DESIGN AND DEVELOPMENT OF AN IMAGE-GUIDED VISION SYSTEM FOR ROBOTICS PALLETIZING

MOHAMAD ZAID BIN MOHAMAD ZAIHIRAIN

FACULTY OF ENGINEERING
UNIVERSITY OF MALAYA
KUALA LUMPUR

2021

DESIGN AND DEVELOPMENT OF AN IMAGE-GUIDED VISION SYSTEM FOR ROBOTICS PALLETIZING

MOHAMAD ZAID BIN MOHAMAD ZAIHIRAIN

RESEARCH PROJECT SUBMITTED TO THE FACULTY OF ENGINEERING UNIVERSITY OF MALAYA, IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF MECHANICAL ENGINEERING

FACULTY OF ENGINEERING
UNIVERSITY OF MALAYA
KUALA LUMPUR

2021

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: Mohamad Zaid Bin Mohamad Zaihirain

Matric No: S2000546/1

Name of Degree: Master of Mechanical Engineering

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

Design and Development of and Image-Guided Vision System for Robotics

Palletizing Field of Study: Computer vision, Robotics and Automation

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:

Designation:

DESIGN AND DEVELOPMENT OF AN IMAGE-GUIDED VISION SYSTEM FOR ROBOTICS PALLETIZING

ABSTRACT

The ever-evolving trend in modern manufacturing techniques has led to a shift from the conventional method to the heavily automated manufacturing process. Modern intelligent technologies are being used to simplify, accelerate, and improve the quality of traditional manufacturing methods. Automation of production lines through robotic implementation can improve the manufacturing performance while lowering the associated costs because it helps standardize the stacking and palletization procedures. Current automation on palletizing system relies heavily on a predetermined sorting system due to its inability to detect irregular-shaped object. A vision system is needed to increase the flexibility of the robotic palletizing system by detecting the type of object and its orientation. To this end, this project aims to design and develop an image-guided vision system through the application of YOLO object detection and OpenCV for small-scale robotics palletizing of non-uniform shaped object, i.e., 3D printed chicken wings and drumsticks. A YOLO object detection model is trained using 5000 images containing the chicken wings and drumstick. This object detection model is then used to determine the type of object. Then, this information on type of object is used along with an orientation detection program to find each object's orientation. The orientation detector is programmed with a contour detector called "Canny Edge detector" and a "fitEllipse" function that generates the angle of orientation. Using the location and orientation information generated by the detection programs, a pick and place operation is simulated in RoboDK. Through several case studies, this detection models works great if the objects in the images are arranged in a certain way, i.e., not closely packed together, or overlaps

onto one another. In an ideal case, the pick and place simulation work flawlessly with the information obtained from the YOLO object detection and orientation detection program.

Keywords: Robot vision, YOLO object detection, Orientation detection, RoboDK

REKA BENTUK DAN PEMBANGUNAN SISTEM PENGLIHATAN BERPANDUKAN IMEJ BAGI PALETISASI ROBOTIK

ABSTRAK

Kecenderungan ke arah teknik pembuatan moden yg sentiasa bertambah baik telah menyebabkan peralihan daripada kaedah konvensional kepada proses pembuatan secara automatik. Teknologi pintar moden digunakan untuk mempermudah, mempercepat, dan meningkatkan kualiti kaedah pembuatan tradisional. Automasi barisan pengeluaran melalui pelaksanaan robot dapat meningkatkan prestasi pembuatan sambil menurunkan kos yang berkaitan kerana membantu menyeragamkan prosedur susun dan palletisasi. Automasi semasa pada sistem palletizing sangat bergantung pada sistem penyusunan yang telah ditentukan kerana ketidakmampuannya untuk mengesan objek berbentuk tidak seragam. Sistem penglihatan robotic diperlukan untuk meningkatkan fleksibiliti sistem paletisasa robotik dengan mengesan jenis objek dan orientasinya. Untuk tujuan ini, projek ini bertujuan untuk mereka bentuk dan membangunkan sistem penglihatan berpandukan imej melalui penerapan pengesanan objek YOLO dan OpenCV untuk paletisasi robotik skala kecil untik objek berbentuk tidak seragam, sebagai contoh, kepak dan paha ayam yg dicetak secara 3D. Model pengesanan objek YOLO dilatih menggunakan 5000 gambar yang mengandungi sayap dan paha ayam. Model pengesanan objek ini kemudian digunakan untuk menentukan jenis objek. Kemudian, maklumat mengenai jenis objek ini digunakan bersama dengan program pengesanan orientasi untuk mencari orientasi setiap objek. Detektor orientasi diprogram dengan detektor kontur iaitu "Canny Edge detector" dan fungsi "fitEllipse" yang menjana sudut orientasi. Menggunakan maklumat lokasi dan orientasi yang dihasilkan oleh kedua-dua program pengesanan, operasi memetik dan meletakkan disimulasikan dalam RoboDK. Melalui beberapa kajia, model pengesanan ini berfungsi dengan baik jika objek dalam gambar disusun dengan cara tertentu, iaitu, tidak terlalu rapat antara satu sama lain, atau saling bertindih. Dalam kes yang ideal, simulasi

memetik dan meletakkan berfungsi dengan sempurna menggunakan maklumat yang diperoleh dari program pengesanan objek YOLO dan pengesanan orientasi objek.

Kata kunci: Penglihatan robotic, Pengesanan objek YOLO, Pengesanan orientasi, RoboDK

ACKNOWLEDGEMENTS

All praises to Allah S.W.T. for granting me the knowledge and perseverance in completing my Master of Mechanical Engineering research project report. I am forever grateful to Allah for all the blessings that has been blessed upon me.

I would like to express my deepest appreciation to my project supervisor, Dr. Yap Hwa Jen from the Mechanical Engineering department for sharing his knowledge and expertise with me. The ongoing help and guidance from him are greatly appreciated.

To my parents and family who supported and motivated me throughout my life, I am infinitely grateful. I would not be able to complete my master's degree without the constant support of my beloved family.

I am grateful for the lecturers at University of Malaya that have persevered during this pandemic in which they continued to provide helpful knowledge to us students. It has been wonderful and enjoyable learning experience despite the online learning environment.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AP	:	Average Precision
CAD	:	Computer Aided Design
CCD	:	Charge Coupled Device
CCW	:	Counterclockwise
CNN	:	Convolutional Neural Networks
CPU	:	Central Processing Unit
CW	:	Clockwise
FPS	:	Frame Per Second
GUI	:	Graphical User Interface
IC	:	Integrated Circuit
IDE	:	Integrated Development Environment
mAP	:	Minimum Average Precision
OpenCV	:	Open Computer Vision
PLA	:	Polylactic Acid
PLC	:	Programmable Logic Controller
RAM	:	Random-access Memory
RoboDK	:	Robot Development Kit
YOLO	:	You Only Look Once

LIST OF APPENDICES

CHAPTER 1: INTRODUCTION

1.1 Overview of the Research

As the world moves towards a new age of technology, particularly the fourth industrial revolution, the automation of traditional manufacturing practices advanced rapidly. Modern intelligent technologies are being used to simplify, accelerate, and improve the quality of traditional manufacturing methods. Robots are great examples of current intelligent technology that have been used for almost six decades since the first industrial robot, the Unimate, was implemented at General Motors production plant in 1961 for die casting handling and spot welding (RobotWorx, n.d.).

Automation of production lines through robotic implementation can improve the manufacturing performance while lowering the associated costs because it helps standardize the stacking and palletization procedures (Wurll, 2016; Yi et al., 2017). For the past decade, the amount of stock required to fulfil the impending operations for industrial robots has steadily increased and is expected to continue with the same trend in the near future (Moura & Silva, 2018). This demonstrates that the need for industrial robots has continued to rise, since most manufacturing companies would like to invest in technologies that may boost production rates while keeping costs low.

Moura and Silva (2018) mentioned that among the numerous applications for robots in the industrial sector, handling activities such as palletizing have been prevalent since they play an essential role in the last step of contemporary production lines. Since palletizing is one of the last processes before product transportation, it has a substantial impact on the manufacturing line. The order in which the items are placed on the pallets has a major influence on the stack's stability, which has a profound influence on the product's transportation process to the consumer. Most automated palletizing operations rely on the accuracy and repeatability of the industrial robot as it will be doing the same operations multiple times. These operations usually do not depend on variation in the

products locations since the products will be packaged in a box and placed on a specified location and orientation. The robot would just need to pick the already packaged products and place it on the pallet without the integration of a live visual input. Although this appears to be an efficient method, it lacks flexibility because it can only be used when the product is already packed or boxed and relies on the assembly line to orient the packaged commodity. However, it is not a terrible technique of employing robots to automate the manufacturing line, it is dependable since it can perform the same work repeatedly while using less resources.

For other palletizing systems, in addition to the fixed location and orientation of the object, employ sensors to detect whether the packaged products have arrived at the picking point, and the robot will subsequently select the already packed goods. Sensors are the preferred approach for regularly shaped objects such as boxes and packed products since just the location of the product is required for the robot to pick up the object. The packaged products will be sorted by the custom mechanism on the assembly line, and the robot will subsequently pick the box using suction tools or specialized grippers. This method is preferred as fewer variables is needed to program the robot since the location of the object to be picked up is consistent and the orientation can be neglected since they are all in a uniformly shaped packages or already oriented through the custom mechanisms in the assembly line.

In order to improve the flexibility of the robotic automation system, an artificial intelligence aspect can be integrated into the system. A vision system can be implemented on the industrial robot through the use of cameras and a programmable logic controller (PLC). A live image will be captured through the camera and will be processed to determine the location and orientation of the goods. Thus, the sorting mechanism can be removed from the assembly line since the robot itself can distinguish the location and orientation of the goods through its own vision system. By employing an image-guided

vision system on the industrial robot, an irregular-shaped object can also be distinguished and picked up by the robot. A much more complicated programming is required as well as the application of machine learning algorithms on the vision system so that the robot can determine the type, location, and orientation of the irregularly shaped object. The vision system would be trained to identify the specific objects to be palletized. Therefore, automation of palletizing operations based on machine vision is a viable option for reducing reliance on human labor and can decrease costs while boosting assembly line productivity.

In order for the industrial robot to detect irregularly shaped objects, an object detection algorithm need to be applied. There are two types of deep learning-based object detection methods: candidate region-based models and regression-based models. The basis of a two-stage detection model is the candidate region-based model, which generates the region proposals in the first stage. Following that, features are then extracted from the proposals to obtain the bounding box and classification regression (Ren et al., 2017). Many applications such as fruit recognition had implemented this object detection method through the use of deep Convolutional Neural Networks (CNN). Despite the recognition accuracy of the two-stage object detection, it is still considered as a slow approach and cannot be applicable to real time applications due to its requirements of needing two-stages of processes to recognize the object This is where single-stage detectors flourish, since they approach object detection problem as a straightforward regression model that accepts the entire image as input and outputs the bounding boxes and class probabilities at the same time. Class probabilities is the type of the object detected. This single-stage approach has made this model much faster than the two-stage object detection methods. You Only Look Once (YOLO) model is a single-stage object detection model that is developed by Redmon et al. (2016) to be use in real time application as it is fast. Multiple versions of YOLO models have been developed to either increase it detection time and

accuracy, or to be use for various applications such as number plates detection or electronic object detection (Junos et al., 2020).

Although the YOLO object detection can detect the type of object and its location through the bounding boxes, it cannot establish the orientation of the object. Thus, another algorithm is required in addition to the object detection model in order to accurately determine the object's orientation to be picked up by the robotic arm. If a gripper is used on the robotic arm, the orientation of the object is critical as each object has a specific location on its body that can be readily picked up by the robotic arm and the orientation is also needed so that the robot can place the object in the desired orientation to be packaged. This is done through the use of a contour detection algorithm along with a fit ellipse function in Open Computer Vision (OpenCV) python. The contour detection method is used to trace the boundary of the object, which can determine its shape/form, and with the help of the fit ellipse function, the orientation of the object can be determined. The process begins by extracting each bounding boxes from the object detection image (i.e., image of 3D printed chicken drumstick and wing), then the contour detection traces the perimeter of the object, which are then fitted with an ellipse to establish the orientation of the object.

These detection models would then be utilized to identify and locate the object on the pallet for the pick and place operation, which would be simulated in a software called Robot Development Kit (RoboDK), an industrial robot and robot programming simulator that is both comprehensive and cost-effective. The advantage of using the simulation and offline programming capabilities of RoboDK is that robots may be programmed outside of the production environment which saves both time and utilities (RoboDK, 2021b). The RoboDK simulator is a universal offline programming tool that makes it possible to program robots of all kinds as well as to generate brand-specific robot programs. RoboDK owns a library of more than 500 industrial robot arms, from famously known robot

manufacturers such as ABB, Fanuc, KUKA and Universal Robots, as well as numerous additional tools and objects that may be used in the simulations of robots (Garbev & Atanassov, 2020). RoboDK's simple and user-friendly Graphical User Interface (GUI) makes it easier to simulate and program industrial robots without the need in field experience. The free version of RoboDK, as well as the trial edition of RoboDK, are adequate to inform and familiarize the programmer/user with the system's capabilities.

Thus, the combination of the vision system equipped with the YOLO object detection and the orientation detection program will enable the simulation of pick and place of irregular shaped object, in this case, 3D printed chicken wing and drumstick are used. The location and orientation provided by the vision system will enable the robotic arm equipped with a gripper to pick the object and place it in the correct pallet with the desired orientations.

1.2 Problem Statement

The following are some of the issues that prompted the creation of this project:

- Current automation on palletizing system relies on a predetermined sorting system due to its inability to detect irregular-shaped object. A vision system is needed in order to increase the flexibility of the robotic palletizing system by detecting the type of object.
- Most robotic palletizing system can only work on already packaged/boxed object since they are easily picked up by the robot because of their uniform shape. However, if the object is irregular, i.e., chicken drumstick, the orientation of the object is needed to enable the robot to pick it up using the gripper as random picking position would result in improper placement of the object.

1.3 Aim of the Research

To design and develop an image-guided vision system through the application of YOLO and OpenCV for small-scale robotics palletizing of non-uniform shaped object, i.e., 3D printed chicken wing and drumstick.

1.4 Objectives of the Research

The objectives of the research are listed below:

1. To develop an algorithm to identify the product with non-uniform shapes.
2. To identify the location and orientation of the products for end-effector picking/grasping.
3. To integrate the vision system with robotic arms for the palletizing process.

1.5 Scope of the Research

This research is limited to only developing a vision system that could detect the type, location, and orientation of the object for a pick and place operation. The vision system will provide the data for the robotic arm for the palletizing operation. Only simulation in RoboDK will be used to demonstrate the detection algorithm workings with the use of real images of randomly sorted objects. The objects used for detection in this research are 3D printed chicken wings and drumstick, and the robotic arm used for simulation is the UR10e from Universal Robots. The camera used in this research is a computer webcam that is easily accessible.

1.6 Report Organization

The remaining parts of this research report will be organized and modelled as following:

- Chapter 2 deals with the literature review of relevant research associated with this research project namely robotic palletizing, YOLO object detection and orientation detection.

- Chapter 3 consist of the research methodology which describes the methods and procedures used in conducting this research project along the simulations involved.
- Chapter 4 contains the implementation of the detection models through simulation of an ideal case and other case studies used to test the detection models.
- Chapter 5 concludes the research project by summarizing the report and providing future work on this research.

CHAPTER 2: LITERATURE REVIEW

The study of industrial automation has now taken on a worldwide platform, with engineers and researchers focusing largely on ease of implementation. In this literature review, this research subject of image-guided vision system for robotic palletizing will be divided into four sections namely robotic palletizing, vision/image-guided systems, YOLO object detection and Orientation detection.

2.1 Robotic Palletizing

Several studies had been conducted with regards to industrial robotics palletizing to either improve the stacking algorithm or to improve the structure of the robotic palletizing system. Wurll (2016) did a study on mixed case palletizing using industrial robots which starts by converting the customer's order into stable, organized, and dense stacks, which will then give instructions to the robots to build the pre-determined pallet stacks before being transported. Mixed palletizing is an industrial stacking technique applied by placing boxed/packaged products of different types and sizes onto a single pallet (palletizer.org). Mixed palletizing allows the factory to transport a smaller number of pallets since each pallet is able to carry a variety of products with the same level of stability as stacking one type of product due to the stacking algorithm developed. The advantages of implementing a robotic mixed pallet building system is that it can reduce labour and save shipping cost by fully filling the pallets by mixing the variety of products instead of just shipping one type of product on each single pallet (Nowak, 2011). For instance, the grocery store might just need a few bottles of detergents, however, in order to fully utilize the pallets, the warehouse will want to stack the pallets with other products that the store needs. However, to enable the automated mixed palletizing system, more data or information regarding the products and consumer need to be organized and kept track of for the warehouse management system. This mixed palletizing system relies heavily on stacking

algorithm as different products have different shape and size for their packaging. Most of the companies presented in the studies regarding robotic palletizing relies heavily on a palletizing algorithm which calculate the optimum arrangement of the packaged products with regard to the hardware constraints of the robotic gripper arm in order to automate the stacking process (Wurll, 2016; Xu et al., 2016). The stacking algorithm used would improve the production efficiency by increasing the rate at which the packaged product is palletized and also increasing the number of products being palletized. However, the stacking algorithm used are pre-determined as it only depends on the weight and size of the packaged product since most the finalized products would be in a regular shaped boxes or packaging.

Other than relying on the stacking algorithm to further improve the pelletizing system, some researchers aim to increase the efficiency of the palettizing robot by increasing its mobility. A typical robotic palletizing involves three main parts which are the palletizing robot, driving or conveyer belt and the pallets (Xu et al., 2016). The palletizing robot is usually a robotic arm with four degrees of freedom which enables it to lift the products from the conveyer belt to the pallets. The robotic arm itself is fixed onto stationery platform which constrains the workspace for the robot as it only allows the robot to be placed on one specific location. This would result in a transporting system need to be included in order to transfer the pallets in another location. Despite this disadvantage, this fixed palletizing robot allows for a much bigger and heavier products to be palletized as a large robotic arm can be use. Yang et al. (2018) developed an omnidirectional compact palletizing robot that combines transport and stacking functions for small product palletizing in order to rival the standard robotic palletizing that are large and stationery. They developed this robot with increased mobility in mind as they aim to integrate the staking ability of the robot and its ability to relocate the finished pallets. This robot uses a Mecanum wheels to provide more mobility as it can travel to the targeted location using

the fastest path due to its capability to turn in narrow spaces. Mecanum wheels is a type of wheel with multiple diagonally positioned roller attached to its circumference (Dejan, 2019). These multiple diagonally positioned rollers enables the wheels to move in any direction using the same driving components as for regular wheels. Each Mecanum wheels are independently drive, in order to utilise its multidirectional capabilities. By combining different rotation of the wheels, the robot will be able to move in any direction using only a small space. To further clarify on this notion of combining different rotation of the wheels, an image showing the direction with regard to the rotation of the wheels is shown in Figure 2.1. The movement of the Mecanum wheel, (blue: wheel drive direction, red: vehicle moving direction). a) Moving straight ahead, b) Moving sideways, c) Moving diagonally, d) Moving around a bend, e) Rotation, f) Rotation around the central point of one axle. This robot is able to achieve high stacks of cargo through this unique mechanical design and staking features. This high stacking method combine with robot's ability to move in narrow spaces allows for more efficient space usage.

