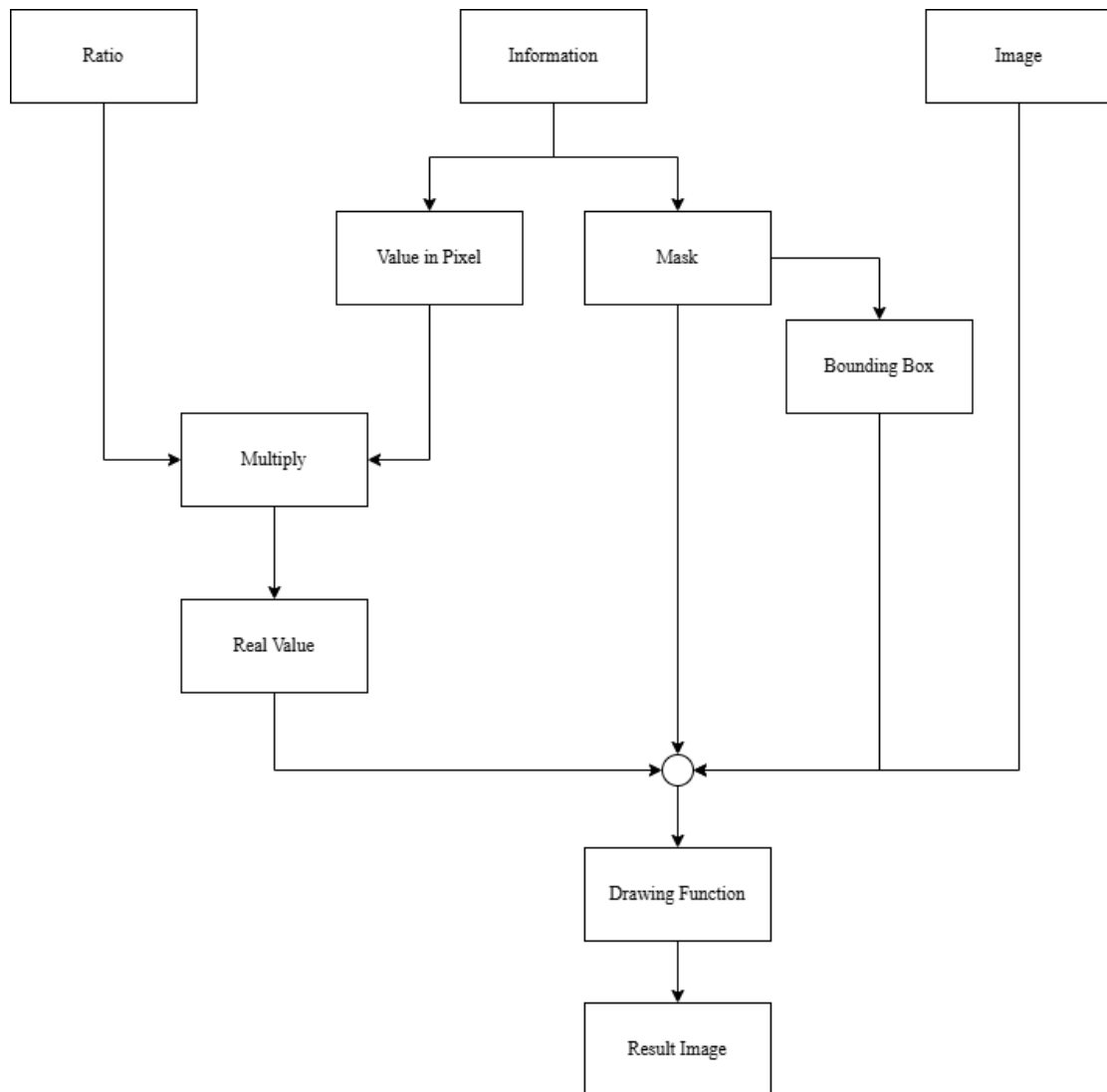


Figure 4.3 Flow Chart of Visualization Algorithm

The ratio, original image and predicted information will be passed into the visualization algorithm to calculate the real length of the fish pellet and then draw the mask, bounding box and label the length on the bounding box. The ratio will first be multiplied by the length in pixels predicted by the model to get the real length in cm (or mm). The predicted mask will be extracted to get the bounding box. The real length, mask and bounding box will be passed into the drawing and labelling function to visualize them on the image. After this process, the resulting image with the mask and bounding box on each fish pellet with the label will come out.

## CHAPTER 4

The resulting image will be passed to the MQTT server and sent back to the Raspberry Pi.



Figure 4.4 Flow Chart of Raspberry Pi Capturing the Input

After receiving the resulting image, Raspberry Pi will post it on the webpage for the user to observe.

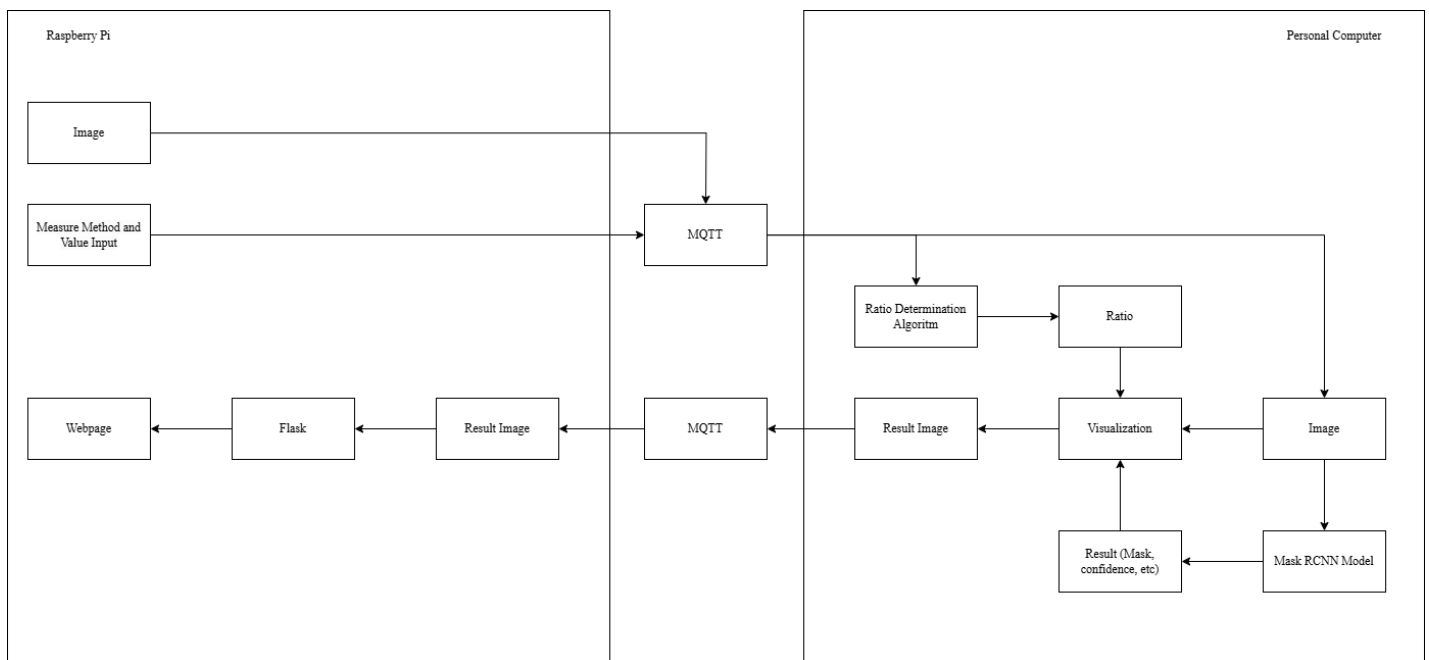


Figure 4.5 Flow Chart of Overall System

Figure 4.5 shows the flowchart of the overall system. In conclusion, it will contain two devices, which are Raspberry Pi to get input and show output and a personal computer to do ratio calculation, fish pellet detection and visualization. These devices will communicate with each other to achieve the purpose of detecting and measuring the fish pellet.