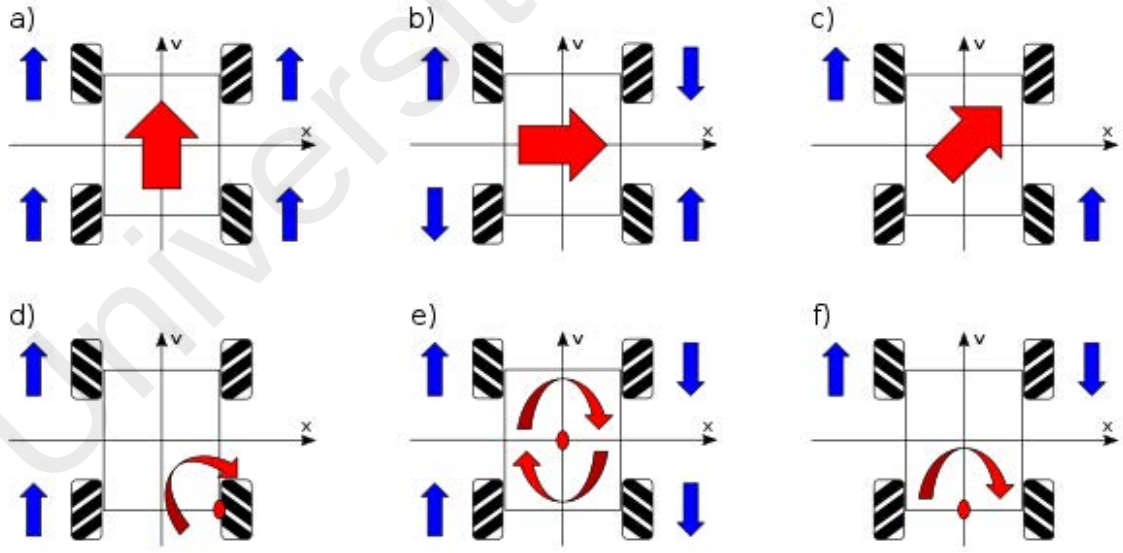

Figure 2.1: Movement of Mecanum wheel (WIKIPEDIA, 2020b)

To further improve upon the palletizing robot, several studies had been done in order to find the suitable method to analyse the performance of the robots. Guan and Wang (2011) did a study on the mechanical design and kinematic analysis on a palletizing robot. This study provided some mathematical derivation using kinematic analysis in order to

be used to analyse the mechanical aspect of the robotic arm to be used in palletizing system. The kinematic analysis was testified using practical experimental data which is then used to simulate the working space of the robotic arm using MATLAB. This kinematic analysis would allow the users to understand the degree of freedom of the robotic arm through the expression provided which would then allow for a more complex movement to be applied by to the robot by manipulating the provided expressions. The results obtained from this research would help in conducting theoretical analysis and further exploitation of using the robotic arm in a production palletizing system. In addition to analysing the kinematic elements of the robot, a study in reducing the energy consumption of the robot had also been conducted. Fu et al. (2019) did a research by using RobotStudio software in reducing the energy consumption for the palletizing robot. This study aims to increase the efficiency of the robot by implementing an optimal speed and movement of the robot that would use less energy but still accomplishing the task. RobotStudio is a software that allows user to program a robot in the virtual world before beginning its real-life operations, which means this software is able to provide simulations of the working program for the robot. Using this software, Fu et al. (2019) investigated the energy required with each type of movement commands and compared them in order to find which command suits the tasks while using less energy. RobotStudio enables a realistic simulation to be perform and can create a much easier energy-optimal path planning for the robotic arm in the palletizing operations.

There are also a few studies done to further increase the efficiency and production rate for robotic palletizing by implementing redundant safety mechanism. Since it is hard to ensure a fully successful application of using robots in palletizing, the overall automation system must have the capability to restore quickly when failure or by equipping a redundant alternative mechanism to continue the production when the main palletizing

system breaks down (Yuan & Wang, 2016). This feature is needed as it would help in stabilizing the production rate of the factory.

All of the studies mentioned regarding robotic palletizing really highlights the importance of automation in the manufacturing industry as it helps to reduce the manufacturing time while increasing the manufacturing quantity. Despite all this advantages, the studies on robotic palletizing does not include the use of machine learning especially with regards to robotic vision system which is what this research report is aiming for.

2.2 Vision/Image-Guided Systems

Multiple studied had been done regarding vision or image-guided system for robotic applications. This vision system would be use to guide the robot in order to accomplish a specific task. Zhang and Skaar (2009) developed a robotic de-palletizing system by combining camera-space manipulation with laser-assisted image analysis to find the gaps on paper bag stack. Robotic de-palletizing is more difficult to automate compared to palletizing work as it requires a much more sophisticated vision system that would allow the robot to detect the gaps between each layer of paper stack in this case. The vision system applied to this study is by using laser spots and camera-space manipulation to locate the edges and corners of the stack in order to find the gaps to be fitted with an insertion dowel. Although this study includes an image-guided system, it can only be implemented on objects that have a regular sides, edges, and corners as it is used to insert a tool into the gaps on the stack to lift the paper bags. In order for this vision system to be implemented to irregular packaged products, a more sophisticated algorithm need to me applied in order to identify the overall shape and orientation of the product.

There are also other studies regarding vision-guided system that are not specific to palletizing process. Ji et al. (2012) conducted a study on a machine visions system that is

used to recognize and locate apples in order to harvest the fruits. This study aims to develop an automatic harvesting system that is guided using image processing algorithm in order to recognize which apples that can be harvested. They first start by implementing a colour Charge Coupled Device (CCD) camera to capture the images of the fruits, and these images will be processed using an industrial computer in order to recognize the fruits. Charge coupled device camera is a video camera that contain a transistorized light sensor on an integrated circuit (IC), in other word a digital camera (KISI, n.d.). CCD camera allows for visual input to be converted into digital image or video. During the image processing stage, a vector median filter is applied to remove the noises that would influence the image quality. This filter also highlights the apple fruits forward by weakening the noise and reducing the sharpness of the background. This filtered image is then followed to an image segmentation process where the apple in the image is further differentiated form the complex background that include leaves and branches. Then, an image shape recognition feature extraction algorithm is applied by removing the background completely after the apple fruit is recognized. These data would be used by the harvesting machine to locate the ripe apples to be harvested.

There are also other studies about image-guided robotic applications in the sports industry. In 2002, a robot that is capable of replacing a ball boy for a tennis court is developed (Webster & Brannon, 2002). This study aimed to develop a mobile robot that can interact with its environment by locating the objects near it and determine what are the objects. The robot relied on a charge-coupled device (CCD) camera as its vision system to identify the location of the tennis balls. The vision system employed in this study is an overhead camera mounted with its optical axis is directly perpendicular to the tennis court surface. They chose this type of configuration for the vision system as it helps to eliminate the error in distance measurements that come with onboard camera systems where the camera reference frame is always changing as the robot moves. The other

advantage of using the overhead camera vision system is that the pixels of the camera can be directly translated into physical coordinates. To identify the robot's own location and the location of the tennis balls, the vision system converts the pixels of the image into

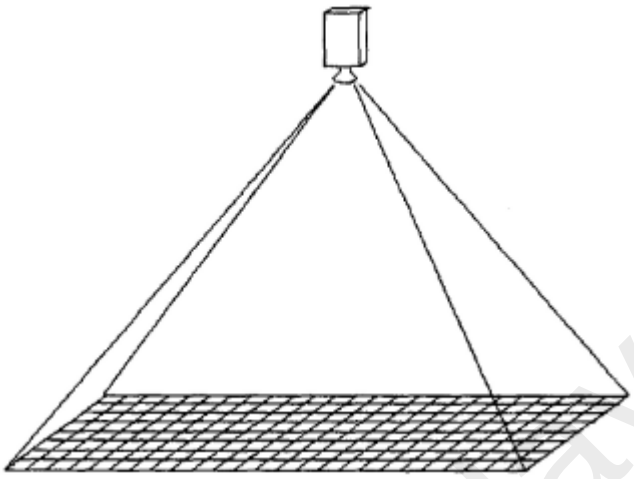

Figure 2.2: An overhead fixed camera causes the pixels to correspond to physical Cartesian coordinates (Webster & Brannon, 2002)

Cartesian coordinates which would simplify the calculations for location mapping as shown in Figure 2.2. For each image captured, the vision system must be able to identify the location and the orientation of the robot, the location of the tennis balls and any obstacles on the court before calculating the path for the robot to follow. The path must be able to collect all the ball in the shortest amount of time while avoiding the obstacles on the tennis court. In order for the vision system to recognise the orientation of the robot, they place an identifying marking on robot that is uniform in diameter with a recognizable shape in order to identify it orientation. This vision system also implements an RGB filter that removes any unnecessary information and classify the pixels into four categories: the robot, ball, obstacle, and the background. Several algorithms are then used to create a path for the robot to follow in order to collect all the tennis balls in the shortest amount of time. The algorithm used for this case, is quite simple as it only depends on 2-dimensional coordinate systems for the path of the robot to follow.

The studies mentioned regarding vision guided system for robotic application signifies many advantages to adding vision system in robots. For most of this vision system, the type of object is identified by a specific label, or they each have distinct features that is easily recognize by the robot. Since only few objects are involved, they only need to take into account a fixed number of parameters which is why machine leaning is not needed here. However, if multiple type of objects with irregular shapes and orientations are involved, an object detection algorithm is required.

2.3 YOLO Object Detection

You only look once (YOLO) model is a single-stage object detection model that is developed by Redmon et al. (2016) to be use in real time application as it is fast. Single-stage object detection is much faster and more robust compared to two-stage detectors as YOLO treats object detection problems as a simple regression problem that takes the image as an input and simultaneously generates class probabilities and multiple bounding boxes (Junos et al., 2020). Multiple applications have applied these YOLO detection model by modifying and training the model to suite the respective applications. The most common and latest model by Redmon et al. (2016) which is the YOLOv3 have the best detection performance with respect to its previous model but a long computation time is required to train the model due to network complexity. Therefore, a lighter version of YOLOv3 that can be trained a shorter time and satisfy real time object detection is developed which is called YOLOv3 tiny. YOLOv3 tiny is based on a constrained environment in which the memory, storage capacity and processing power are limited (Li et al., 2020) which means that a low computation cost is needed in order to train the model. In this research report, YOLOv4 tiny is used instead of YOLOv3 as it can detect object faster and more accurate as shown in Figure 2.3. YOLOv4 improves YOLOv3 average precision (AP) by 10% and frame per second (FPS) by 12% (Bochkovskiy et al., 2020). YOLOv4 tiny is applied in this research as it can be trained much faster with low

computation cost compared to YOLOv4, thus making it suitable when the equipment is

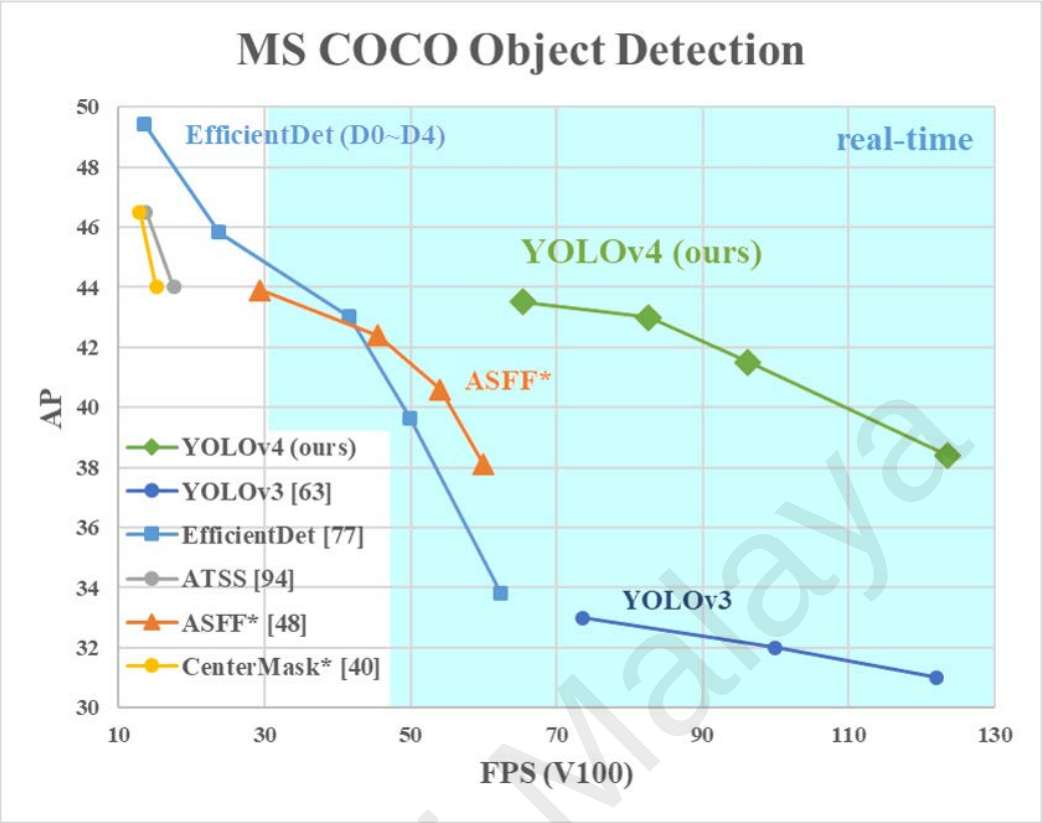

Figure 2.3: Comparison of the proposed YOLOv4 and other object detectors (Bochkovskiy et al., 2020)

limited. Due to the increase in the FPS, the AP will be slightly sacrificed but acceptable since the environment used for object detection in this research will not change.

2.4 Orientation and Shape/Contour Detection

Most of these studies on robotic palletizing are heavily on the stacking algorithm of the packaged products and for heavy-duty palletizing. The vision system used are for identifying the location of the product only and not for shape recognition. For this project, a vision system that can identify the product with non-uniform shape will be develop and will work alongside a robotic arm to correctly orient the product before palletizing. Shape recognition algorithm can further increase the flexibility of the robotic palletizing system as it allows for the vision system to recognise the shape and orientation of any products. Not many studies can be found regarding implementing a shape recognition algorithm in the palletizing processes. However, some studies regarding this matter are conducted for

other usage such as recognizing hand-palm orientation for sign language by Phadtare et al. (2012). These researchers aimed to develop an automatic gesture recognition specifically purposed of understanding sign languages by translating the hand-palm orientations. Their vision system also implements Microsoft Kinect to capture the colour and depth of the images being processed.

The vision system analyzed the depth data corresponding to only the hand-palm region and compare it to the examples shapes in the predetermined sign language databases. The other benefit of using the Kinect system is that it can provide human skeletal joint location which is then can be used to determine the location of the hand of the user. The Kinect system is manufactured for gaming purposes (XBOX) which lets the players controls the game using only their body movements instead of the traditional controllers (WIKIPEDIA, 2020a). The motion sensing abilities of the Kinect vision system allows for a better hand gestures recognition. Junyeong et al. (2013) also did a similar study using the Kinect vision system to develop a hand shape recognition for interaction between robots and humans. This study translated the hand gestures by laying simple shapes on top of the hand such as using ovals as the palm and lines as the fingers. They then translate these lines and ovals to specific meaning by using the predetermined databases.

Pagano et al. (2020) did a study aimed to develop and automatic gluing system for footwear industry that is guided using a vision system. The objective is that to further increase the manufacturing productivity by implementing a vision system in order to identify the shape and position of the unknown object and identify the gluing area that is usually located along the object perimeter. The vision system for this footwear gluing traces the perimeter/contour of the sole and it needs to be able to process 3-dimensional coordinates as it need to be able to identify the depth of the sole of the shoe as it often has

raised edges. The study uses a Microsoft Kinect V2 vision system as it consists of a RGB camera with a high resolution and an infrared sensor that enables distance measurement to be conducted. This vision system allows for the location and reconstruction of the shape of the object virtually which enables the trajectory of the gluing robot to be planned and executed.

In summary, several studies had been conducted in improving the palletizing process using robots. However, many studies regarding robotic palletizing are heavily rely on the stacking algorithm of a predetermined or uniformly shaped products. By implementing a vision system along with a shape recognition algorithm, a more flexible robotic palletizing process can be developed in order to reduce the amount of human interaction needed in the palletizing process. In this study, an image-guided vision system would be used along with a robotic arm to palletize an irregular shaped product. A shape recognition algorithm would be employed in the vision system to recognize the shape and the orientation of the product before packaging.

CHAPTER 3: METHODOLOGY

3.1 Introduction

This chapter presents the method and procedures that was done in developing the image-guided vision system for robot. The objects used for detection are 3D-printed chicken wings and drumstick as they are easier to manage compared to raw chicken. The objects used here are just for demonstration and can be swapped with other objects with new object detection training. Most of the coding and detection are done using a regular desktop webcam (Logitech C170 webcam) and PyCharm which is an Integrated Development Environment (IDE) in computer programming that is specific for the Python language. The robot simulation is done in the free version of RoboDK. Figure 3.1 is a flowchart showing the sequence of how this research project is conducted.

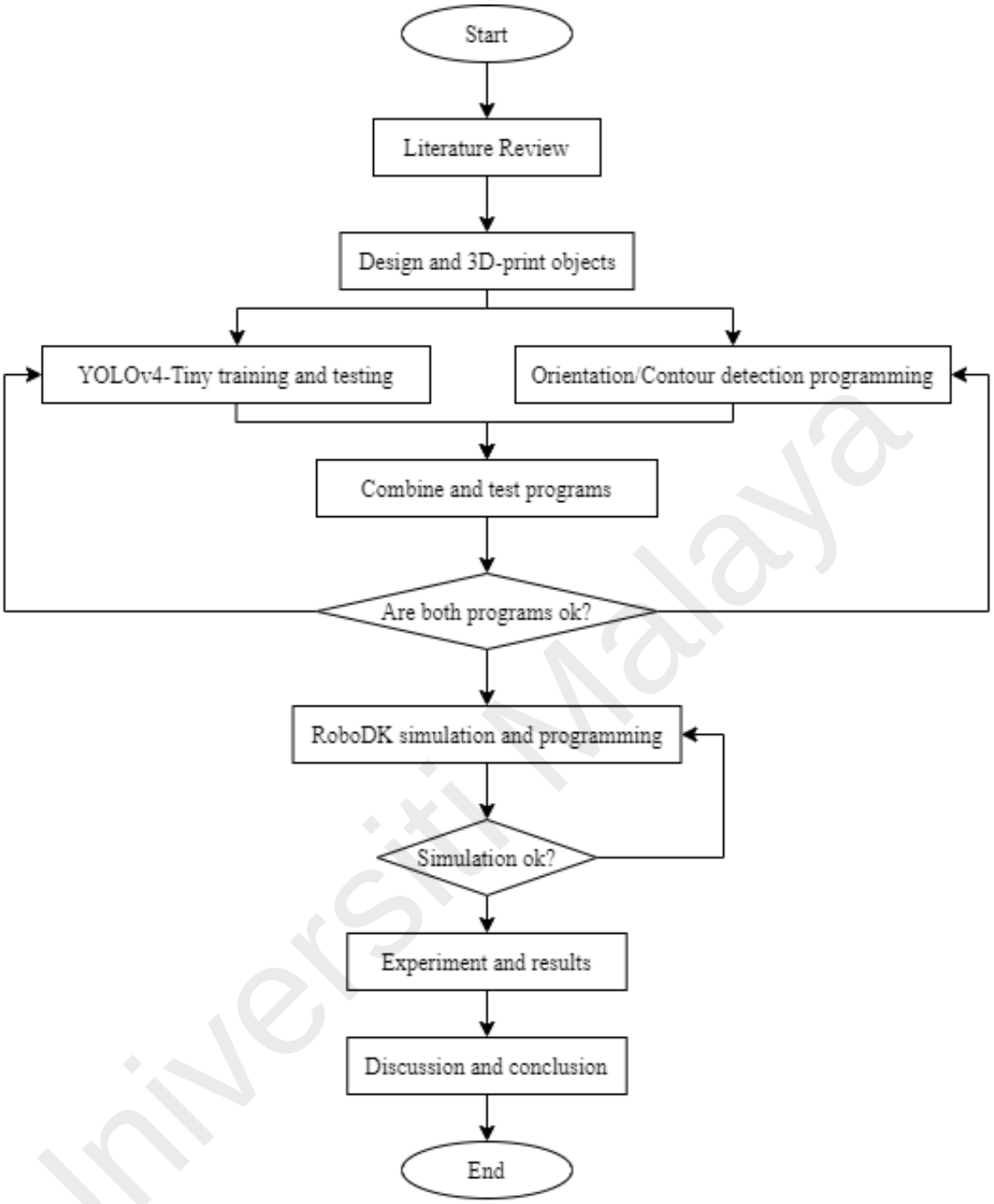

Figure 3.1: Research project flow chart

3.2 3D Printed Wings and Drumstick

The objective of this research is to detect non-uniformly shaped object; thus, the chosen object are chicken wings and drumsticks. The wings and drumsticks used here can be replaced with any other object, they are merely chosen in order to demonstrate the vision system developed. Using raw chicken wings and drumsticks would induce other non-essential problems such as hygiene, thus 3D-printed versions of the object will be used. The 3D model of the drumstick is obtained from an online 3D model platform "free3D.com" that offers a variety of 3D models shared throughout the community. Figure 3.2 shows the 3D model of the drumstick obtained from the website. As the 3D

Figure 3.2: 3D model of a chicken drumstick (printable_models, 2019)

model of the chicken wing is not available for free online, the wing model is designed in Solidworks, a Computer Aided Design (CAD) software. The design is not an exact copy

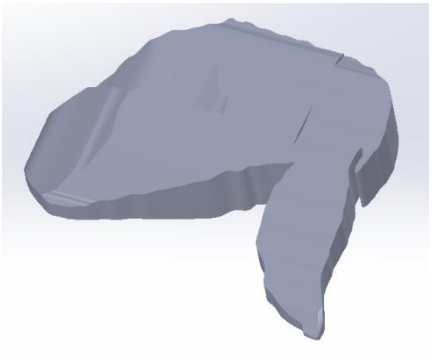

Figure 3.3: 3D model of a chicken wing designed in Solidworks

of the real chicken wing but have similar size and shape. The models are 3D printed using the Ender3 3D printer with the Polylactic Acid (PLA) filament. The type of material used to print the objects is not relevant to these study as the objects is only used to demonstrate the vision system.

3.3 YOLOv4-Tiny Training and Testing

For the object detection algorithm, the YOLOv4-tiny is used with three object classes namely drumstick, right wing, and left wing. For drumsticks, the left and right wide are similar in shape and size, thus only needing one class. However, for wings, the right and left side are mirrored, thus needing two classes to differentiate the wings. Figure 3.4 shows the top view of the drumstick and Figure 3.5 shows the top view of both the right wing and the left wing.

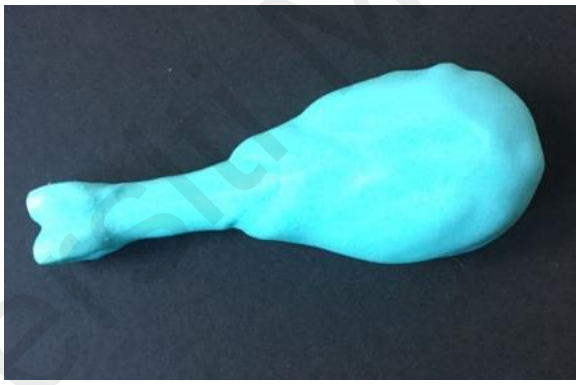

Figure 3.4: Top view of drumstick

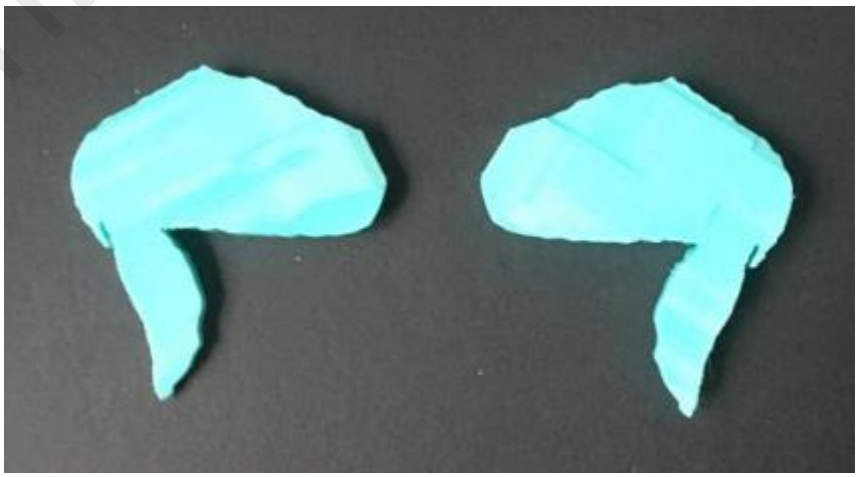

Figure 3.5: Top view of left wing and right wing in order from the left

Figure 3.6 shows the flowchart for the YOLO object detection training and testing procedure. The training process starts with the image acquisition and ends after testing is done.

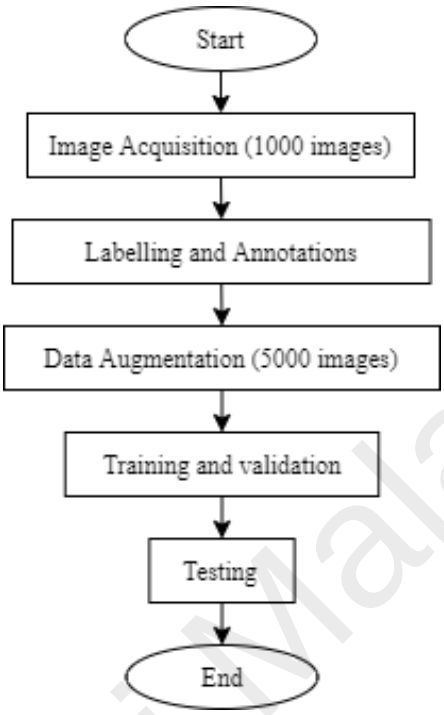

Figure 3.6: YOLOv4-tiny training and testing flowchart

3.3.1 Image Acquisition

The images are collected by using a regular desktop webcam (Logitech C170 webcam) through a Python code. Table 3.1 shows the specification of the C170 webcam used for data collection. Figure 3.7 shows the equipment setup for the image acquisition that consists of a desktop webcam, fluorescent light, and black panels. Black panels are used so that it would not reflect the light from the fluorescent bulb and damaging the image. The python code, "1-DataCollection1.py" in Appendix A is used to automate the image acquisition process by specifying the minimum blurriness percentage (50%). This means that if the image is below 50% in blurriness, then it will not be saved. The images are saved with a 640 x 480 resolution which in this project, translates to a 640 mm by 480 mm physical cartesian coordinates. This allows for a one-to-one ratio between the images and the coordinates for the robot later. This resolution can be scaled to match any physical

coordinates, but, in this case, it is not significant, since this research would only be a demonstration of a vision system.

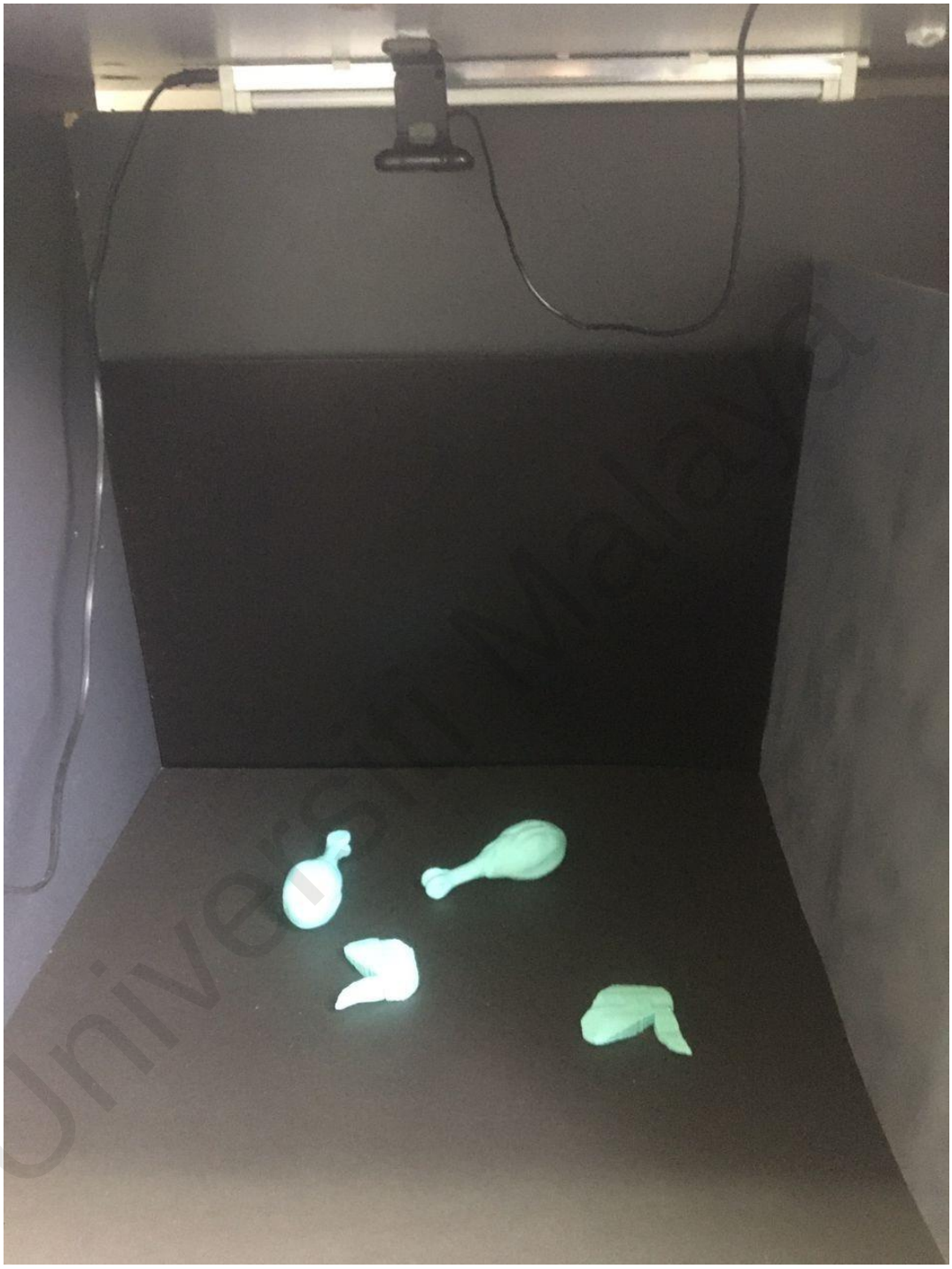

Figure 3.7: Equipment setup for image acquisition and object detection

The images are then filtered through manually to check for repetitions and hand movement since the object are placed randomly by hand in each frame. All the objects are kept inside the resolution boundary so that none of the object is only showing part of its body, and objects that overlaps are not taken as part of the data. The code is executed

until 1000 images are obtained and all the images are sorted/shuffled randomly and renamed. Figure 3.8 shows four sample images out of the 1000 images gathered for the YOLO training process.

Table 3.1: Specification of Webcam used for image acquisition

Type	USB Webcam
Max Resolutions	1024 x 768
Interface	USB 2.0 – 4 pin USB Type A
Manufacturer	Logitech
Model	C170

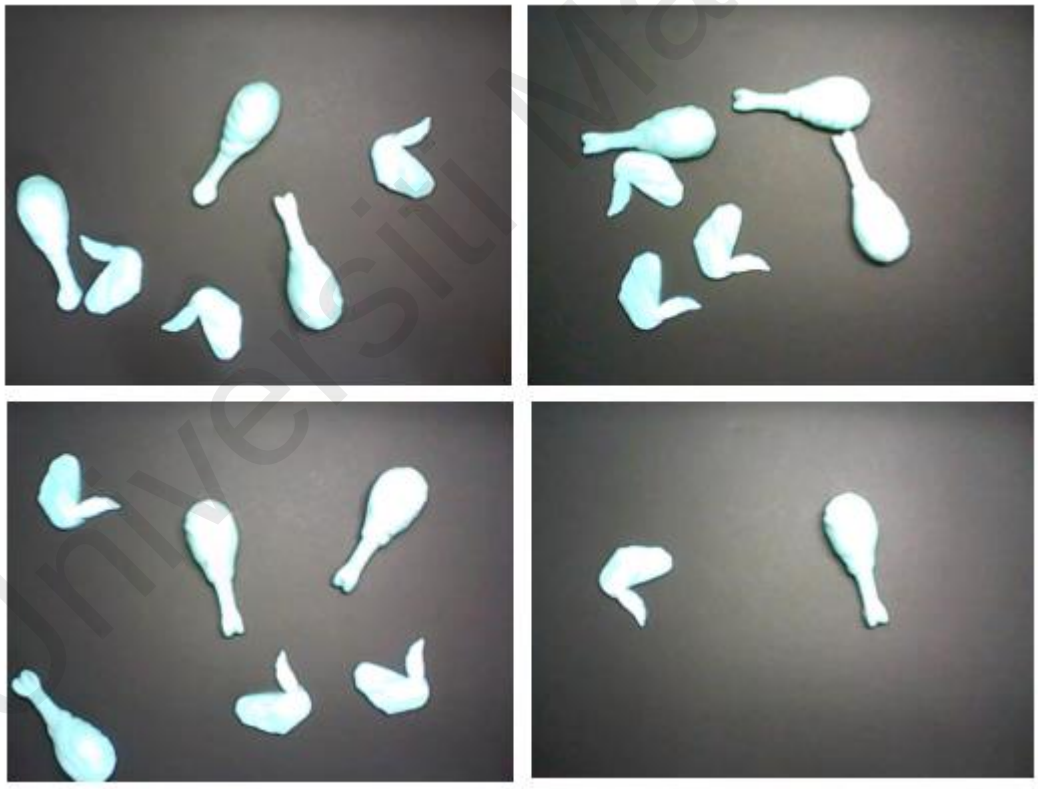

Figure 3.8: Sample output of the image acquisition process (4 out of 1000 images)

3.3.2 Labelling and Annotation

The 1000 images were manually labelled and annotated by drawing bounding boxes and classifying the class/categories for each object in the images. An open-source labelling and annotation tool called LabelImg created by Lin (2017) was used for this process. All the images were annotated into three classes that were named LeftWing, RightWing and Drumstick. In each image, every object was labelled with a bounding box that represents the location of the object in the image. Figure 3.9 shows one of the images being labelled and annotated with different bounding box colors corresponds to different classes.