## 4.2 Prototype Design

The overall system will be placed on a platform called “measuring station”, such as a container or a tray, to measure the fish pellet. The setup for the measuring station and the overall system is shown in Figure 4.6 and Figure 4.7.

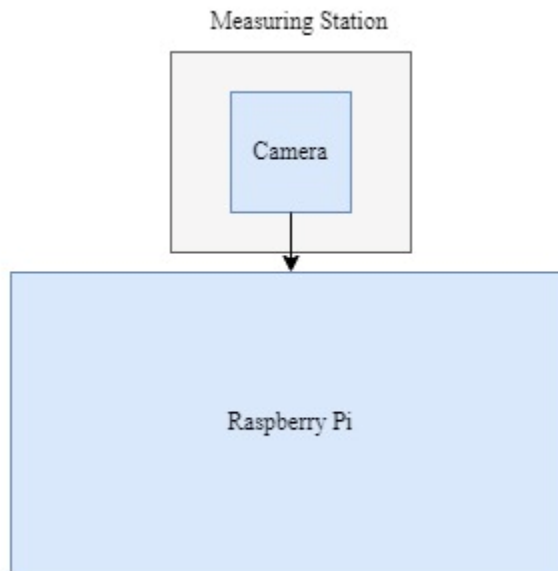


Figure 4.6 Top View of The System

The system from the top side is shown in Figure 4.6. In this diagram, the Raspberry Pi will be the main component in this system, and it is placed beside the measuring station. The camera is interfaced with the Raspberry Pi, and it will be mounted at the top of the measuring station to capture the fish pellet.

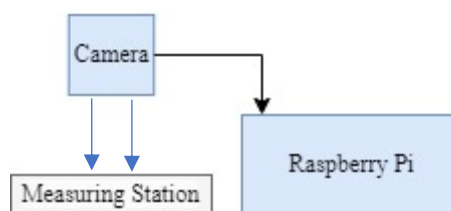


Figure 4.7 Side View of The System

Additionally, the side view of the system is shown in Figure 4.7. This diagram places more emphasis on the distance between the system and the measuring station. From the diagram, the Raspberry Pi is not stuck to the measuring station, and the camera will be

## CHAPTER 4

set up above the measuring station by using something to support it, and the camera will be facing the measuring station.

## CHAPTER 5

### System Implementation

In this chapter, the results of the work will be shown.

#### 5.1 Prototype

This section will show the current prototype, which is also the measuring station. The station is built with some boxes.



Figure 5.1 Raspberry Pi and The Measuring Station



Figure 5.2 Measuring Station



Figure 5.3 Top View of the Measuring Station



Figure 5.4 Camera Angle

These four figures show the setup prototype, users can put the fish pellet inside through the hole in the box. The camera can also be adjusted to fit the frame to the white paper.

### 5.2 UI Design

The UI is designed using HTML as shown below:

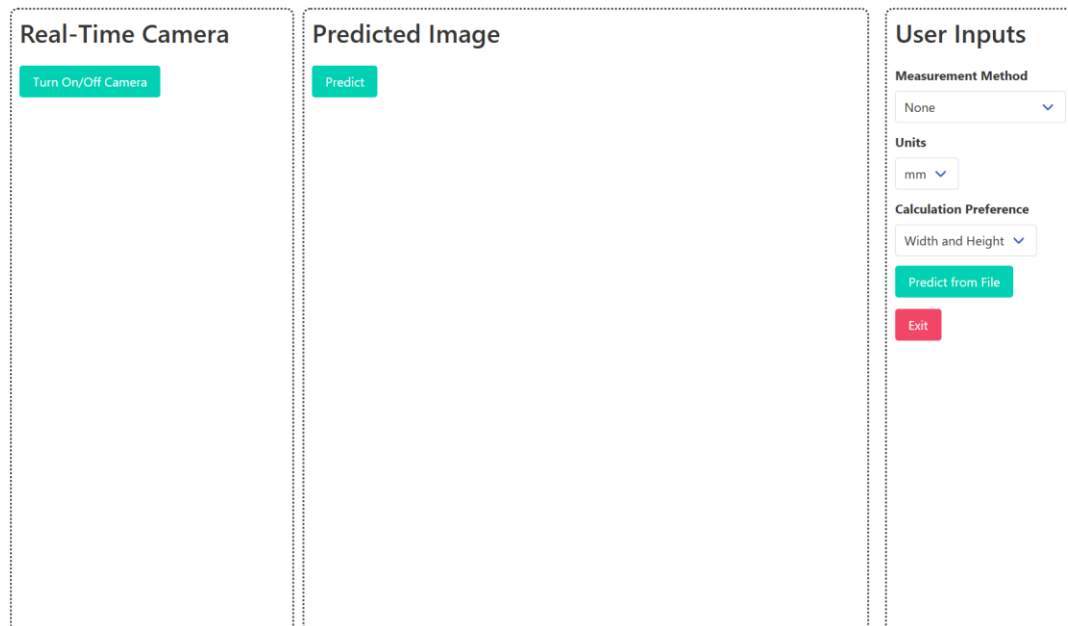


Figure 5.5 Webpage of the Project

In this UI, there are three main columns: the first column is the real-time camera display, the second is the resulting image shown, and the third is the user input, which allows the user to choose the measurement method and calculation preference.

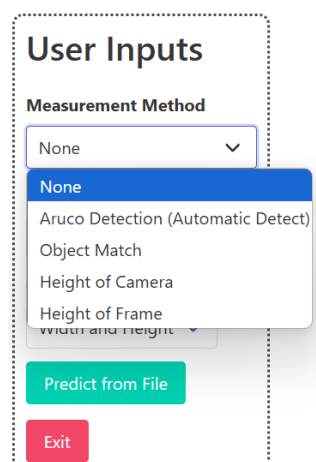
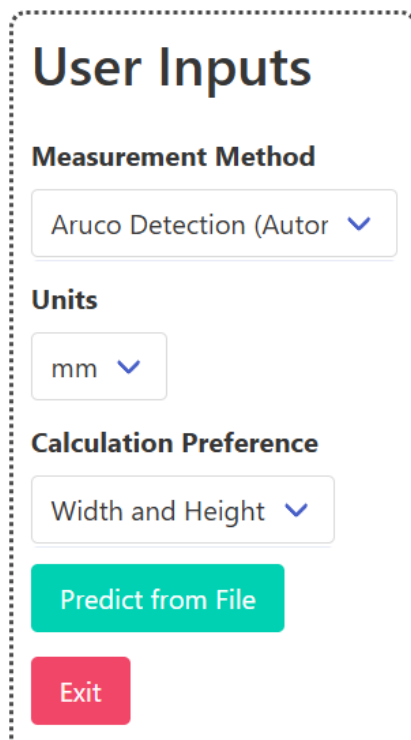


Figure 5.6 Measurement Methods

The users can choose the measurement method in the drop-down list and there are four methods that can be used.