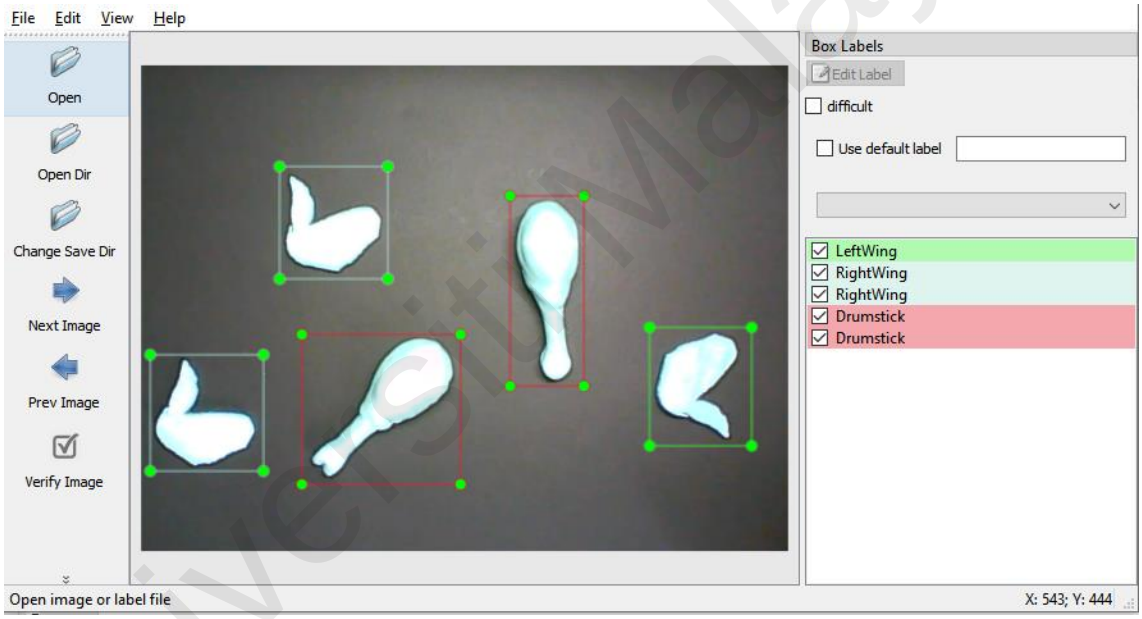

Figure 3.9: LabelImg labeling and annotation sample

Each labelled image is then saved using the YOLO format in a txt file that contains the class and the bounding box coordinates of each object in the image as shown in Figure. The txt file is formatted as object-class, x, y, width, and height where:

- Object-class is an integer number of objects from 0 to 2 in which, 0 is LeftWing, 1 is RightWing and 2 is Drumstick
- x, y, width, and height are float values relative to width and height of the images which can be equal from 0.0 to 1.0 (x and y are center of the rectangle)

- for example:
    - $x = absolute\ x\ /\ image\ width$
    - $height = absolute\ height / image\ height$

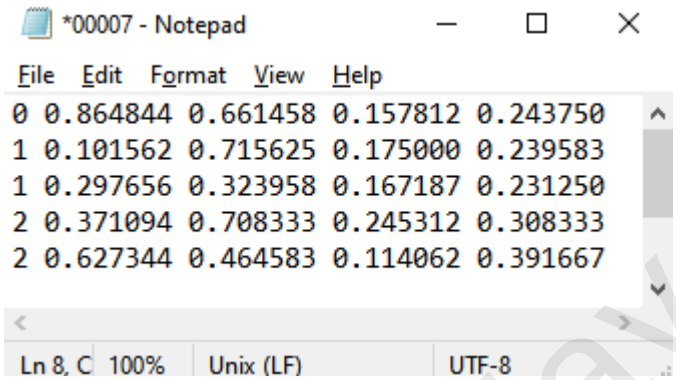

Figure 3.10: Text file containing the labels of an image

3.3.3 Data Augmentation

The 1000 labelled and annotated images are then augmented to obtain 4000 more images for the YOLO training. Data augmentation enables user to significantly increase the number and diversity of the data available for training models, without the need of gathering new data. There are multiple methods for data augmentations such as image rotation, adding noise, image flipping etc. In this research, only the image rotation method is used. An open source image augmentation code created by whynotw (2019) in GitHub is used. The rotation angles used for this image augmentations are 0º, 45º, 90º, 180º and 270º. This resulted in each original image is augmented to produce four new images and labels. Figure 3.11 shows a sample of the augmentation process of an image and its resultant 45º, 90º and 180º rotated image and labels. This augmentation process results in a total of 5000 images for YOLO training and validation. These 5000 images are then divided into 70% as training data and 30% as validation data for the training process.

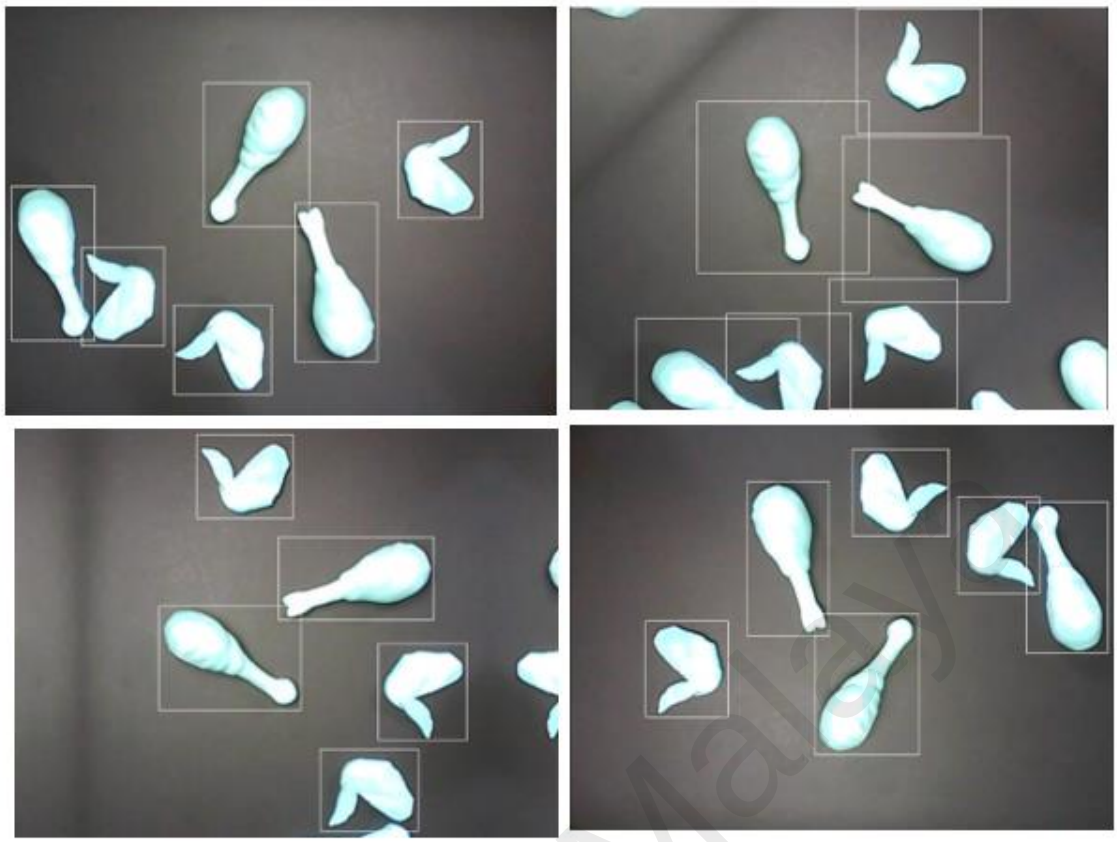

Figure 3.11: Image and label augmentation through rotation (original 0º, 45º, 90º and 180º in order from left to right

3.3.4 YOLO Training and Validation

YOLO training is done using a modified YOLOv4 training tutorial by theAIGuysCode (2020) in GitHub through Google Colab using its free graphic processing unit (GPU). In the training configurations file, 'yolov4-tiny-custom.cfg' shown in Appendix B, several parameters are modified to match the research. The batch size and subdivision of 64 and 32 was used to prevent error from GPU limitations. The image width and height are set to 416x416 to enable faster training durations. Other variables are configured as below:

- max_batches = (# of classes) *2000 = 3*2000 = 6000
- steps = (80% of max_batches), (90% of max_batches) = (4800), (5400)
- filters = (# of classes + 5) *3 = (3+5) *3 = 4
- classes = 3

A pre-trained weight for the convolutional layers created by AlexeyAB (2020) called "yolov4-tiny.conv.29" is used to help trained the custom object detector. This pre-trained weight helps this custom detector to be more accurate and not have to train as long. The training is executed until the loss is converged to the lowest average loss. For this research, the training process continued until about 6000 iterations and the lowest average loss of about 0.2 with a minimum average precision (mAP) of 99.63%. During the training, the resultant weights file was saved for every 1000 iteration as a backup and also to find the best weights file. More iterations do not mean greater precision but sometimes can lead to overfitted detection. Figure 3.12 shows a chart of both the loss and mAP for the training process. This chart shows that at about 5000 iterations, the mAP remains

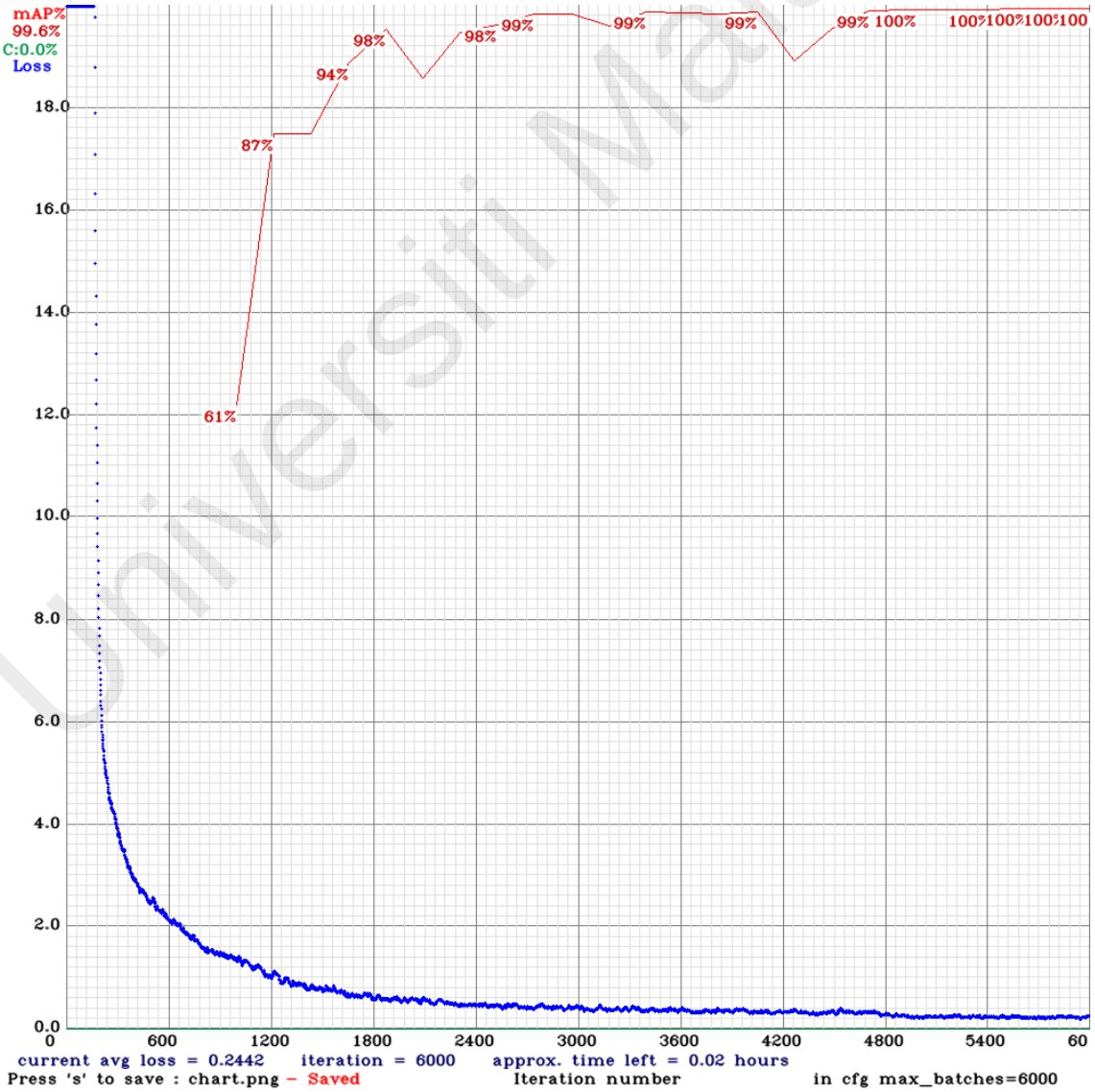

Figure 3.12: YOLO training and validation chart

relatively constant, thus the best suited weight for detection is the 5000th iteration weight file.

3.3.5 Testing

The resultant weight file is tested using several images and live video input from the webcam using a modified python code by Hassan (2020) to suit the new YOLO model, with the same equipment setup. When testing, the objects are placed randomly and not close together similar to the images for training. This testing is just to observe whether the weights file obtain from the training process can be used to detect the wings and drumsticks from other images and the live input video from the webcam. Other case study

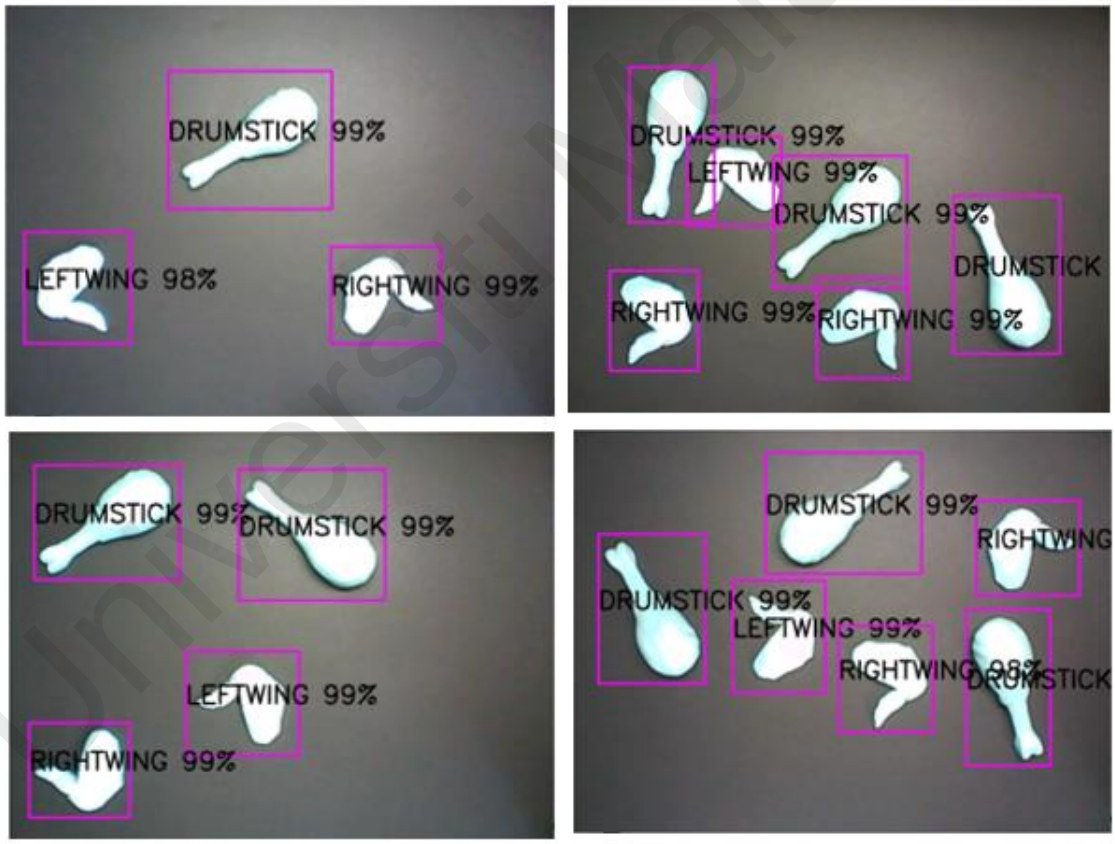

Figure 3.13: YOLO object detection testing results

that involves object stacked onto each other and placed close together will be shown in CHAPTER 4:Experiments and Discussion. Figure 3.13 shows the result obtained when using the detection with the weight file obtained from the training process. These images

show the bounding boxes that surround each object that contains the class type and the detection confidence for each object that is detected.

3.4 Contour/Orientation Detection

3.4.1 Contour Detection

The program for contour detection is based on a OpenCV shape detection tutorial by Hassan (2019). In the tutorial, the shapes are identified by using edge detectors which detects the outer perimeter of each object. The edge detector used is called the "Canny Edge detector" which is an operator that uses a multi-stage algorithm to detect a wide range of edges in images. This operator was developed by John F. Canny in 1986. The Canny edge detector used in this project is the one available in the OpenCV library. Before running the edge detector algorithm, a few preparations need to be done to the image to avoid errors in the detection. The edge detection is highly sensitive to image noises, which is why noise reduction is required. Noise reduction in the image is done through a blurring/smoothing process available in OpenCV called "GaussianBlur". This smoothing process prevents unnecessary edges to be detected. Figure 3.14 shows the

Figure 3.14: Original image and blurred image in order from the left

original image and the Gaussian blurred image that have reduced noise level.

The blurred image is then converted to a grayscale format through the use of a OpenCV function, COLOR_BGR2GRAY. This function cenverts any image to a single channel

grayscale image with every pixel is assigned a value from 0-255 to represent its intesity. By converting the image to grayscale, a thresholding process can be done for the Canny edge detection. These thresholds converts the image to black and white, which highlight the object of interest for the edge detection algorithm. Figure 3.15 shows the grayscaled image and the resulted edge detection image after the thresholds are applied. Thresholding

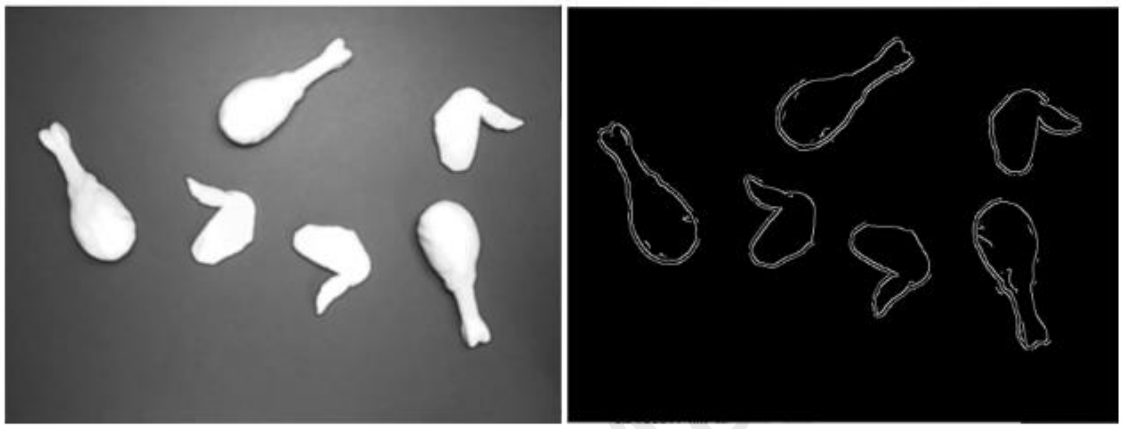

Figure 3.15: Gray scaled image and the resulted image after Canny edge detection.

converts the borders/perimeter of the object in the image completely white, with all the pixels having the same intensity and the rest of the pixels, the background into black. The Canny edge detection is use along with these thresholds to detect the borders of these

Figure 3.16: Trackbars for Canny edge detection algorithm containing the thresholds and minimum area

white pixels. The threshold values is tested by using trackbars in the code to find the suitable values for the edge detector. Other than the thresholds, the minimum area for the contour is also applied in the trackbars. This is so that only contours that have area greater than the specified minimum area to be generated which prevents any unwanted contour

to be displayed. Figure 3.16 shows the values obtained for the thresholds and minimum area for the Canny edge detection algorthm. The contours obtained from these Canny

Figure 3.17: Dilated image showing thicker contour lines

edge detection is then dilated to obrained a more thicker contour lines as shown in Figure 3.17. Figure 3.18 shows the resulted dilated contour overlayed on the original image with number of points and the contour area information. The contours make up of points that

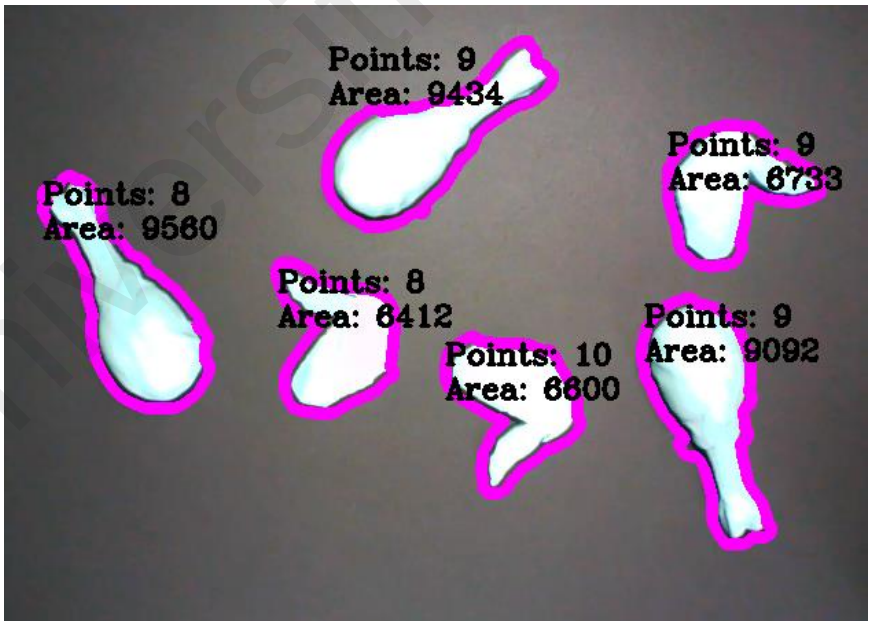

Figure 3.18: Image of contour lines overlayed onto original image

are connected together to form the edges. These points obtained from the detected edges/contours is then used for the orientation detection process.

3.4.2 Orientation Detection

The orientation of the contours/objects are determined by using a function available in the OpenCV library called “fitEllipse”. This function takes in the contours which contains several sets of 2D coordinates that represent each object and fits an ellipse on the contour (OpenCV, 2021). This function outputs several parameters that contains the center of the ellipse, the major and minor diameter of the ellipse, and the angle of the major diameter. The angle of the major diameter is measured from a vertical axis towards the major diameter in a clockwise (CW) direction. Figure 3.19 shows the resultant angle and ellipse

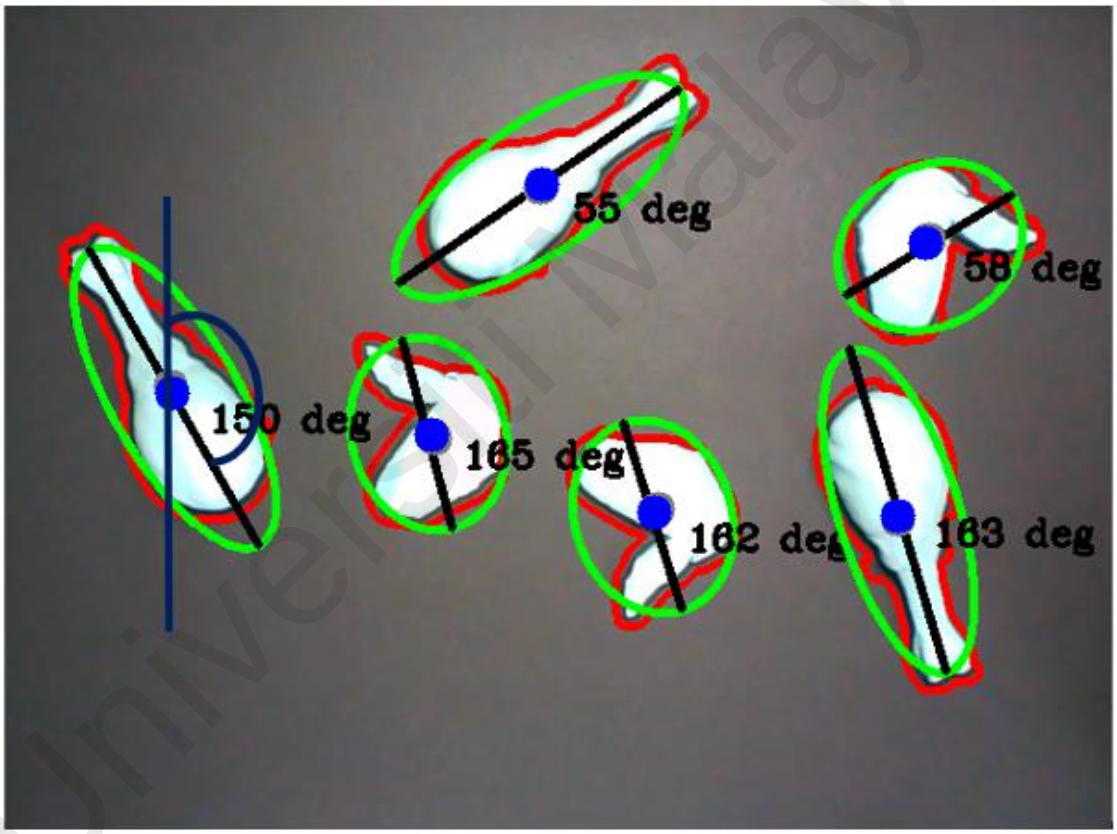

Figure 3.19: Result of fitting an ellipse on the contours using fitEllipse function

after using the fitEllipse function on the contours. The navy-colored lines on the drumstick on the left side of the figure are showing how the angle is measured which applies to the other objects as well. The angle is measured from vertical axis to major diameter, however, it is not consistent with the shape of the object, i.e., for the drumsticks

in Figure 3.19, the angle 150º is measured from the vertical axis to the meat side while the 55º is measured to the bone side.

In order to obtain a default and consistent angle of orientation, all the contours are rotated counterclockwise (CCW) of the angles obtained from the fitEllipse function as shown in Figure 3.20. This would make all the rotated contours to have a zero-degree

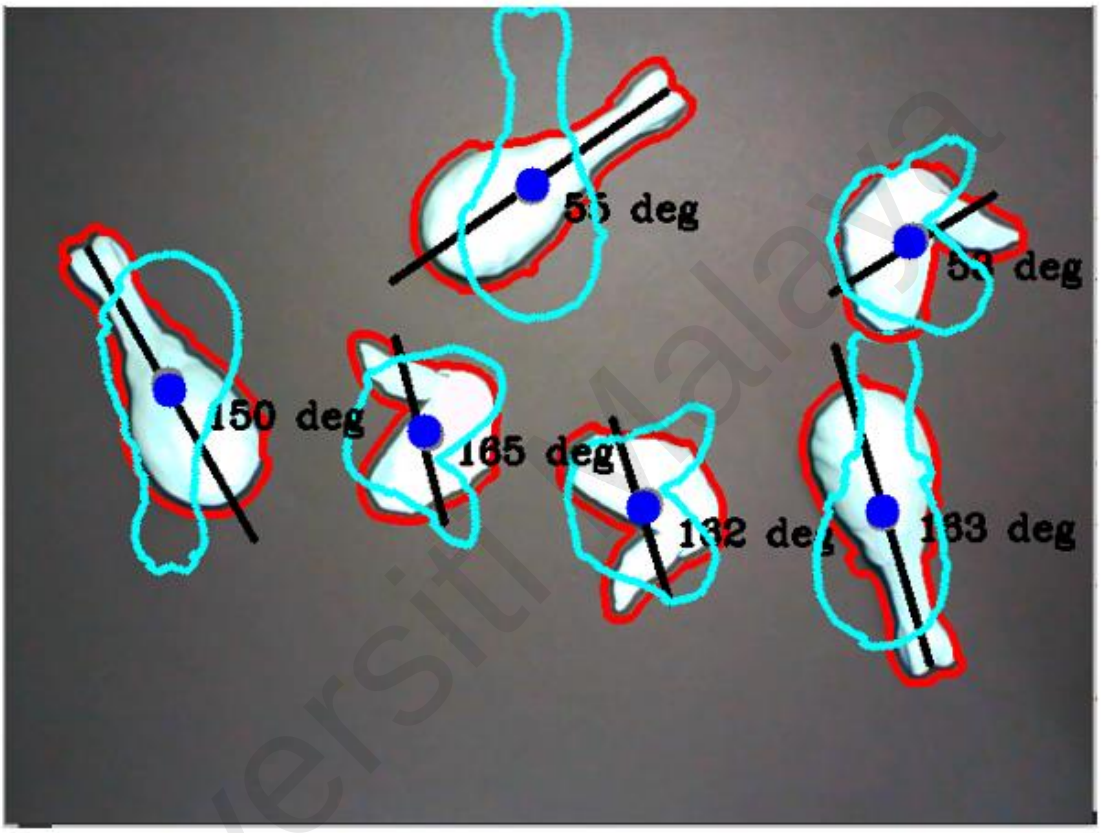

Figure 3.20: Contour in red is rotated CCW of the angles displayed to produce contours in cyan

rotation. The contours in teal are displaying the rotated contour and as observed, the drumsticks rotated contours does not match since the way the angle measured is different as mentioned above.

To correct the issue of the inconsistent angle measured, the fitEllipse function is used on the rotated contour (teal colored) to obtain the center points of the ellipse. The center points of the ellipse are then compared with the center points of the rotated contour. If the y-value of the center of the contour is greater than the y-value of the ellipse, then the

contour will be rotated clockwise of 180° and if it is not greater, then the contour will not be rotated. Figure 3.21 shows the final contour in orange color and the blue circle is the

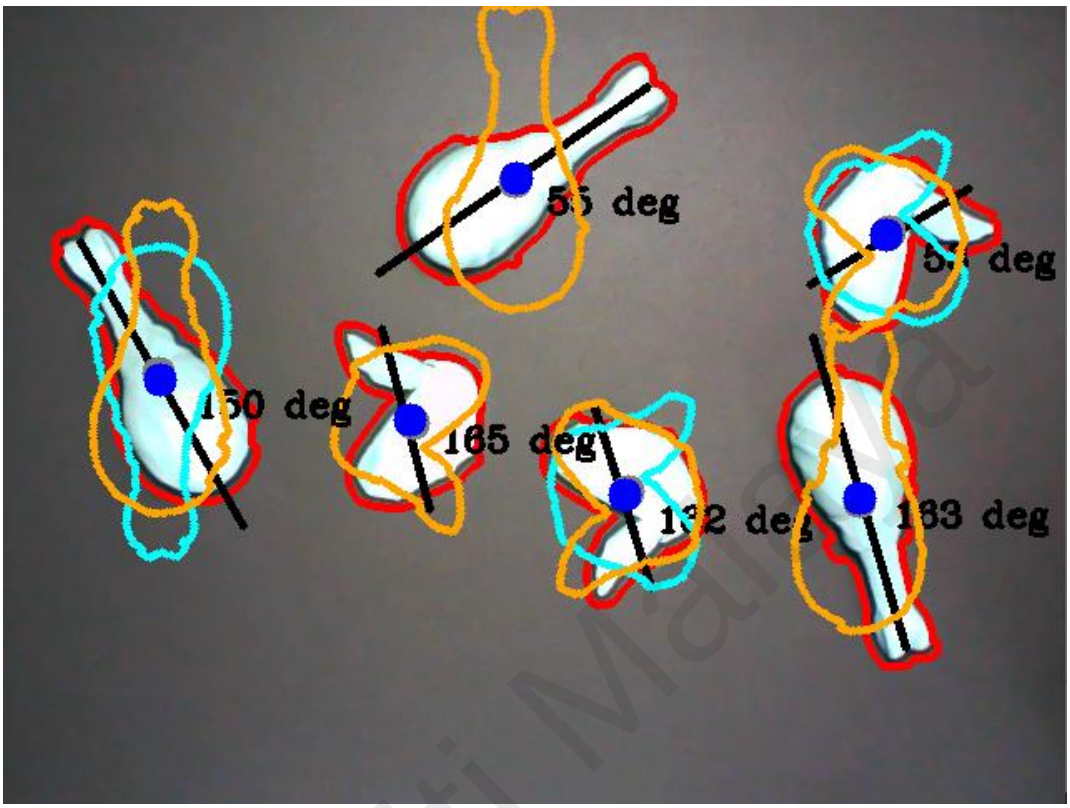

Figure 3.21: Contour in red is the original, contour in teal is rotated using the angles displayed and contour in orange is the final contour

center of the contour in teal and gray circle is the center of the ellipse fitted to the teal contour. As observed, all the orange contour of the drumstick match with each other. All the value of the angles used for each rotation are added together (negative angles for CCW rotations) to be used as the object's orientation information.

To summarize the contour/orientation detection section, a flowchart showing the steps for the orientation detection program is shown in Figure 3.22. The angles of rotation is added together, i.e., for the drumstick on the left in Figure 3.21, the final angle would $-150^o + 180^o = 30^o$.

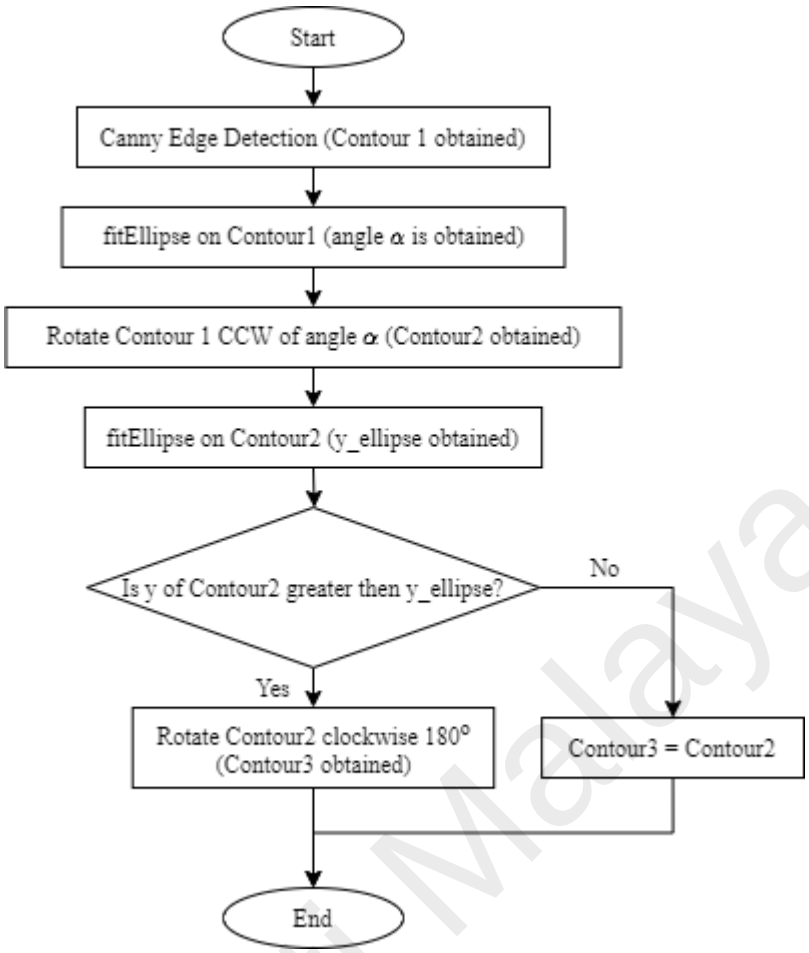

Figure 3.22: Flowchart of the contour/orientation detection program

3.5 Combining YOLO and Orientation Detections

From the YOLO detection program, each object in the image will be bounded by a box, in which the coordinates and dimension of the box can be obtained. Each object in the bounding box is then used along with the orientation detection program to determine their orientation. Using the final angle obtained from the orientation detection and the class information from the YOLO object detection, the suitable picking orientation for each object is calculated. The added angle of rotation for each wing and drumstick are different in which the LeftWing is rotated 45° CW, RightWing is rotated 45° CCW and drumstick is rotated 90° CW. Figure 3.23 showing the sample result of combining the two detections programs for the drumstick on the left of the figure. The image is cropped to only the drumstick in the bounding box and it is run through the orientation detection

program that displays the contours and calculates the angle for pick and place procedure in RoboDK. The same process is repeated for other object in their respective bounding

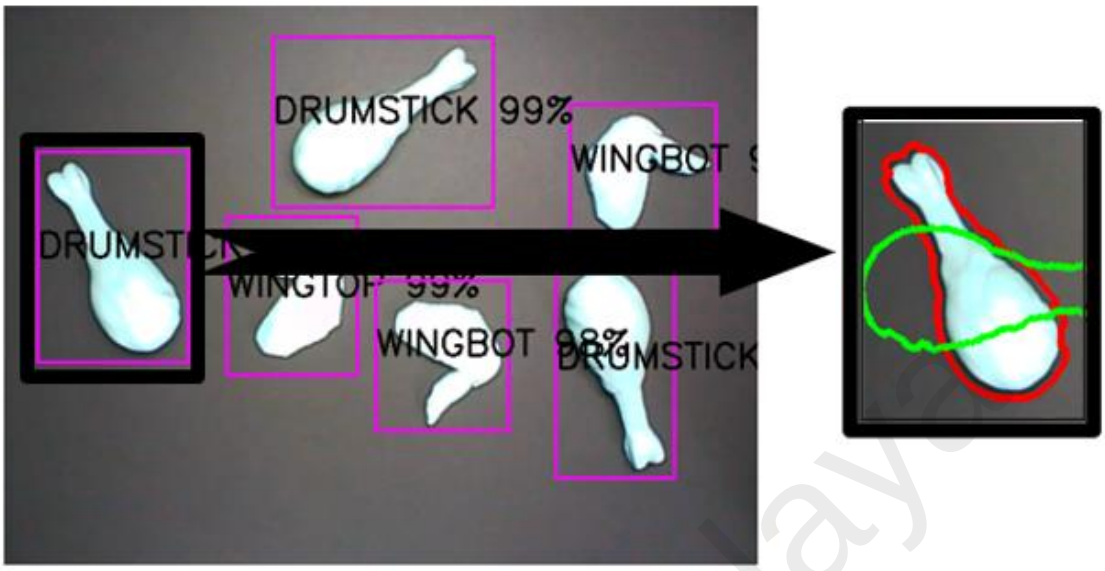

Figure 3.23: Sample image demonstrating the combination of the YOLO object detector and orientation detection program

boxes until all the orientations of each object is obtained. The green contour shows the final orientation of the drumstick to be placed by the robot arm onto the pallet. Figure 3.24 shows the picking angle for the robot gripper (negative of the angle needed to rotate the objects to the desired final orientation), the blue circle represents the centroid of the objects and the green contour shows the final orientation desired when placing the objects on the pallet in the RoboDK simulation.