**User Inputs**

**Measurement Method**

Aruco Detection (Autor ▼)

**Units**

mm ▼

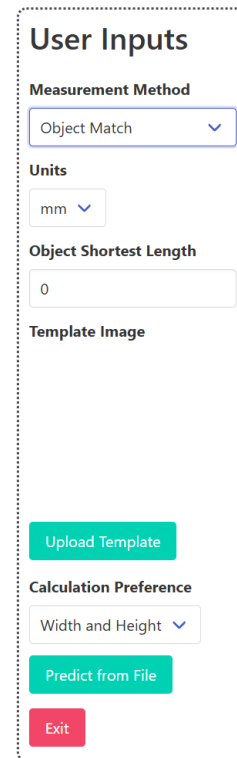
**Calculation Preference**

Width and Height ▼

Predict from File

Exit

Figure 5.7 Aruco Detection



**User Inputs**

**Measurement Method**

Object Match ▼

**Units**

mm ▼

**Object Shortest Length**

0

**Template Image**

Upload Template

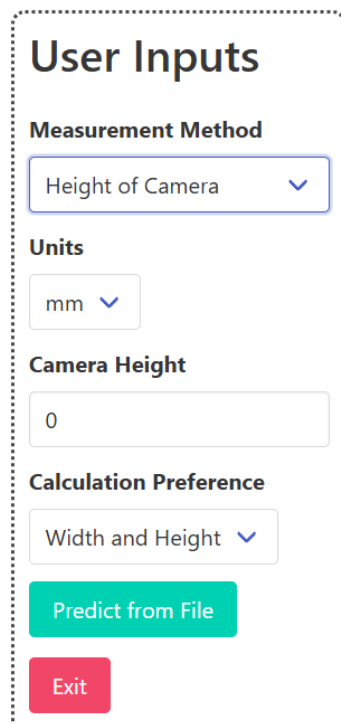
**Calculation Preference**

Width and Height ▼

Predict from File

Exit

Figure 5.8 Object Match



**User Inputs**

**Measurement Method**

Height of Camera ▼

**Units**

mm ▼

**Camera Height**

0

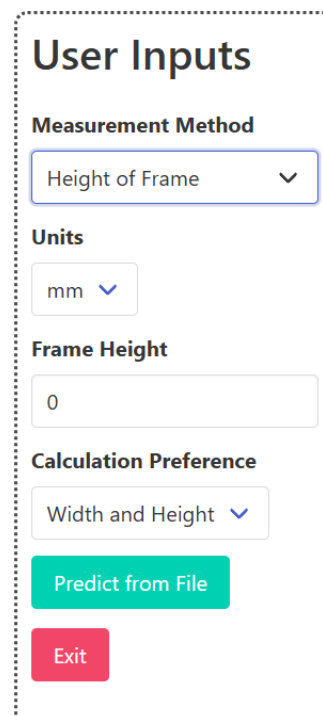
**Calculation Preference**

Width and Height ▼

Predict from File

Exit

Figure 5.9 Height of Camera



**User Inputs**

**Measurement Method**

Height of Frame ▼

**Units**

mm ▼

**Frame Height**

0

**Calculation Preference**

Width and Height ▼

Predict from File

Exit

Figure 5.10 Height of Frame

## CHAPTER 5

The four figures above show that the input required the user to key in. In Figure 5.7, there is no input required as the Aruco Image will be detected and the ratio will be calculated automatically. In Figure 5.8, the object matching method will require the user to input the template image and its shortest side length to determine the ratio. In Figure 5.9 and Figure 5.10, the user just needs to input the height of camera or height of frame, then the ratio will be calculated with the algorithm.

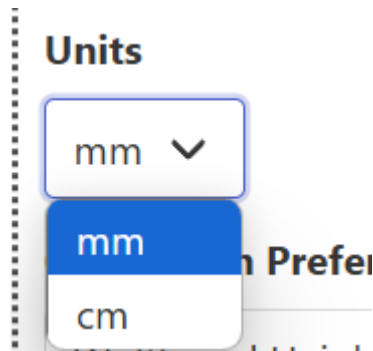


Figure 5.11 Unit Selections

Besides, the UI also has the customization on unit selection, which the user can choose to measure in mm or cm.

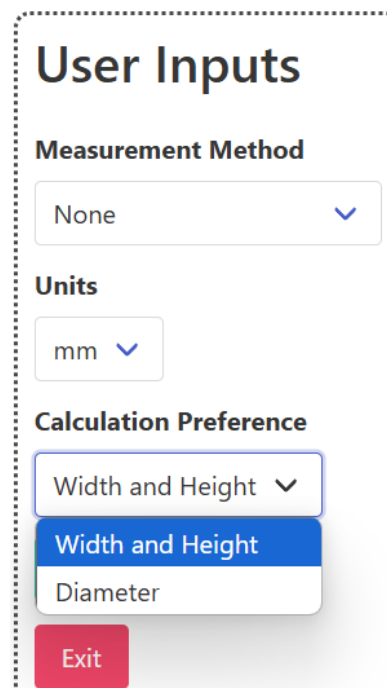


Figure 5.12 Calculation Preference Selections

The UI can also let the user choose to come out with the result in diameter or width and height.

### **5.3 Difficulties and Challenges**

In this project, the main difficulty is the computing power of Raspberry Pi. This Raspberry Pi is used for providing a convincing platform, as it is tiny and cheap. However, running a deep learning detection is a heavy task for a Raspberry Pi, even if it has 8 GB RAM. Hence, another computer is used to overcome this problem, these two devices will need to communicate with each other to exchange data and perform their tasks. Raspberry Pi will act as an input-gathering and GUI-showing device, while the computer will act as the predicting device. By applying this method, the speed of the whole process has significantly increased.

## CHAPTER 6

### System Evaluation and Discussion

In this section, the system evaluation and discussion will be done.

#### 6.1 Result of Mask-RCNN Model

This project has 1200 images for training and 340 images for validation. The final model shows a significant result in the detection.