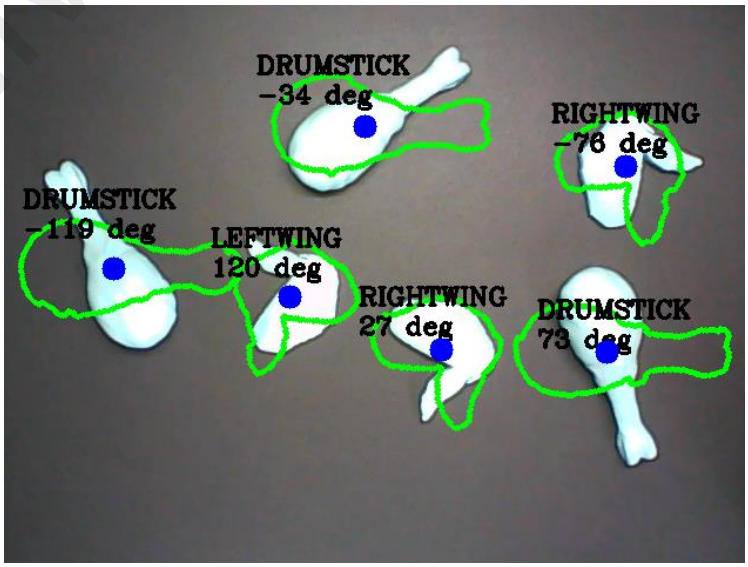

Figure 3.24: Image showing the result of the detector program

3.6 RoboDK Simulation

In order to simulate the pick and place procedure in RoboDK, a virtual environment need to be set up. The virtual environment would include a robot arm, gripper, tables etc. As mentioned previously that the 640 x 480 resolution of the images, translates to a 640 mm by 480 mm physical cartesian coordinates, thus the table used should be bigger than the specified length and width. All the virtual equipment is obtained in the RoboDK library except for the 3D model of drumstick and wings. Figure 3.25 shows the full virtual

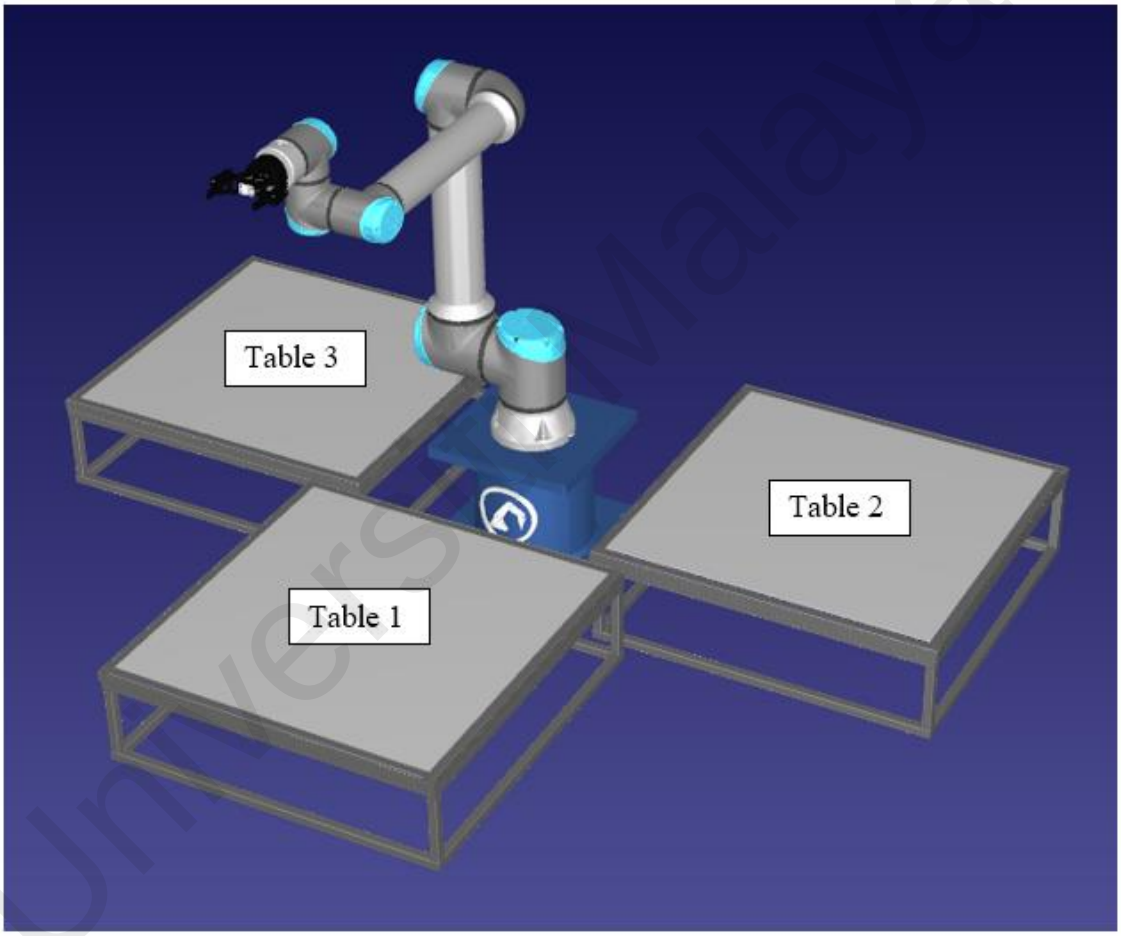

Figure 3.25: Virtual environment for pick and place in RoboDK

environment used fot the pick and place simulation in RoboDK. Table 1 is for picking operation and the other tables are for placing proceedures. The Robot used in this simulation is the UR10e from Universal Robots and its specification is shown in Table 3.2. This robot is chosen as it have a reach of 1300 mm which is barely enough to pick and pllace between the tables. As this is just for simulation, the payload of the robot which

exceeds the weight of the objects to be picked up by a large amount is ignored. The robot is merely chosen for its reachability. The robot is placed on top of a pedestal so that its base is higher than the table.

Table 3.2: Robot specifications (RoboDK, 2021a)

Brand	Universal Robots
Type	6 DOF
Axes	6
Payload	10 kg
Reach	1300 mm
Repeatability	0.050 mm
Weight	29 kg

The gripper used in this simulation is the Gripper RobotiQ 2F 85 (open). This gripped cannot open or close in the simulation, it is there to show that the gripper can use the data obtain from the detection programs to find and pick the object. As this is using the free version of RoboDK, the accessibility is limited, thus, the most suitable method is chosen to show the workings of the smiulation. Figure 3.26 shows the close-up image of the gripper used in the simulation. The way to pick and place the object using the gripper is by using the build in command “gripper.AttachClosest” and “gripper.DetachAll”. This does not show that the gripper will grip the object but merely attaching the object to the gripper.

Figure 3.26: Close-up image of gripper

3.6.1 Reference Frames

For the pick and place simulation to work, each object would need reference frame. All the equipment used from the library already have their own reference frame. Thus, only the reference frame of the 3D model of wings and drumstick are needed. The 3D models are first imported into the RoboDK environment, and their centroid is placed similar to the centroid obtained in the orientation detection. However, the centroid in orientation detection only contains 2D data, thus the z-coordinates is approximated. Figure showing the reference frame for each object in which the red is the x-direction, green is y-direction and blue is z-direction.

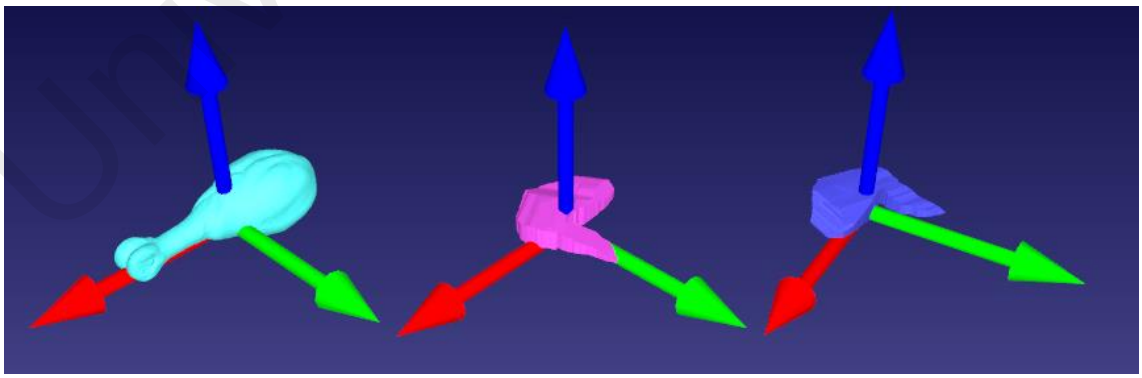

Figure 3.27: Reference frames for the drumstick, left wing and right wing in order from the left

For placing operations, a target reference frame needs to be added. The targets in Table 2 would be for placing the drumstick and targets in Table 3 would be for both left and right wing. The targets are placed using a python code developed in RoboDK (see

Appendix C: placeTargetsGenerator.py) using a txt file that contains the class type, coordinates, and orientation of the placing targets. The targets are placed within the 640 mm by 480 mm on the table with Table 3 is divided in half for each wing. Figure 3.27 shows the position and orientation of the targets to be used for the placing operations. The

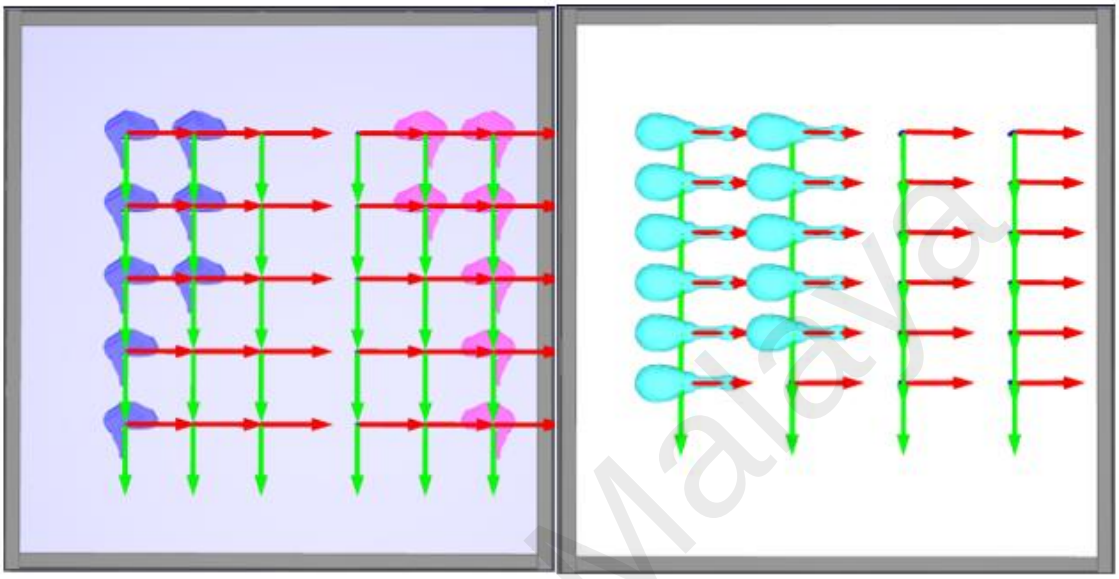

Figure 3.28: Targets for the placing operations on Table 3 and Table 2 in order from the left

objects shown are for demostrating the placing orientation, only the reference frame is generated for the targets.

3.6.2 Pick and Place Operations

Before beginning the pick and place procedure, a txt file containing the class type, coordinates and orientation need to be generated. This txt file is generated from the YOLO and orientation detector program. The txt file will contain the x, y, z coordinates and the orientation (rotation about x, rotation about y and rotation about z). The rotation about z is obtain from the orientation detection while the rotation about x and y are constant for the objects as they are laid flat on a surface. The z-coordinates depends on the thickness of the objects. The program containing the YOLO and orientation detection that generates this txt file is available in Appendix D: YOLOv4-Orientation-multipleImages.py. Figure

3.29 shows the txt file generated that contains the class type, x, y, z coordinates and the rotation about x, y, and z.

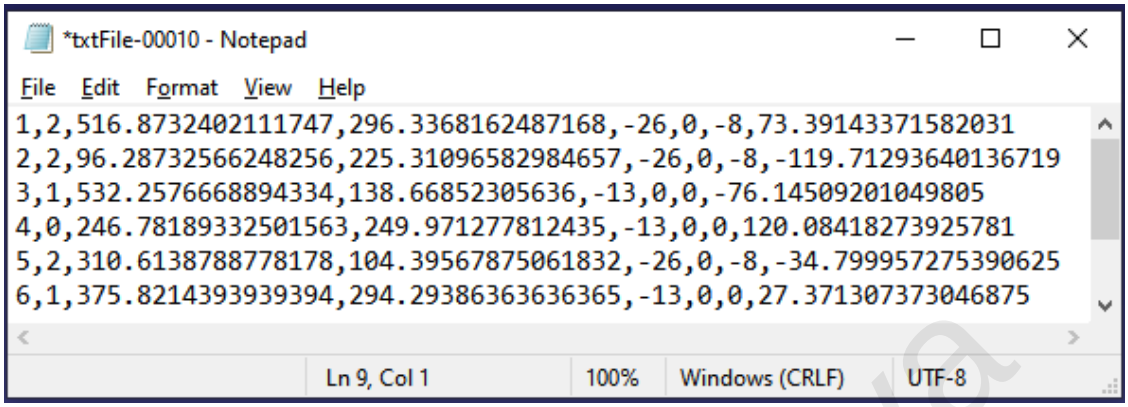

Figure 3.29: Sample txt file generated from the detection program

This generated txt file will be used alongside a python code in RoboDK to generate virtual objects to be picked and placed in RoboDK (see Appendix E: createObj-Pick-Place.py). This code will copy the 3D model of drumstick and wing, to be generated in a location and orientation similar to the image used in the detection program. Figure 3.30

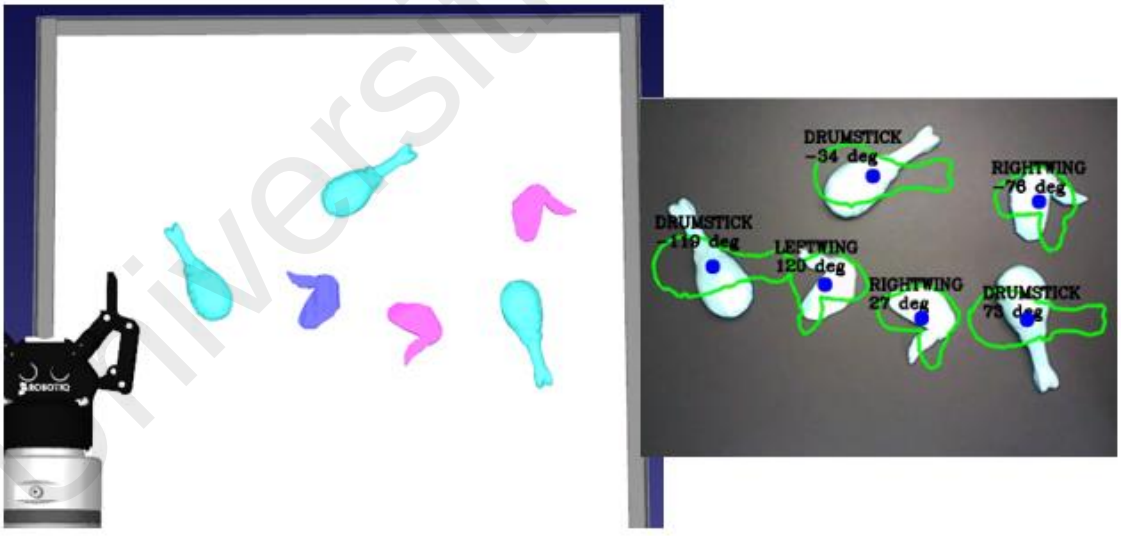

Figure 3.30: Virtual model of the objects generated in RoboDK and the image from the detector programs

shows the generated virtual model of the objects in RoboDK from the image used in the detector program. To differentiate between the models, colors are applied to the virtual models, blue is for left wing and pink is for right wing. After all the virtual models have been generated, the pick and place operation begin. Figure 3.31 shows some sample

images of the gripper picking up the objects. For the drumstick, as the gripper claw is shorter than the length of drumstick, it seems like it is poked through the drumstick. As mentioned previously, the gripper cannot be opened or closed as this is the limitation of the free version of RoboDK. However, the images prove that the data obtained from the detector programs can be used as the picking information for the gripper. The objects are just attached virtually to the gripper using the command "gripper.AttachClosest" and will release the object using "gripper.DetachAll". These images shows that the location and orientation of the gripper match with the object that it's going to pick up. Thus, if another bigger gripper is used, the orientation can still match.

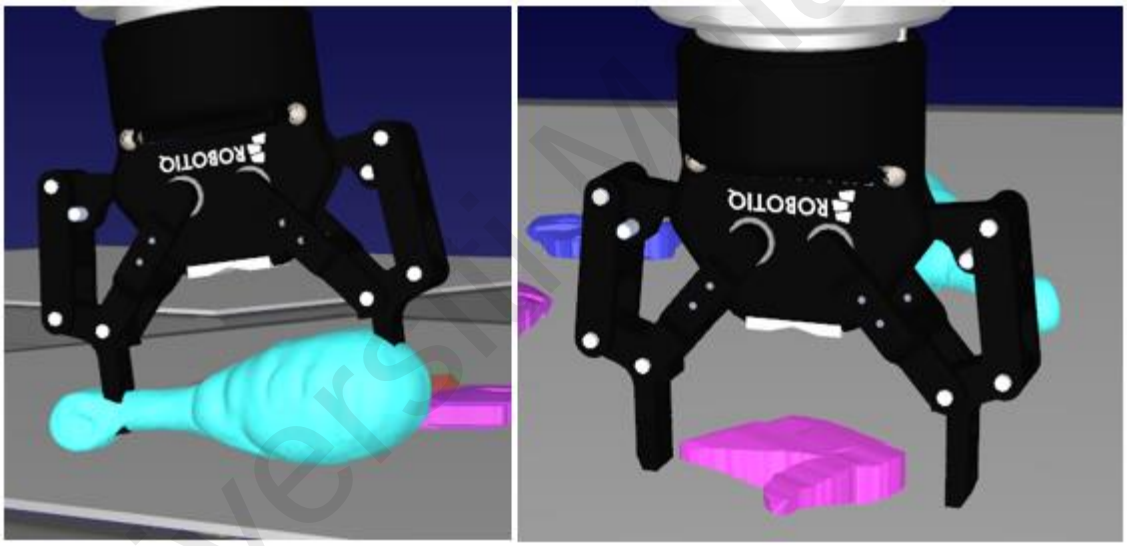

Figure 3.31: Gripper picking sample images

In summary, the images will be run through the detector programs to generate txt file that contains the class type, locations, and orientations of the objects. Then, that information is used in RoboDK to generate the virtual object models on Table 1 to be picked up by the robot and gripper and placed to Table 2 if it's a drumstick, and to Table 3 if it's a wing. Figure 3.32 shows an image of the ongoing pick and place operations in RoboDK.

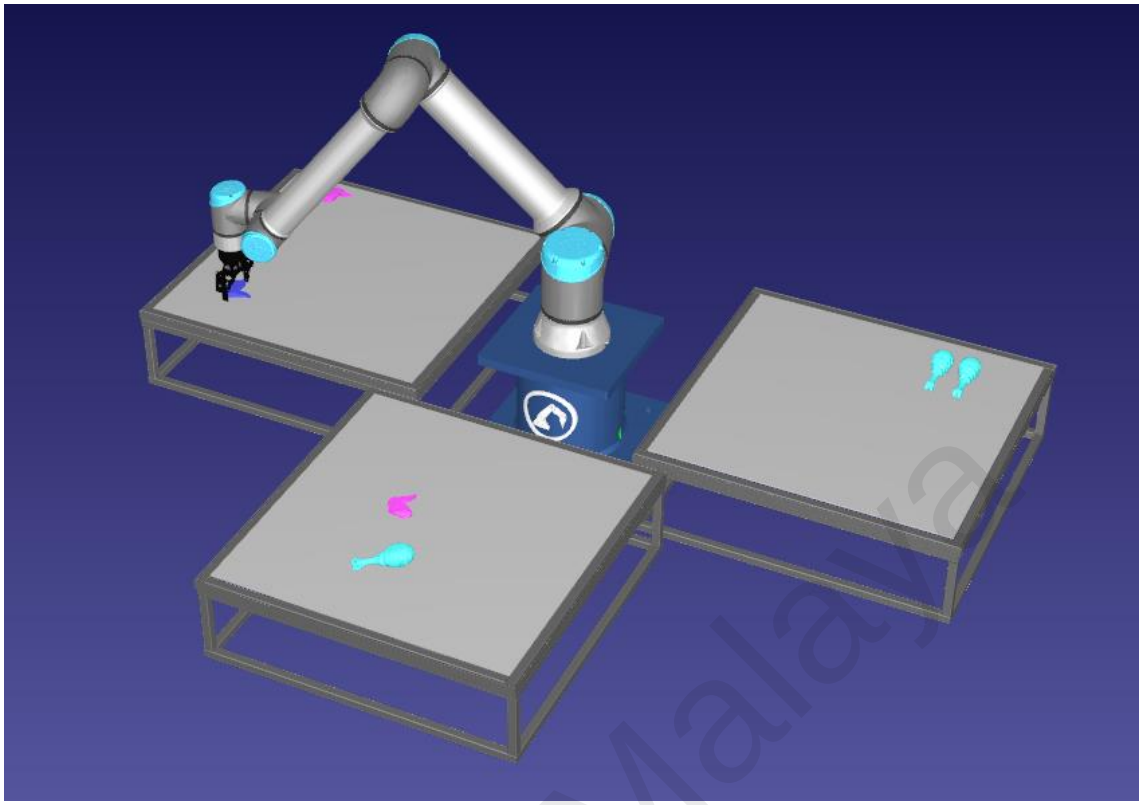

Figure 3.32: Image of pick and place operation in RoboDK

CHAPTER 4: EXPERIMENTS AND DISCUSSION

Using the programs developed, several case studies are conducted to test the performance of the vision system.

4.1 Ideal Case

For the ideal case, 10 images are acquired for the detector programs. In each image, the objects are fully within the 640 x 480 resolution meaning that 100% of the object's body are in the frame. In addition, the objects are not closely packed together and have at least 30 mm separation between them. This also means that there are no objects that are stacked or overlapped onto each other. This is so that there is no problem in object detection and also the contour detection. Figure 4.1 shows the six images used for the

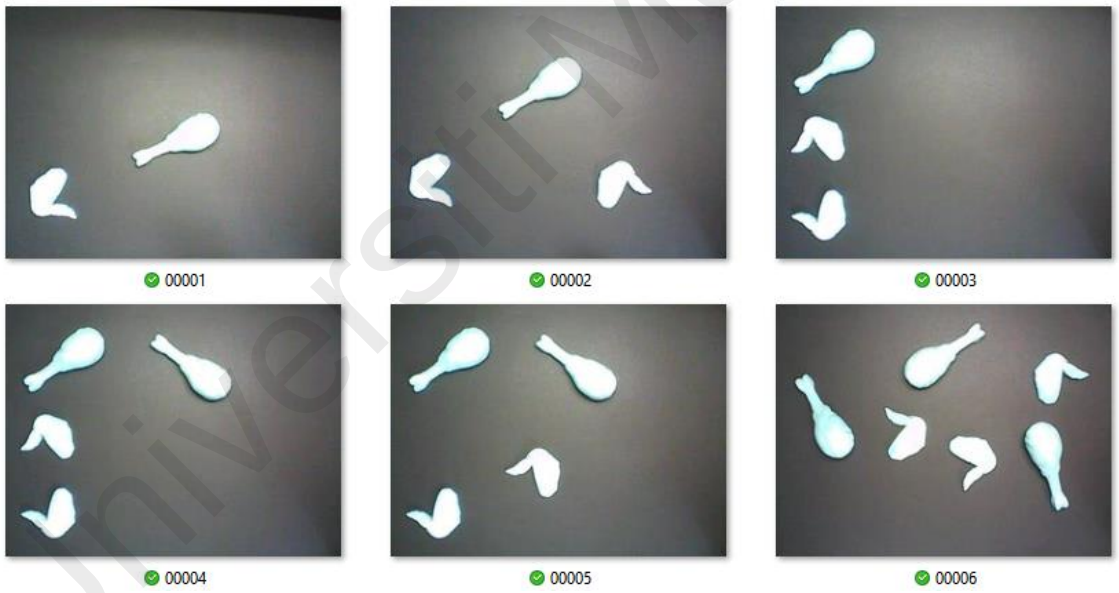

Figure 4.1: Images used for the ideal case

ideal case study. All of the objects in each image are not close to each other and 100% of their body is visible in the frame. Each image will represent one pallet that contains the objects. The total number of drumsticks are 10, left wings are six and right wings are six.

4.1.1 Detector Results and RoboDK Simulation

The images are ran through the detector programs to detect the type of objects and location as well as each object's orientation. Figure 4.2 shows the result after the six

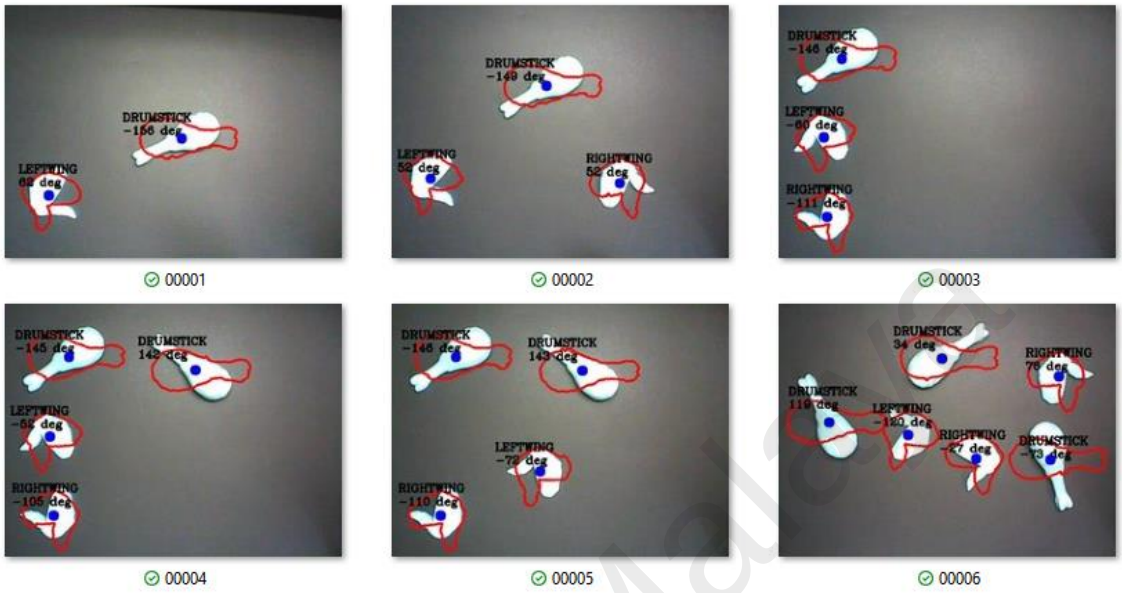

Figure 4.2: Resultant images from the detector program

images area ran through the detector program. In each image, the class type, picking orientations and the final placement contours are displayed. For each image, a txt file containing the class type, x, y, z coordinates and rotation about x, y, z is generated. Figure

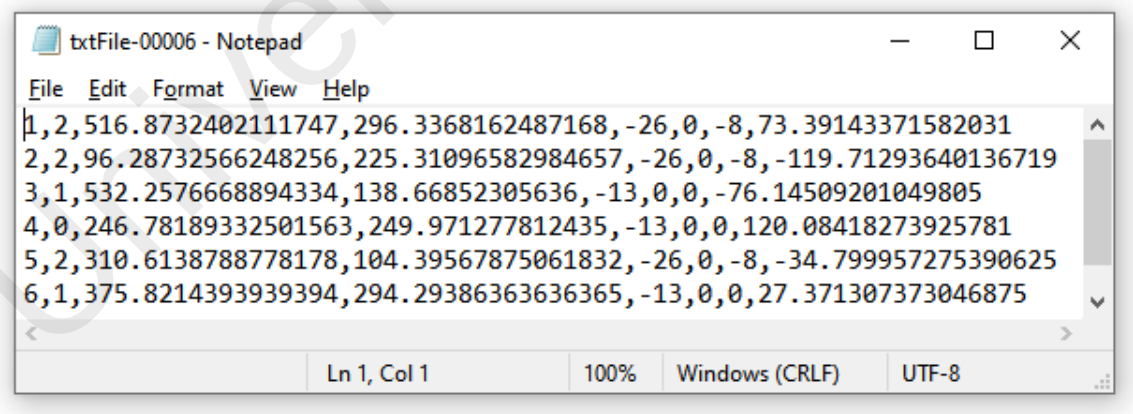

Figure 4.3: Txt file generated for image 00006

4.3 shows the txt file generated for image 00006 that contains the class, location, and orientation information to be passed through in RoboDK. In RoboDK, the pick and place are done through one image at a time. The objects in the first image are virtually generated, then they are picked and placed by the robot. After all the generated object

from first image is picked and placed, then, the objects in the second image are generated, picked and placed beside the first generated objects. The process is then repeated until all the objects in the last image have been generated, picked, and placed. Figure 4.4 shows

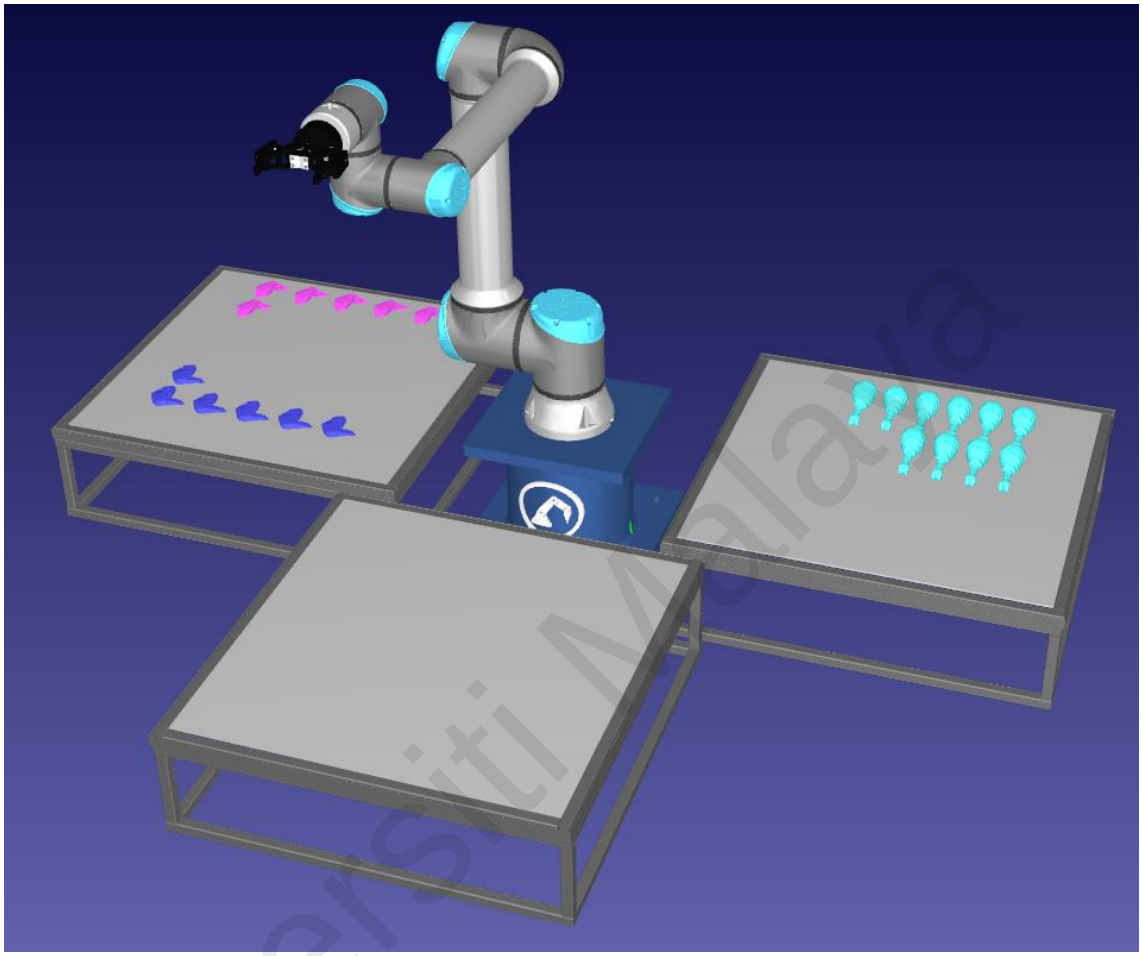

Figure 4.4: Image after all the objects in all images have been picked and placed in RoboDK

the images after the pick and place operation are done in RoboDK and the total number of drumstick placed are 10, left wings are six and right wings are also six. These numbers match with the number of objects in the six images used in the detector program which means that all objects have been accounted for.

For most of the case studies, the images gathered are first tested using the YOLO object detection due to the fact that the orientation detection rely on the bounding boxes generated by the object detector program.

4.2 Case Study 1: Objects close together

For this case study, a set of six images are gathered. In each image, the objects are placed close together with minimal gaps. These images are tested using the YOLOv4-tiny object detector first to determine whether this object detection program can detect every object on these images. Figure 4.5 shows images where all the objects are closely packed together with minimal gaps.

Figure 4.5: Images used for Case Study 1

These images are ran through the YOLO object detector program to test the detector's performance on objects that are closely arranged. Figure 4.6 shows the results obtained

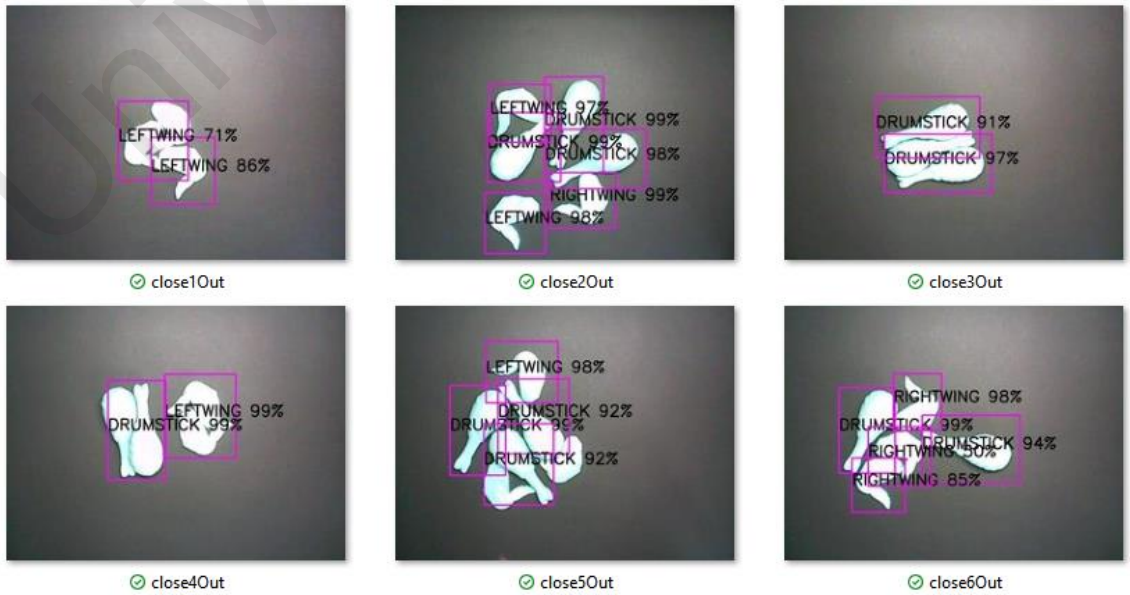

Figure 4.6: Results from YOLO object detection for Case Study 1

from the YOLO object detection program. For most of the images, the number of detection by the program are less than the number of objects. For example in image “close4out”, the program managed to only detect two objects out of the four, and in image “close3out”, it can only detect two out of the three drumsticks. And in some images, the wings are incorrectly detected, i.e., in image “close6out”, both the LeftWing are detected as RightWings. This is maybe due to the the initial images used for training. Most of the training images does not contain objects that are closely packed together. Since the orientation detection program uses the bounding boxes from the YOLO object detection, its result are also not reliable when the objects are placed close together. This is due to more than one object in the same bounding boxes, thus, creating more contours. One way to prevent the objects being closely together in a pallet is by applying a separator or by vibrating the pallet to loosen the arrangement.