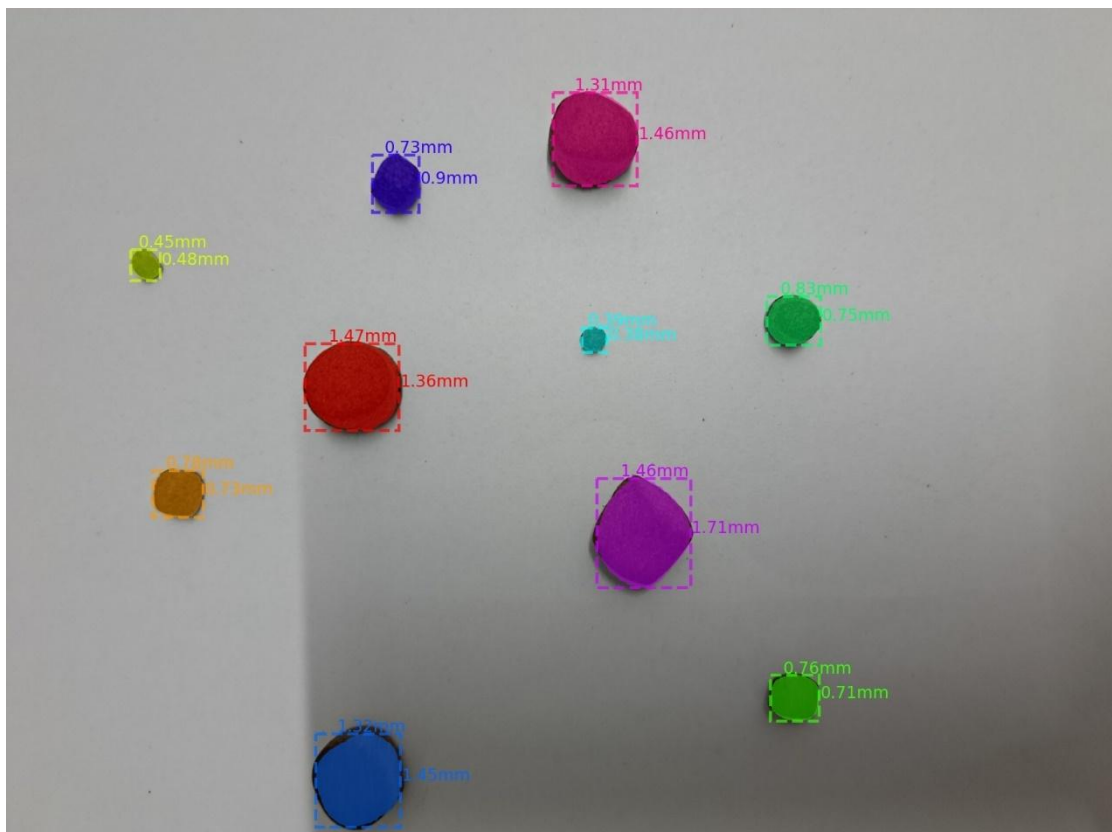


Figure 6.1 Result of The Model

Figure 6.1 shows the mask detected in the image. It will also show the height and width of the fish pellet.



### 6.2 Result of Ratio Determination Algorithm

After applying the four methods, the result will also be shown below:

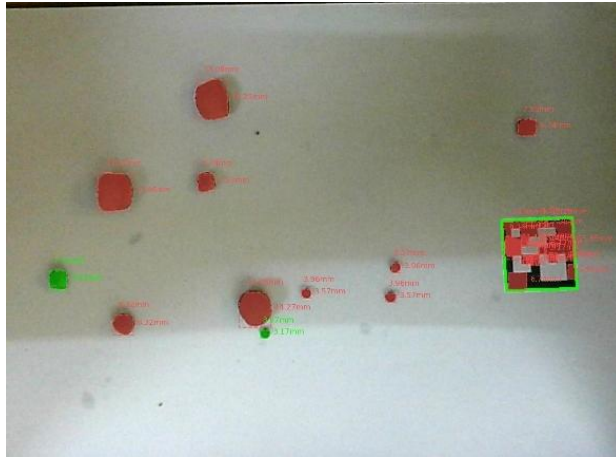


Figure 6.2 Result of Aruco Makers



Figure 6.3 Result of Object Matching Method

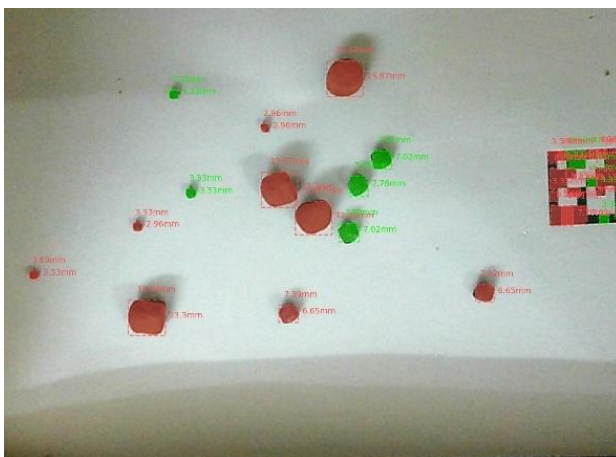


Figure 6.4 Result of Height of Camera Method

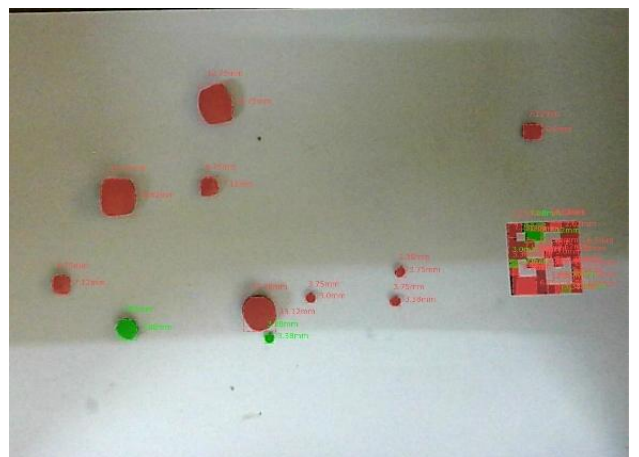


Figure 6.5 Result of Height of Frame Method

The four images above show the result of each method, and the measurement is done in height and width. The accuracy of the object matching method, the height of the camera method and the height of the frame will depend on the user input, while the ArUco maker has already been fixed, and the real size of its length is already keyed in in the system.

### 6.3 Discussion of Height and Width Measurement

Several potential issues will affect the accuracy of the measurement. The detected height and width are dependent on the rotation of the fish pellet, as its shape is irregular.

To compare the height and width calculated by the algorithm and the real height and width, some tools can be used for measuring. First, the figure below is an example of the calculation by the algorithm:



Figure 6.6 Algorithm Calculation



Figure 6.7 Width of Small Fish Pellet



Figure 6.8 Height of Small Fish Pellet



Figure 6.9 Width of Big Fish Pellet



Figure 6.10 Height of Big Fish Pellet

## CHAPTER 6

A table comparing the algorithm calculations and the real height and width can be listed below:

	Algorithm Calculation (mm)	Real Value (mm)	Error
Height (Small Pellet)	4.07	3.98	2.26%
Width (Small Pellet)	3.62	3.62	0%
Height (Big Pellet)	14.48	14.44	0.28%
Width (Big Pellet)	14.03	14.02	0.07%

Table 6.1 Comparison Between Algorithm Calculation and Real Value

From the table above, the error of the calculation by the algorithm is within 3%. The error is calculated by

$$Error = \frac{|Real Value - Algorithm Calculation|}{Real Value} \times 100\%$$

As a result, the error of calculation is acceptable, and when the fish pellets get bigger, the error becomes more stable when the fish pellet are bigger. By comparing the big fish pellets and small fish pellets in the image, a significant difference, which is the pixels representing the fish pellet, has become the main reason to affect the difference in error between big and small fish pellets.



Figure 6.11 Comparison of Pixels Between Big and Small Fish Pellets

In Figure 6.11, the number of pixels for these two fish pellets is shown. As the number of pixels that represent the small fish pellet is relatively low, a small misclassification of one pixel has a higher impact on the total area percentage-wise, meaning that a small fish pellet will have a more unstable error value compared to the bigger fish pellet.

## CHAPTER 6

On the other hand, if the error is calculated by subtracting the Algorithm Calculation from the Real Value, the error will fall within the range of  $\pm 0.1\text{mm}$ , which represents a more stable error for both small and large fish pellets.

However, the reliability of manual measurement will also be affected by several factors:

- i. Degree of rotation: When humans measure the fish pellet manually, it might not have the same degree of rotation as the one measured by the algorithm. As the fish pellet's shape is irregular, there might be some errors in measuring by the algorithm and manually.



Figure 6.12 Different Degrees of Rotation of The Same Pellets

In Figure 6.12, the three fish pellets are the same but in different degrees of rotation, and the three orange rectangles are the same width and height. As a result, the rectangle can fit only one of the fish pellets, indicating that the same fish pellet may have a different width and height in a different degree of rotation.

- ii. Irregular shape: The irregular shape of fish pellets is an important factor, especially for the larger size fish pellets.



Figure 6.13 Method 1 for  
Manual Measurement



Figure 6.14 Method 2 for Manual  
Measurement



Figure 6.15 Result of Method 1



Figure 6.16 Result of Method 2

From the four figures above, the differences between these methods are obvious. The shape of the fish pellet is irregular, when it is measured at different heights, as shown in Figure 6.13 and Figure 6.14, the result will be slightly different.

The result from Table 6.1 has already eliminated these two factors by fixing the degree of rotation and comparing the degree of rotation with the fish pellet in predicted image for overcome the first factor and confirm the manual measurement point by comparing it to the fish pellet's area detected by the AI model for overcome the second factor. However, the precision of humans is limited, so the real result might be slightly different.

#### 6.4 Result and Discussion of Diameter Algorithm

The diameter calculation algorithm is designed to overcome the problem stated in section 6.3, which uses a different calculation to come out with a single diameter value. First, let's separate the four methods and view the result.

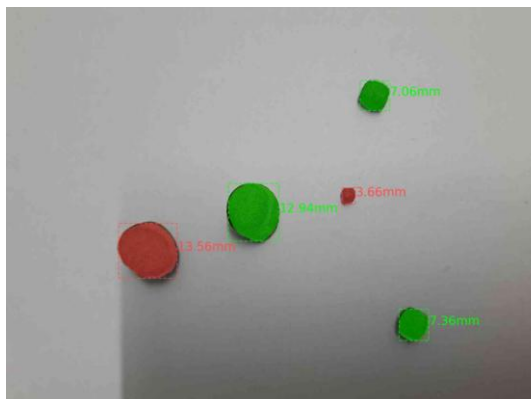


Figure 6.17 Result of Circle Estimation

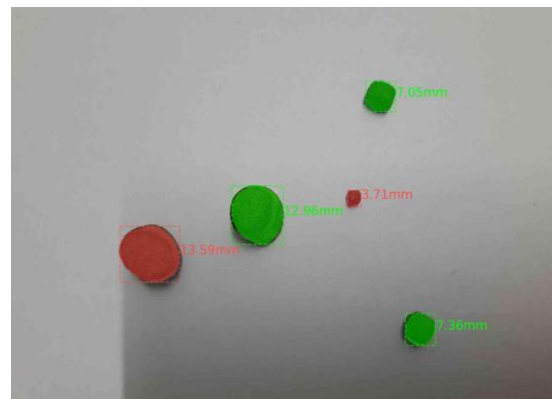


Figure 6.18 Result of Ellipse Estimation



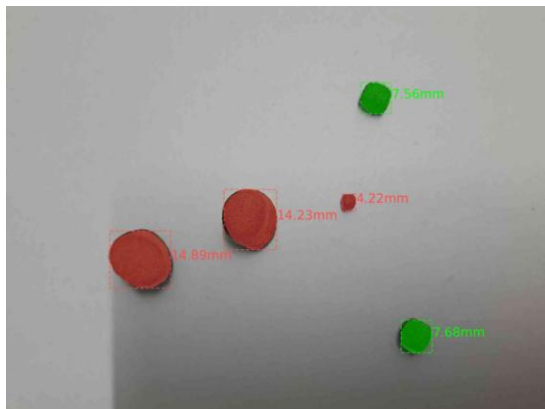


Figure 6.19 Result of Convex Hull  
Approach

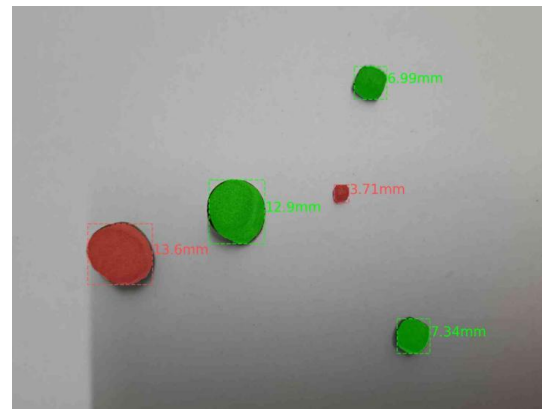


Figure 6.20 Result of Average Diameter  
Approach

To calculate the difference between the two methods, the equation below can be used:

$$Difference(\%) = \frac{|Value\ 1 - Value\ 2|}{\frac{Value\ 1 + Value\ 2}{2}} \times 100\%$$

The equation to calculate the percentage of difference is to divide the difference between the two values by the average of the two values.

Comparison	Difference (%)	Notes
Circle and Ellipse	1.36%	Compare 3.66mm with 3.71mm
Circle and Convex Hull	14.21%	Compare 3.66mm with 4.22mm
Circle and Average Diameter	1.36%	Compare 3.66mm with 3.71mm
Ellipse and Convex Hull	12.86%	Compare 3.71mm with 4.22mm
Ellipse and Average Diameter	0.85%	Compare 7.05mm with 6.99mm
Convex Hull and Average Diameter	12.86%	Compare 4.22mm with 3.71mm

Table 6.2 Comparison Between Each Approach

From the table above, the difference between each approach is shown and indicates the similarity of each approach. As a result, the Convex Hull approach has the most difference with other methods, as the Convex Hull is mainly for extremely irregular shapes. The other three methods, which are circle estimation, ellipse estimation, and average diameter approach, have a smaller difference, indicating that these three algorithms have a similar result.

To achieve a more accurate result, circle estimation, ellipse estimation and average diameter approach should be used together. By using the determination algorithm stated

## CHAPTER 6

in section 3.5.6, which will determine the circularity and aspect ratio and then determine which approach will be used. The result of the mixture of these three approaches is shown below:

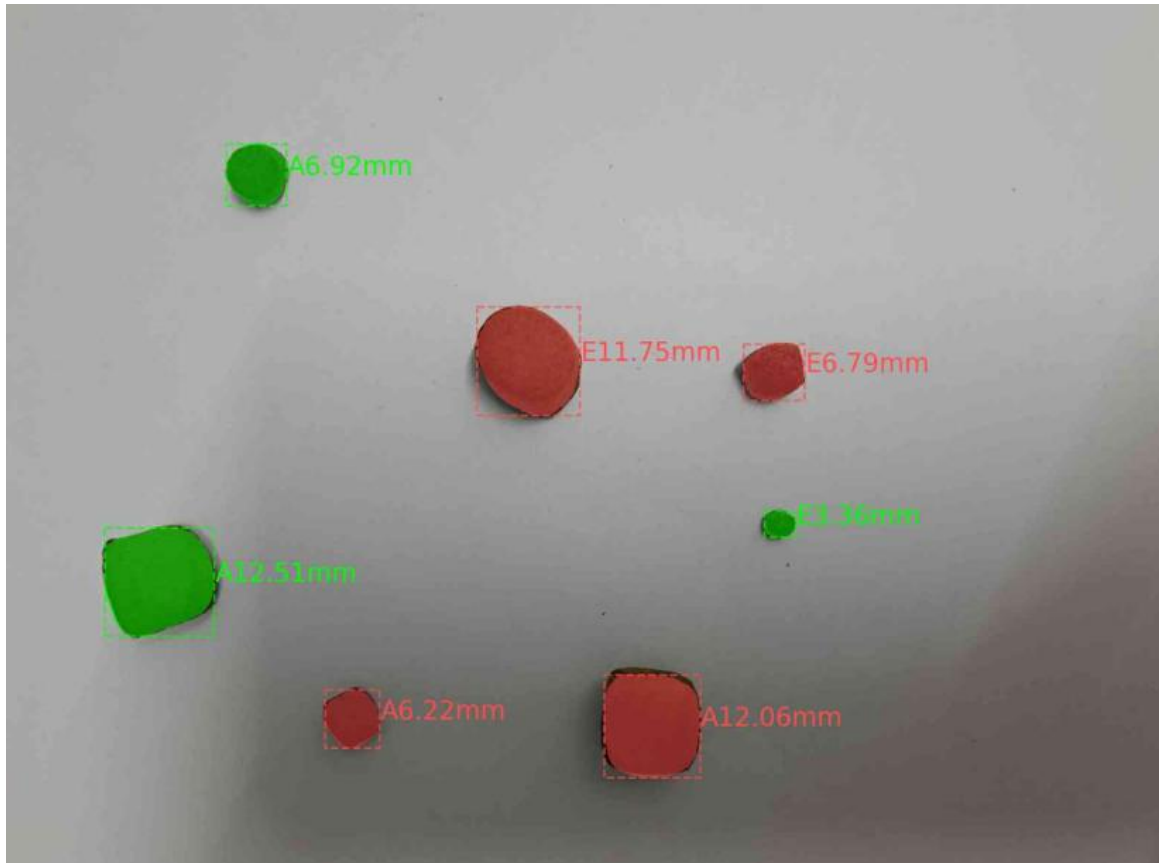


Figure 6.21 Result of Estimating the Shape of Fish Pellet

From Figure 6.21, the “C” labelled at the front of the value indicates that it is estimated as a circle, “E” indicates that it is estimated as an ellipse, and “A” indicates that it is neither a circle nor an ellipse.

As a result, the algorithm has given an efficient shape classification for the fish pellets. The fallen fish pellets, which are likely a rounded rectangle, will be used with the average diameter approach, while the properly aligned fish pellets will be used with the ellipse estimation. In this case, there are no circles estimated as it is hard to fulfil the condition of a circle. Additionally, the threshold value can be adjusted to determine circles and ellipses more accurately or loosely.

### 6.5 Evaluation and Discussion of Time Efficiency

Initially, the ratio determination algorithm, which is the algorithm to calculate the ratio between the real length and the pixel, was processed in the Raspberry Pi. The time to process this algorithm depends on the methods and the computing power of the Raspberry Pi. Therefore, this section is designed to improve the time efficiency of the overall system.

In the new system, the ratio determination is processed in the PC, which will have more computing power than the Raspberry Pi to reduce the time taken. Therefore, the Raspberry Pi will take the role of a lightweight server to host a Flask server for input and output purposes, the overall backend process will be done on the PC for efficiency purposes.

The result is shown below:

	Version 1 (s)	Version 2 (s)	Improvement (%)
General			
Overall Flow	13.67	11.82	13.53
Mask RCNN Detection	3.1	2.6	Not Applicable
Visualization and Length Calculation	0.77	0.46	
Ratio Determination Algorithm			
Aruco Image Method	0.033633	0.009833	70.764
Camera Height Estimation	0.000167	0.000967	-479.042
Frame Height Estimation	0.000667	0.000733	-9.895
Custom Object Matching Method	0.479967	0.0713	85.145

Table 6.3 Comparison Between Two Versions of the System

In version 1, the ratio determination algorithm is implemented on the Raspberry Pi, whereas in version 2, it is implemented on the PC. The results in Table 4.4 represent the average value obtained from running the system three times. The completion time depends on various factors, including the current computing power, other programs running simultaneously, cache memory, and RAM usage, all of which can affect the system's performance.

From Table 6.3, the time of overall system flow has improved by 13.53%, which is from 13.67 seconds to 11.82 seconds. In the meantime, two sections step back, which are camera height estimation and frame height estimation. It might depend on the



current system resource or even the CPU architecture, as the ARM RISC architecture might be faster in processing simple commands than a CISC CPU [37]. However, the step back didn't affect the overall system very much, as the time taken for these two methods is likely as small as  $10^{-4}$  second. It can be ignored as the other processes are much slower compared to these two methods.

The Aruco image and custom object matching method are complex and require the analysis of the image to predict the Aruco on the image and match the custom object. Therefore, they are improved by 70.764% and 85.145% respectively as the computing power of PC is stronger than Raspberry Pi.

However, the time efficiency can also be improved by reducing the time used in data transfer, which reduces the time in MQTT transmission. Initially, the MQTT broker is “test.mosquitto.org”, which will transmit the data over the Internet to the server and transmit back. This process will take some time and can be improved by setting the local device, such as the Raspberry Pi, as the broker. With this approach, the data will not be transmitted to the Internet but only local area network.

	<b>Version 2 (s)</b>	<b>Version 3 (s)</b>	<b>Improvement (%)</b>
Test 1	11.8457	9.365	
Test 2	11.8466	9.3282	
Test 3	11.7813	9.4704	
Average	11.8245	9.3466	20.96

Table 6.4 Comparison of Time Used Between Different Flows of Data Transmission

From Table 6.4, version 3 is the improved version, which transmits the data in the local area network only. The improvement is significant as it improves by 20.96%, while in Table 6.3, it only improves by 13.53%, meaning that the time used in data transmission will be the most critical factor influencing overall system performance. This indicates that minimising external network dependencies can substantially reduce latency and enhance communication reliability.

## 6.6 Objective Evaluation

After the project is done, the objectives should be reviewed and determined if they have been achieved. The objectives are shown below:

## CHAPTER 6

1. To present an improved and efficient method for determining fish pellet diameter to address the current industry issue.
2. To develop a deep learning model to predict the area of the fish pellet.
3. To develop an algorithm to calculate the height and width, or diameter of the fish pellet by extracting the information from the detection.
4. To integrate the computer vision, deep learning model and algorithm with the Raspberry Pi hardware.
5. To develop a user-friendly GUI and measurement station for convenient observation.
6. To validate the performance and accuracy of the measuring system.

For the first objective, it is achieved as this project has demonstrated the use of AI and algorithms to calculate their diameter and solve the accuracy and efficiency problems faced by the industry nowadays.

For the second objective, it can be proved in Section 6.1, as this section has shown the prediction of the area using the Mask-RCNN model, which is an AI model.

For the third objective, it can be achieved in Sections 6.2 and 6.4, as they demonstrated the calculation of height and width and diameter of the fish pellets.

For the fourth objective, it is shown in Sections 5.1, 5.2 and Chapter 6, as they show the result of communication between the Raspberry Pi with the Mask-RCNN model and algorithms.

For the fifth objective, it is shown in Section 5.2, as it shows the UI design.

For the sixth objective, it can be shown in Chapter 6. It shows that the system is evaluated, and the performance and accuracy are improved.

In summary, all the objectives of this project have been successfully achieved. Overall, the project provides a solid foundation for future expansion and real-world industrial applications.

## CHAPTER 7

### Conclusion and Recommendation

#### 7.1 Conclusion

In conclusion, some industries may face difficulties when measuring their product due to accuracy and efficiency problems. They might face low accuracy and low efficiency as they mostly use manual measurement. Therefore, this project aims to overcome the problem by using computer vision and AI methods.

The methods include obtaining and labelling the dataset to train a Mask RCNN model. At the same time, a webpage will be developed to act as the GUI to convince users. The GUI will get the input, and the Raspberry Pi will pass the measurement method and the image to the computer for detection, calculation, and visualization. The resulting image will be passed back to the Raspberry Pi, and it will render it on the webpage by using Flask as a communication tool between Python and JavaScript.

After an initial prototype is done, comparison and analysis of each sector is done to determine the best performance and efficiency of the system. To improve the accuracy, a thorough analysis of results' accuracy is done, and results are obtained with a tuning algorithm to calculate the diameter or height and length of the fish pellets. Besides, the improvement of time efficiency is made by comparing and restructuring the flow to reduce the overall operating time.

As a result, this project has achieved its objectives, which are model training, algorithm development, Raspberry Pi integration and performance evaluation. Optimisation is also applied to the system, and the improvement is obvious. However, there is also significant potential for improvement once it has been tested in an industry setting and feedback has been gathered.

### 7.2 Recommendation

Based on the outcomes of this project, it is recommended that further testing and validation be conducted in real industrial environments to ensure the system's robustness under various working conditions. Collaboration with industry partners would provide valuable insights and feedback that could help in refining the accuracy and efficiency of the system further.

Additionally, it is advisable to expand the dataset with more diverse samples to improve model generalization. Incorporating real-time monitoring features and automatic calibration tools can also enhance usability and reduce manual intervention. Future work may include exploring alternative AI models or lightweight architectures for faster inference, especially when deploying entirely on edge devices like the Raspberry Pi.

Lastly, continuous system updates and user-centred GUI enhancements are recommended to maintain a seamless and intuitive experience, ensuring higher user adoption in the long term.

## REFERENCES

- [1] L. Calatroni, Y. van Gennip, C.-B. Schönlieb, H. Rowland, and A. Flenner, “Graph clustering, variational image segmentation methods and Hough transform scale detection for object measurement in images,” *Journal of Mathematical Imaging and Vision*, vol. 57, no. 2, pp. 269–291, Feb. 2017, doi: <https://doi.org/10.1007/s10851-016-0678-0>.
- [2] X. Zhang, “Traffic Sign Detection Based on YOLO v3,” IEEE Xplore, Jan. 01, 2023. <https://ieeexplore.ieee.org/document/10075795?denied=> (accessed Mar. 09, 2024).
- [3] B. Patel, Deep Kothadiya, and R. Patel, “COVID-19 detection on chest x-ray image using yolo based architecture,” Jan. 2023, doi: <https://doi.org/10.1109/aisc56616.2023.10085498>.
- [4] “Computer Vision: History & How it Works | Sama,” [www.sama.com](http://www.sama.com). <https://www.sama.com/blog/computer-vision-history-how-it-works> (accessed Mar. 09, 2024).
- [5] H. Wang and B. Raj, “On the Origin of Deep Learning On the Origin of Deep Learning,” 2017. Available: <https://arxiv.org/pdf/1702.07800.pdf>
- [6] I. H. Sarker, “Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions,” *SN Computer Science*, vol. 2, no. 6, Aug. 2021, doi: <https://doi.org/10.1007/s42979-021-00815-1>.
- [7] “The History of Convolutional Neural Networks,” Glass Box, Apr. 13, 2019. <https://glassboxmedicine.com/2019/04/13/a-short-history-of-convolutional-neural-networks/>
- [8] O. G. Yalçın, “The Brief History of Convolutional Neural Networks,” Medium, Feb. 24, 2021. <https://towardsdatascience.com/the-brief-history-of-convolutional-neural-networks-45afa1046f7f>
- [9] “Convolutional Neural Networks: 1998-2023 Overview | SuperAnnotate,” [www.superannotate.com](http://www.superannotate.com). <https://www.superannotate.com/blog/guide-to-convolutional-neural-networks>

## REFERENCES

- [10] K.-S. Oh and K. Jung, "GPU implementation of neural networks," *Pattern Recognition*, vol. 37, no. 6, pp. 1311–1314, Jun. 2004, doi: <https://doi.org/10.1016/j.patcog.2004.01.013>.
- [11] R. Sarki, S. Michalska, K. Ahmed, H. Wang, and Y. Zhang, "Convolutional neural networks for mild diabetic retinopathy detection: an experimental study," Sep. 2019, doi: <https://doi.org/10.1101/763136>.
- [12] Md. M. Rahman, Md. S. Islam, R. Sassi, and Md. Aktaruzzaman, "Convolutional neural networks performance comparison for handwritten Bengali numerals recognition," *SN Applied Sciences*, vol. 1, no. 12, Nov. 2019, doi: <https://doi.org/10.1007/s42452-019-1682-y>.
- [13] I. Naseer, S. Akram, T. Masood, A. Jaffar, M. A. Khan, and A. Mosavi, "Performance Analysis of State-of-the-Art CNN Architectures for LUNA16," *Sensors*, vol. 22, no. 12, p. 4426, Jan. 2022, doi: <https://doi.org/10.3390/s22124426>.
- [14] B. Phan, "Comparing the Performance of Fully-Connected, Simple CNN, and ResNet50 for Binary Image...", *Medium*, Jul. 20, 2020. <https://towardsdatascience.com/comparing-the-performance-of-fully-connected-simple-cnn-and-resnet50-for-binary-image-5dae3cea034> (accessed Mar. 09, 2024).
- [15] B. K, "Object Detection Algorithms and Libraries," *neptune.ai*, Aug. 18, 2021. <https://neptune.ai/blog/object-detection-algorithms-and-libraries>
- [16] Jason Brownlee, "A Gentle Introduction to Object Recognition With Deep Learning," *Machine Learning Mastery*, Jul. 05, 2019. <https://machinelearningmastery.com/object-recognition-with-deep-learning/>
- [17] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5)." Available: <https://arxiv.org/pdf/1311.2524v5.pdf>
- [18] Chingis Oinar, "Object Detection Explained: R-CNN - Towards Data Science," *Medium*, Mar. 20, 2021. <https://towardsdatascience.com/object-detection-explained-r-cnn-a6c813937a76>
- [19] P. Potrimba Sep 25, "What is R-CNN?," *Roboflow Blog*, Sep. 25, 2023. <https://blog.roboflow.com/what-is-r-cnn/>

## REFERENCES

- [20] A. F. Gad, "Faster R-CNN Explained for Object Detection Tasks," Paperspace Blog, Nov. 13, 2020. <https://blog.paperspace.com/faster-r-cnn-explained-object-detection/>
- [21] "R-CNN, Fast R-CNN, and Faster R-CNN for Object Detection - Shiksha Online," Shiksha.com, Jan. 06, 2023. <https://www.shiksha.com/online-courses/articles/object-detection-using-rcnn/> (accessed Mar. 10, 2024).
- [22] "VisibleBreadcrumbs," Mathworks.com, 2014. <https://www.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html>
- [23] "YOLO models for Object Detection Explained [Yolov8 Updated]," encord.com. <https://encord.com/blog/yolo-object-detection-guide/>
- [24] R. Kundu, "YOLO: Real-Time Object Detection Explained," www.v7labs.com, Jan. 17, 2023. <https://www.v7labs.com/blog/yolo-object-detection>
- [25] "Understanding a Real-Time Object Detection Network: You Only Look Once (YOLOv1)," PyImageSearch, Apr. 11, 2022. <https://pyimagesearch.com/2022/04/11/understanding-a-real-time-object-detection-network-you-only-look-once-yolov1/>
- [26] Z. Kelta, "YOLO Object Detection Explained: A Beginner's Guide," www.datacamp.com, Sep. 2022. <https://www.datacamp.com/blog/yolo-object-detection-explained>
- [27] "SSD: Understanding single shot object detection," Manal El Aidouni, Jun. 25, 2019. <https://manalelaidouni.github.io/Single%20shot%20object%20detection.html>
- [28] "14.7. Single Shot Multibox Detection — Dive into Deep Learning 1.0.3 documentation," d2l.ai. [https://d2l.ai/chapter\\_computer-vision/ssd.html](https://d2l.ai/chapter_computer-vision/ssd.html) (accessed Mar. 10, 2024).
- [29] P. P. Aug 9, "What is Mask R-CNN? The Ultimate Guide.," Roboflow Blog, Aug. 09, 2023. <https://blog.roboflow.com/mask-rcnn/>
- [30] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN." Available: <https://arxiv.org/pdf/1703.06870v3.pdf>

## REFERENCES

- [31] E. Odemakinde, "Mask R-CNN: A Beginner's Guide," *viso.ai*, Mar. 19, 2021. <https://viso.ai/deep-learning/mask-r-cnn/>
- [32] X. Xu et al., "Crack Detection and Comparison Study Based on Faster R-CNN and Mask R-CNN," *Sensors*, vol. 22, no. 3, p. 1215, Feb. 2022, doi: <https://doi.org/10.3390/s22031215>.
- [33] A. Ammar, A. Koubaa, M. Ahmed, A. Saad, and B. Benjdira, "Aerial Images Processing for Car Detection using Convolutional Neural Networks: Comparison between Faster R-CNN and YoloV3," doi: <https://doi.org/10.3390/1010001>.
- [34] L. Tan, T. Huangfu, L. Wu, and W. Chen, "Comparison of YOLO v3, Faster R-CNN, and SSD for Real-Time Pill Identification," Jul. 2021, doi: <https://doi.org/10.21203/rs.3.rs-668895/v1>.
- [35] K. Chahal and K. Dey, "A Survey of Modern Object Detection Literature using Deep Learning." Available: <https://arxiv.org/pdf/1808.07256.pdf>
- [36] S. Kumar, Dr. A. K. Sinha, and Dr. N. Kumar, "Literature Review of Moving Object Detection Using Machine Learning," *International Journal for Research in Applied Science and Engineering Technology*, no. 9, pp. 796–801, Sep. 2022, doi: <https://doi.org/10.22214/ijraset.2022.46702>.
- [37] J. Zhou and Q. Liu, "Computer vision-based method for online measuring the moisture of iron ore green pellets in disc pelletizer," *Powder Technology*, vol. 408, 2022.
- [38] M. T. Wang et al., "Corner detection-based dimensional measurement of industrial components using smartphone cameras," 2023. [Online]. Available: <https://www.semanticscholar.org/paper/be498bf88cb0207765fa1f1b5b34134bee30966a>
- [39] Y. Zhang et al., "Lay length measurement in wire ropes using Mask R-CNN and phase correlation," 2023. [Online]. Available: <https://www.semanticscholar.org/paper/8d701e1d0207c3fc48c8327b1d61b9cb99d3cade>
- [40] H. Yin et al., "A computer vision-based method for measuring object dimensions using improved Canny edge detection and RHT-LSM model fitting," 2022. [Online]. Available:



## REFERENCES

<https://www.semanticscholar.org/paper/9ad9a33bdc363c688a6afc8bd6909524a0adb5c5>

[41] K. Gupta and T. Sharma, “Changing Trends in Computer Architecture: A Comprehensive Analysis of ARM and x86 Processors,” *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 7, no. 3, pp. 619–631, Jun. 2021, doi: <https://doi.org/10.32628/cseit2173188>.

# FISH PELLET MEASUREMENT SYSTEM FOR FOOD INDUSTRY



## METHOD

- 1. Capture Images:**
  - Capture images of fish pellets to build a dataset.
- 2. Preprocess Data:**
  - Label images with VGG Image Annotator and preprocess for training.
- 3. Train Model:**
  - Train a Mask R-CNN model to detect and measure fish pellets using TensorFlow and Keras.
- 4. Develop Algorithm:**
  - Create an algorithm to calculate fish pellet dimensions using detected masks.
- 5. System Integration:**
  - Connect Raspberry Pi for image capture and personal computer for processing, using MQTT for communication.
- 6. Build GUI:**
  - Develop a user-friendly interface with HTML/CSS/JavaScript to display real-time measurements.
- 7. Measure and Visualize:**
  - Convert pixel values to real measurements, draw bounding boxes, and display results.
- 8. Test and Validate:**
  - Compare algorithm measurements with manual measurements for accuracy.
- 9. Deploy:**
  - Set up the system for real-time operation and adjust for optimal performance

## INTRODUCTION

The project aims to enhance the precision and efficiency of measuring fish pellets in the food industry. Traditional methods are manpower-intensive and prone to inaccuracies, leading to operational inefficiencies. This project introduces an automated system using computer vision and deep learning to streamline the measurement process.

## SCOPE

The system integrates Raspberry Pi and a personal computer to capture images, analyze pellet sizes, and provide real-time feedback. It employs Mask R-CNN for object detection and ensures scalability for industrial applications.

## OBJECTIVE

- To present an improved and efficient method for determining fish pellet diameter to address the current industry issue.
- To develop a deep learning model to predict the area of the fish pellet.
- To develop an algorithm to calculate the height and width of the fish pellet by extracting the information from the detection.
- To integrate the computer vision, deep learning model and algorithm with the Raspberry Pi hardware.
- To develop a user-friendly GUI and measurement station for convenient observation.
- To validate the performance and accuracy of the measuring system.

## CONCLUSION

The automated system significantly improves measurement accuracy and efficiency in the fish pellet industry, reducing reliance on manual processes while delivering precise results. This innovation contributes to the industrial adoption of computer vision technology.