4.3 Case Study 2: Object Overlapped

Six images containing several objects that overlapped or stacked onto each other are gathered. Figure 4.7 shows the images that contain multiple objects being stacked onto

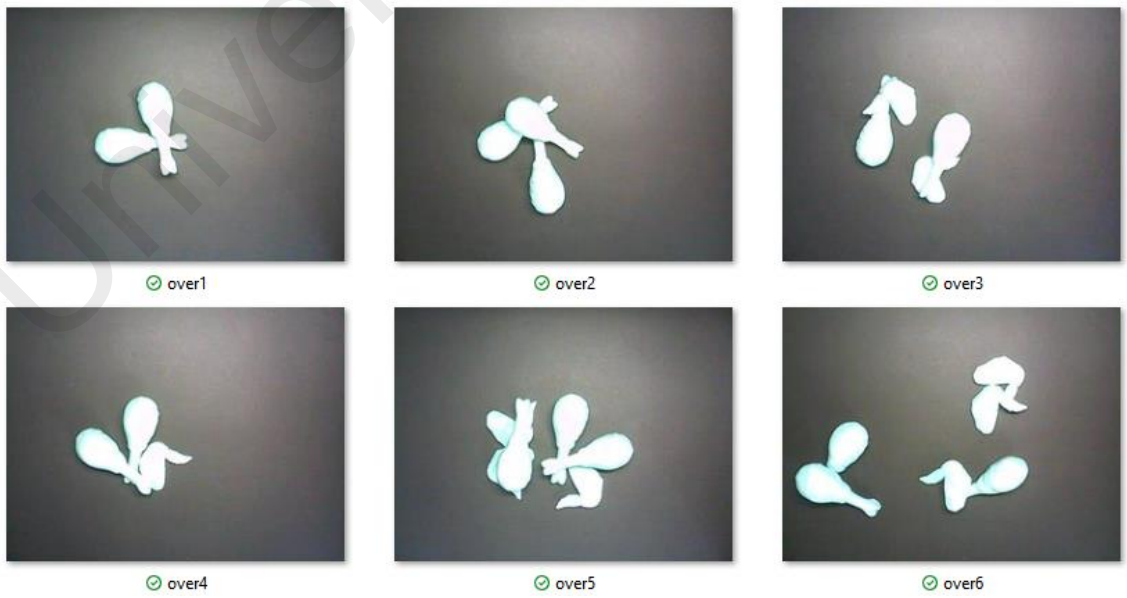

Figure 4.7: Images used for Case Study 2

one another. Smilar results as Case Study 1 are obtained. The number of detection are less than the number of object and several objects are detected with a wrong class type.

Figure 4.9 shows the results obtain from the YOLO object detection. Several objects are missing in the detection that is showed by the lower number of bounding boxes compared to the number of objects in each image. This is may be due to the objects having similar color which is why the objects seems indistinguishable.

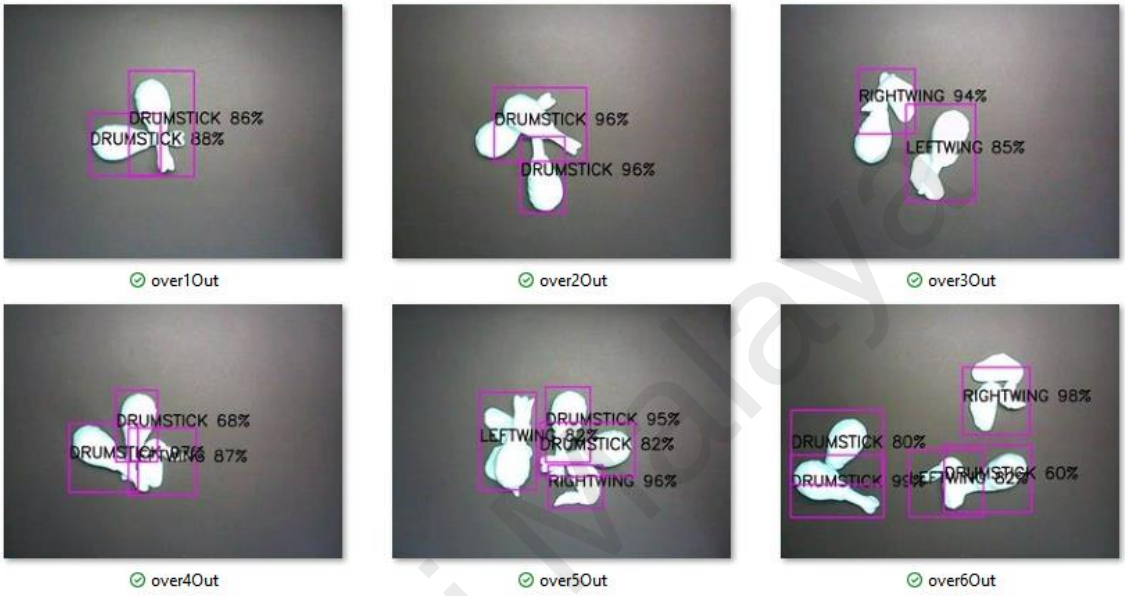

Figure 4.9: YOLO object detection result for Case Study 2

To demonstrate that the orientation detection program result depends on the bounding boxes of the object detection, image "over1" is tested using the orientation detector.

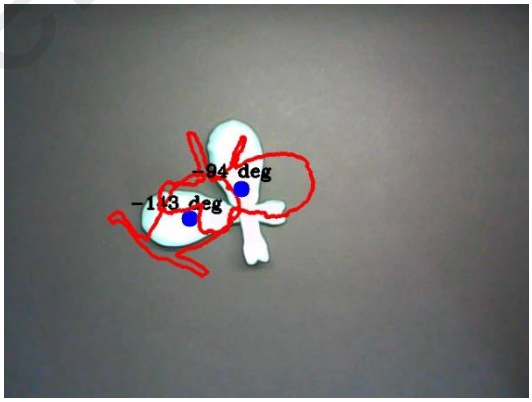

Figure 4.8: Orientation detector result for image "over1"

Figure 4.8 shows the result obtained from using the orientation detector on image "over1". The contour displayed (in red) does not have the shape of the drumstick, since there is more than one object in the bounding box. The angles showed in the images cannot

represent the orientation of each object since the contours are wrong. To prevent object from overlapping with one another, the pallet can be shaken or vibrated using a machine.

4.4 Case Study 3: Partially displayed object

For this case study, the objects are intentionally placed along the border of the platform, so that the image acquired will contain objects that its body is only partially displayed. Figure 4.10 shows the images that contains only partially visible objects.

Figure 4.10: Images used for Case Study 3

These images are first ran through the object detector to test whether the detector can detect object that are only partially visible. Figure 4.11 shows the result obtained from

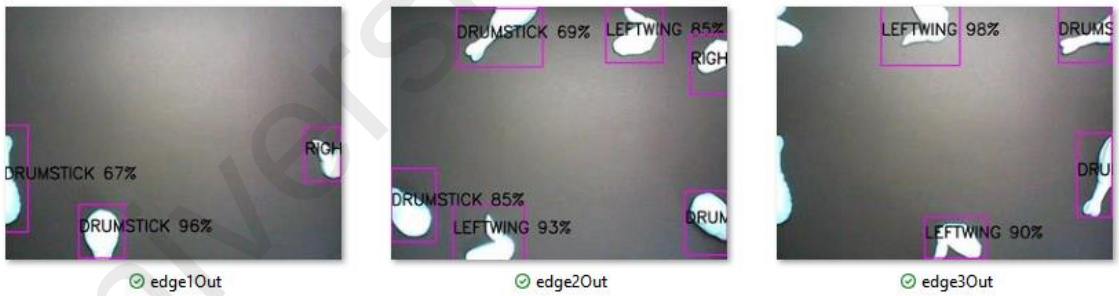

Figure 4.11: YOLO object detector result for Case Study 3

the YOLO object detector. For both images “edge1” and “edge2”, all the objects are detected correctly, but in image “edge3”, two objects were not detected. Then, the two images that their object are detected correctly are tested with the orientation detection program. Whent tested, the program resulted in an error, since it cannot detect the contours of the objects at the borders. The countour were not closed9 to which the detector

produce the error. To prevent the object to be over the edge of the pallet, a physical border could be place around the pallet so that all the objects can be 100% visible.

4.5 Case Study 4: Computer Performance

In this case study, the images from the ideal case were used. However, two different computers with different specifications are used to run the YOLO and orientation detector programs. The program will output all the resultant images and every txt file for each image. The main point of comparison would be the time taken for the program to finish computing. The computers used are a gaming laptop, Lenovo Legion Y530 and the Raspberry Pi 4 computer. The specifications for both computers are shown in Table 4.1 and Table 4.2.

Table 4.1: Specifications for Lenovo Legion Y530 gaming laptop

CPU	Intel(R) Core (TM) i7-8750H CPU @ 2.20GHz
RAM	16.0 GB DDR4 2666 MHz
Storage	ADATA SX8200PNP M.2 SSD

Table 4.2: Specifications for Raspberry Pi 4 Model B computer

CPU	64-bit Quad-Core Cortex-A72 processor @ 1.5GHz
RAM	4GB LPDDR4
Storage	SanDisk Ultra A1 Class 10 Micro SD Memory Card

Both the computers are tested with the same number of images containing the exact same objects. In the program, a delay of 3000 milliseconds will be added so that each detection result can be displayed and observed. In addition, each computer will also be tested when only the txt file is generated without any output images and delays. The objective of this

experiment is to observe the performance of the detection program on different computer systems of different specifications and prices in which the gaming laptop is much more expensive than the miniature computer. Table 4.3 shows the computation time for each computer to run the detection program with and without the delays. The difference in computation time between the two computers are about five seconds for six images. Thus, to have a seamless operation between the program and the simulation, a higher frequency CPU is needed or by modifying the program to have a lighter computational cost.

Table 4.3: Computation time results

Computer	Lenovo Gaming Laptop	Raspberry Pi 4
Computation time with delays and image generation	18.635288953781128 s	23.69989275932312 s
Computation time without delays and image generation	0.44082212448120117 s	5.044687271118164 s

In summary, this YOLO object detection and the orientation detection program can be applied to a robotic palletizing system through a vision system as long as the objects are arranged in a certain manner. The objects should not be overlapped with one another and should have a gap between them, so that the objects are not in contact with each other. To achieve such arrangement, a sorting or vibrating mechanism can be applied in the production line. The size of the pallets can be changed as the image resolution can be scaled to match the desired pallet sizes. In addition, the type of objects can be manipulated by training a new YOLO detection model and its picking orientation can be modified by manipulating the angles in the program.

CHAPTER 5: CONCLUSION

5.1 Summary

In this research project, an image-guided vision system has been developed and tested. A YOLO object detection model was trained using images acquired using a regular desktop webcam (Logitech C170). The object detection model managed to achieve minimum losses and a high average precision. This detection model is able to detect the type of object in the images with high confidence and also generate the bounding boxes that contain the objects' location. The orientation of the objects is detected using a contour detection function in OpenCV called "Canny Edge detection" and "fitEllipse" function. By combining these two functions, the suitable picking orientation and position are generated which would enable a robotic arm equipped with a gripper to pick up the objects.

The class type, coordinates and orientation of the objects are used to simulate a pick and place operation in RoboDK. The objects in the image are successfully generated virtually in the RoboDK environment with the exact location and orientation from the real image. The robotic arm equipped with the gripper manage to pick up the objects with the suitable picking orientation and successfully placed them in the correct pallets with the desired orientations and arrangements.

It is hoped that this image-guided vision system can be applied to real-life robotic applications to increase the productivity and efficiency of manufacturing lines.

5.2 Future Work

Currently, images are gathered and passed through the detection programs, however, in real manufacturing lines, a live visual input of the pallets/conveyer belts would be favorable. This information from the live input would be passed through to the robotic arm to pick the objects. Due to hardware and software limitations, the actual real-life

testing of the detection programs is not able to be conducted, only simulation is able to be done. Thus, to further test this vision system, a real-life testing rig would be built, and the performance of the detection models can be tested.

To apply to other objects, the YOLO object detection model can be retrained with other sets of images containing the objects desired. This would enable this vision system to be applied to any other palletizing operations and not just on chicken wings and drumsticks. And the orientation detection program can be modified to comply with any final placing orientation that is desired to match with the newly trained YOLO model.

REFERENCES

AlexeyAB. (2020). *darknet*. https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v4_pre/yolov4-tiny.conv.29

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-y. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*.

Dejan. (2019). *Arduino Mecanum Wheels Robot*. https://howtomechatronics.com/projects/arduino-mecanum-wheels-robot/

Fu, C., Jing, B., Huiyu, W., & Xiaoguang, X. (2019, 3-5 June 2019). Reducing the Energy Consumption of a Palletizing Robot through RobotStudio. 2019 Chinese Control And Decision Conference (CCDC),

Garbev, A., & Atanassov, A. (2020, 1-3 Oct. 2020). Comparative Analysis of RoboDK and Robot Operating System for Solving Diagnostics Tasks in Off-Line Programming. 2020 International Conference Automatics and Informatics (ICAI),

Guan, X., & Wang, J. (2011, 15-17 July 2011). Mechanical design and kinematic analysis of a new kind of palletizing robot. 2011 Second International Conference on Mechanic Automation and Control Engineering,

Hassan, M. (2019). *Real time Shape Detection using Contours [9] | OpenCV Python Tutorials for Beginners 2020*. https://www.youtube.com/watch?v=Fchzk1lDt7Q&t=490s

Hassan, M. (2020). *YOLO v3 EASY METHOD | OpenCV Python (2020)*. https://www.youtube.com/watch?v=GGeF_3QOHGE&t=4s

Ji, W., Zhao, D., Cheng, F., Xu, B., Zhang, Y., & Wang, J. (2012). Automatic recognition vision system guided for apple harvesting robot. *Computers & Electrical Engineering*, *38*(5), 1186-1195. https://doi.org/https://doi.org/10.1016/j.compeleceng.2011.11.005

Junos, M. H., Mohd Khairuddin, A. S., Thannirmalai, S., & Dahari, M. (2020). An optimized YOLO-based object detection model for crop harvesting system. *IET Image Processing*, *n/a*(n/a). https://doi.org/https://doi.org/10.1049/ipr2.12181

Junyeong, C., Byung-Kuk, S., Daeseon, L., Hanhoon, P., & Jong-Il, P. (2013, 24-26 Oct. 2013). RGB-D camera-based hand shape recognition for human-robot interaction. IEEE ISR 2013,

KISI. (n.d.). *CCD Cameras: An Overview*. https://www.getkisi.com/guides/ccd-camera

Li, T., Ma, Y., & Endoh, T. (2020). A Systematic Study of Tiny YOLO3 Inference: Toward Compact Brainware Processor With Less Memory and Logic Gate. *IEEE Access*, *8*, 142931-142955. https://doi.org/10.1109/ACCESS.2020.3013934

Lin, T. (2017). *LabelImg*. https://github.com/tzutalin/labelImg

Moura, F. M., & Silva, M. F. (2018, 25-27 April 2018). Application for automatic programming of palletizing robots. 2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC),

Nowak, J. (2011, 14 October 2011). *The Pros and Cons of Implementinfg a Robotic Mixed Pallet Building System*. Retrieved 15 December 2020 from https://www.bastiansolutions.com/blog/the-pros-and-cons-of-implementing-a-robotic-mixed-pallet-building-system/

OpenCV. (2021). *Creating Bounding rotated boxes and ellipse for contours*. https://docs.opencv.org/3.4/de/d62/tutorial_bounding_rotated_ellipses.html

Pagano, S., Russo, R., & Savino, S. (2020). A vision guided robotic system for flexible gluing process in the footwear industry. *Robotics and Computer-Integrated Manufacturing*, *65*, 101965. https://doi.org/https://doi.org/10.1016/j.rcim.2020.101965

palletizer.org. *Mixed Palletizing*. Retrieved 12 NOV from http://palletizer.org/mixed-palletizing/mixed-palletizing.html#:~:text=Mixed%20palletizing%20is%20the%20increasingly,sizes%20on%20a%20single%20pallet.&text=Mixed%20palletizing%20is%20a%20challenge,complex%20and%20integrated%20palletizing%20systems.

Phadtare, L. K., Kushalnagar, R. S., & Cahill, N. D. (2012, 9-9 Nov. 2012). Detecting hand-palm orientation and hand shapes for sign language gesture recognition using 3D images. 2012 Western New York Image Processing Workshop,

printable_models. (2019). *Fried Chicken Drumstick V1 3D Mode*. https://free3d.com/3d-model/fried-chicken-drumstick-v1--641412.html

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016, 27-30 June 2016). You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137-1149. https://doi.org/10.1109/TPAMI.2016.2577031

RoboDK. (2021a). *Robot Library*. https://robodk.com/library

RoboDK. (2021b). *Simulate Robot Applications - Program any Industrial Robot with One Simulation Environment*. https://robodk.com/

RobotWorx. (n.d.). *Industrial Robot History*. https://www.robots.com/articles/industrial-robot-history

theAIGuysCode. (2020). *YOLOv4-Cloud-Tutorial*. https://github.com/theAIGuysCode/YOLOv4-Cloud-Tutorial

Webster, R. J., & Brannon, A. S. (2002, 11-15 May 2002). The Electronic Ball Boy: a reactive visually guided mobile robot for the tennis court. Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292),

whynotw. (2019). *rotational-data-augmentation-yolo*. https://github.com/whynotw/rotational-data-augmentation-yolo

WIKIPEDIA. (2020a, 4 December 2020). *Kinect*. https://en.wikipedia.org/wiki/Kinect

WIKIPEDIA. (2020b, 18 August 2020). *Mecanum wheel*. https://en.wikipedia.org/wiki/Mecanum_wheel

Wurll, C. (2016, 21-22 June 2016). Mixed Case Palletizing with Industrial Robots. Proceedings of ISR 2016: 47st International Symposium on Robotics,

Xu, Y., Liu, Y., Hao, L., & Cheng, H. (2016, 6-10 June 2016). Design of palletizing algorithm based on palletizing robot workstation. 2016 IEEE International Conference on Real-time Computing and Robotics (RCAR),

Yang, X., Zhang, H., Cheng, T., Ni, X., Wu, C., Zong, H., Lu, H., Lu, Z., & Shen, Y. (2018, 24-27 Aug. 2018). An Omnidirectional and Movable Palletizing Robot based on Computer Vision Positing. 2018 IEEE International Conference on Intelligence and Safety for Robotics (ISR),

Yi, Z., Nianjun, L., & Jun, C. (2017). A automatic Matching Algorithm and Simulation for Irregular Cigarette Package Stacking. *Procedia Engineering*, *174*, 1235-1243. https://doi.org/https://doi.org/10.1016/j.proeng.2017.01.292

Yuan, J., & Wang, C. (2016, 6-10 June 2016). Innovative design of palletizing system for China's local industries. 2016 IEEE International Conference on Real-time Computing and Robotics (RCAR),

Zhang, B., & Skaar, S. B. (2009, 10-15 Oct. 2009). Robotic de-palletizing using uncalibrated vision and 3D laser-assisted image analysis. 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems,