Vision Based Quality inspection of medicine packages using 3 DOF SCARA Robot

A SENIOR DESIGN PROJECT REPORT

Submitted by

Anushree	**01FE20BAR007**
Kiran	**01FE20BAR020**
Vinayak	**01FE20BAR025**
Varun	**01FE20BAR030**
Keerti S M	**01FE20BAR034**

in the completion of Senior Design Project

Under the Guidance of:

Asst. Prof. Girish Karikatti

Bachelor of Engineering

IN

AUTOMATION AND ROBOTICS

KLE TECHNOLOGICAL UNIVERSITY, HUBBALLI.

DECEMBER 29, 2023

KLE TECHNOLOGICAL UNIVERSITY,

HUBBALLI - 580031

DEPARTMENT OF AUTOMATION AND ROBOTICS

Certified that the project work entitled "Vision Based Quality inspection of medicine packages using 3 DOF SCARA Robot" is a bonafied work carried out by Anushree(01FE20BAR007), Kiran T(01FE20BAR020), Vinayak K(01FE20BAR025), Varun Bhat(01FE20BAR030), Keerti S M(01FE20BAR034) in partial fulfillment for the award of degree of Bachelor Engineering in Automation and Robotics of the KLE Technological University, Hubballi during the year 2023-24.

The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the Bachelor of Engineering Degree.

Project Guide	Head of the Department
Girish Karikatti	Prof. A C Giriyapur

Name of the Examiners	Signature with Date
1.	
2.	

ACKNOWLEDGEMENT

We have put our sincere efforts in this project. However, it would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them.

We are highly indebted to our faculties their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We would like to express our gratitude towards all the members of the department of Automation and Robotics for their kind co-operation and encouragement which helped us in completion of this project.

We would like to express our special gratitude and thanks to Head of the Department, Prof. A C Giriyapur for giving us attention and time.

Our thanks and appreciations to all our classmates and people who have willingly helped us out with their abilities.

01FE20BAR020 01FE20BAR025 01FE20BAR034

01FE20BAR007 01FE20BAR030

ABSTRACT

Quality inspection of medicine box is a critical process in the pharmaceutical industry to ensure the safety, efficacy, and compliance of pharmaceutical products. It involves a systematic assessment of various aspects of the medicine packaging, such as labeling accuracy, packaging integrity, and adherence to regulatory requirements.

This project focuses on advancing pharmaceutical quality control through the implementation of a Vision-Based Quality Inspection system using a 3 DOF SCARA Robot. By integrating deep learning methodologies, the system aims to autonomously inspect and analyze medicine packages for quality assurance.

The project involves the creation of a dedicated dataset comprising diverse medicine box images to train a deep learning model. The trained model enables the robot to autonomously detect and categorize defects or anomalies in medicine packaging, ensuring compliance with stringent quality standards. This innovative approach not only seeks to enhance the efficiency and accuracy of quality inspection but also contributes to minimizing errors, optimizing production quality, and reducing the risk of faulty pharmaceutical products reaching consumers.

Ultimately, the project aspires to establish a robust framework for intelligent quality inspection in pharmaceutical manufacturing, leveraging the synergy between computer vision, deep learning, and robotic automation.

Keywords: SCARA Robot,3 DOF, Deep Learning, YOLOv5, SolidWorks, Raspberry pi.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Quality inspection, also known as quality control or quality assurance, is a process that involves systematically assessing products, services, or processes to ensure they meet predefined quality standards and specifications. It helps identify defects, deviations, and inconsistencies to maintain high-quality output, reduce operational costs, and build customer trust. Quality inspection can be applied at various stages of production or service delivery and is vital for ensuring consistency and safety while facilitating continuous improvement.

Quality inspection is used in various fields and industries to ensure that products, services, or processes meet certain standards and specifications. Some of the fields and industries where quality inspection is commonly employed include:

1. Manufacturing: Quality inspection is crucial in manufacturing industries such as automotive, electronics, aerospace, and consumer goods to check the quality of products on the production line, ensuring they meet safety and performance standards.
2. Construction: In construction, quality inspection is used to verify that buildings, infrastructure, and components meet building codes and safety standards.
3. Food and Beverage: The food industry relies on quality inspection to ensure the safety and quality of food products. This includes checking for contaminants, proper labeling, and adherence to food safety regulations.
4. Pharmaceuticals: In the pharmaceutical industry, quality inspection is used to verify the safety and efficacy of medications and ensure compliance with regulatory standards.
5. Healthcare: Quality inspection in healthcare involves the assessment of medical equipment, devices, and facilities to maintain patient safety and compliance with healthcare regulations.
   Textiles and Apparel: In the fashion and textile industry, quality inspection is used to assess the quality of fabrics, garments, and accessories to meet design and performance standards.

Quality inspection of medicine box is a critical process in the pharmaceutical industry to ensure the safety, efficacy, and compliance of pharmaceutical products. It involves a systematic assessment of various aspects of the medicine packaging, such as labeling accuracy, packaging integrity, and adherence to regulatory requirements. The goal is to guarantee that medicines are correctly packaged and labeled, reducing the risk of errors and ensuring the well-being of patients. This quality inspection process is essential to maintain the high standards of pharmaceutical products and to comply with stringent regulatory guidelines that govern the industry.

Quality inspection of medicine boxes is needed to ensure patient safety, regulatory compliance, and product efficacy in the pharmaceutical industry. It helps to:

1. **Patient Safety:** Ensures that medicines are accurately labelled and properly packaged, reducing the risk of medication errors and harm to patients.
2. **Regulatory Compliance:** Ensures adherence to strict regulatory requirements and standards set by health authorities, demonstrating a commitment to quality and safety.
3. **Product Efficacy:** Verifies that medicines are stored and protected in a way that maintains their effectiveness, preserving their therapeutic value.

Selected Approach for Visual Inspection:
Visual inspection using deep learning is considered one of the best methods for inspecting medicine boxes due to its high accuracy, automation, versatility, and ability to continuously learn and adapt. It reduces human error, offers consistency, and is easily scalable for high-speed production lines, making it an efficient and reliable solution. It can also minimize false positives and provide traceable data for regulatory compliance. However, it requires quality training data and regular maintenance to maintain its effectiveness.

Deep Learning:
Deep learning is utilized in the quality inspection of medicine boxes due to its proficiency in recognizing intricate visual patterns, adapting to diverse packaging designs, and automating the inspection process. With the ability to efficiently detect defects such as damaged packaging and incorrect labeling, deep learning enhances accuracy, reduces false positives, and ensures regulatory compliance in pharmaceutical manufacturing. Its data-driven decision-making, adaptability to changes, and high-speed capabilities make it a valuable technology for maintaining the integrity and safety of medicine packaging, ultimately contributing to enhanced quality control in the industry.

Figure 1.1 Example of sealed and non-sealed boxes.

CHAPTER 2

PROBLEM STATEMENT

Vision Based Quality inspection of medicine package integrity using 3 DOF SCARA Robot

2.1 Motivation

In the landscape of pharmaceutical manufacturing, where precision and quality assurance are non-negotiable, the conventional methods of inspecting medicine packages often fall short of ensuring flawless compliance with stringent standards. Recognizing the pivotal role of packaging integrity in preserving the efficacy and safety of pharmaceutical products, our motivation lies in pioneering a transformative solution – Vision-Based Quality Inspection employing a 3 DOF SCARA Robot. This cutting-edge approach harnesses the power of advanced computer vision algorithms and the dexterity of a 3 DOF SCARA Robot to revolutionize the quality control paradigm. By seamlessly amalgamating technology, our prototype aims to elevate the inspection process by meticulously scrutinizing medicine packages, swiftly identifying and categorizing potential defects or irregularities. This not only guarantees adherence to the highest quality standards but also eliminates the risk of human oversight in the inspection workflow. The envisioned system represents a paradigm shift in quality assurance methodologies, presenting an innovative pathway for pharmaceutical manufacturers to uphold their commitment to delivering medications that are not only effective but also free from any packaging-related issues. Through this initiative, we endeavor to instill a culture of technological innovation, precision, and social responsibility within the pharmaceutical manufacturing sector, fostering advancements that safeguard patient well-being and industry excellence.

The above statistics motivate us to improvise the existing conventional solutions and build a prototype that can help in pharmaceutical manufacturing with minimal or no errors, thus making a contribution towards the economic development and progress of our country and establishing ourselves as responsible engineers andcitizens of our motherland.

2.2 Goals and Objectives

i. To enhance Quality Assurance.
ii. To Automate Inspection Processes
iii. To Optimize Production Quality
iv. To Enhance Patient Safety

CHAPTER 3

LITERATURE SURVEY

An summary of the previously published works on a certain topic is known as a literature survey. The phrase can be used to describe an entire academic document or a specific portion of a book, essay, or other scholarly work. In any case, the goal of a literature review is to give the audience and the researcher/author a broad overview of the body of information that already exists on the subject at hand. A thorough literature assessment can guarantee that the right theoretical framework, research methodology, and/or research question have been selected.

3.1 Research related to problem definition

This aids in our comprehension of the solutions that are currently available on the market for the specified problem statement. It aids in our comprehension of the degree of complexity necessary to tackle the issue and the strategies for differentiating our approach.

3.1.1 Existing similar projects

i. Designing quality control system based on vision inspection in pharmaceutical product lines: Creating vision inspection-based quality control systems for pharmaceutical product lines: This work looks into the design of an intelligent quality control system for pharmaceutical companies' use in vision inspection. The objective of our suggested approach, which relies on template pattern matching and Radon transform, is to tally the quantity of pill blister cards on the product line[5].

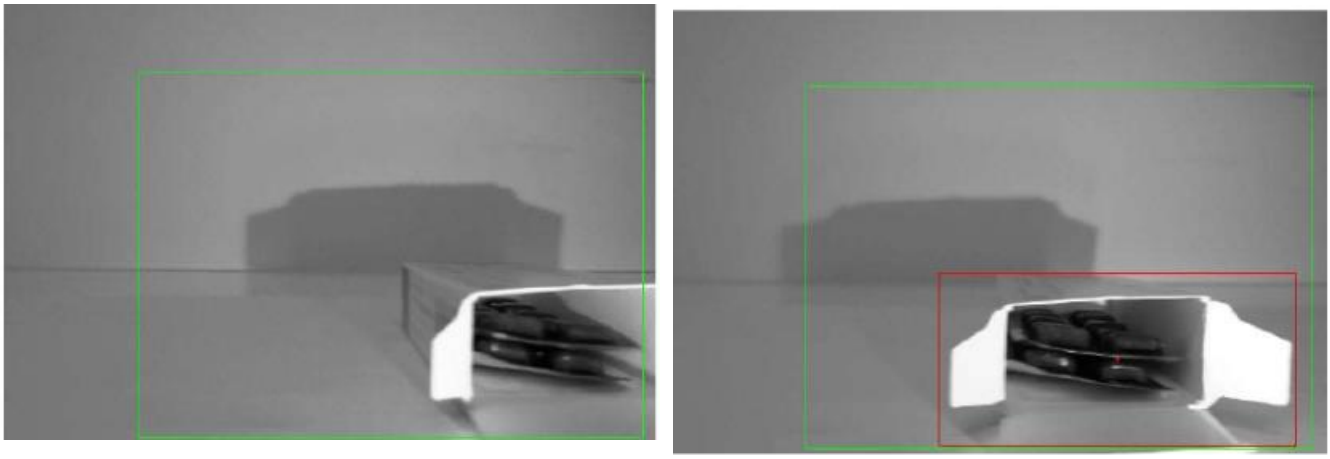


Figure 3.1 images of medicine boxes.

ii. Vision inspection system for pharmaceuticals: This research describes a low-cost automated inspection system that is intended to monitor pharmaceutical product quality on a manufacturing line. The essential components of the system are a smart camera and specially created LabVIEW-based software. It has the ability to identify several flaws of different kinds[7].

iii. Improvement Object Detection Algorithm Based on YoloV5 with BottleneckCSP: The benefit of employing deep learning technology for object detection is high accuracy. The processing speed of deep learning technology determines the accuracy attained. You Only Look Once (YOLO) is one object detection technique. Yolov5, or Yolov5, is the current version of YoLO. This work suggests a real-time object detection technique utilizing a Yolov5-captured video dataset from a highway. The 480x480 augmentation data mosaic was the first addition made to YOLOv5. After practicing object detection with the YOLOV5 - BottleNeckCSP model, we obtained object data that was split into six classes[7].

iv. Deep Learning-Based Rice Leaf Diseases Detection Using Yolov5: This study uses the Yolov5 deep learning model to try and develop the most effective and straightforward model for detecting rice leaf disease. The most recent Yolo version, v5, has been added to the model. When it comes to object detection, Yolov5 performs and is more accurate than Yolov3 and Yolov4 models. This model can effectively distinguish between different types of rice leaf diseases. The 400 photos in the Rice Leaf Images Dataset, which can be downloaded from the Kaggle website, show leaves that are ill. The model for detecting rice leaf disease is trained, validated, and tested in this work using the Google Colab platform. The rice leaf disease has been identified and is thoroughly detailed, along with the required actions to be taken. The developed model makes use of 100 epochs[8].

v. A comparative study of YOLOv5 models performance for image localization and classification: Using a common image dataset, this paper compares various iterations of the YOLOv5 model and offers researchers specific recommendations for choosing the best model for a particular kind of task[9].

vi. Quality Inspection and Maintenance: The Framework of Interaction: The purpose of this article is to determine whether or not the product quality complies with specifications using specified inspection techniques. Issues from the previous production process are one of the many potential causes if the product quality is subpar. To maintain the production process, the manufacturing equipment's performance is crucial. The high-quality product is guaranteed by the well-maintained machinery. The literature has provided a thorough analysis of the functions and benefits of both equipment maintenance and quality inspection [10].

vii. Artificial Intelligence-Based Smart Quality Inspection for Manufacturing: This paper provides a Deep Learning (DL) based Artificial Intelligence (AI) method to the visual assessment process. To make the inspection process more user-friendly, the approach combines a customized Convolutional Neural Network (CNN) for inspection and a computer program that can be installed on the shop floor. Based on image data of casting goods, the proposed model's inspection accuracy is 99.86%[11].

viii. Artificial Intelligence for Product Quality Inspection toward Smart Industries: Quality Control of Vehicle Non-Conformities: Inspection has always been a matter of interest in the industrial setting, as this study demonstrates. In fact, maintaining quality control over the final products depends heavily on the inspection procedure. In this context, an automated artificial vision system is proposed in this study to control vehicle non-conformities[12].

ix. Design of an economical SCARA robot for industrial applications: The mechanical design process of the FUM SCARA robot, an affordable and industrial SCARA robot created by a group of Iranian students from Ferdowsi University of Mashhad, is presented in this study. Because of their superior accuracy and inherent rigidity, SCARA robots are among the most often utilized robots in industry. The design procedure comprised the selection of mechanical and electrical components as well as joint, link, and controller designs. The task at hand involved utilizing inexpensive, easily obtainable parts in Iran. Impressive features of the FUM SCARA robot include a pick and place cycle time of 0.5 seconds, a maximum linear velocity of 8.5 m/s in the xy plane, and a versatile control system with repeatability of ~±0.01 mm[13].

x. SCARA Robot: Modeled, Simulated, and Virtual-Reality Verified: This essay offers The importance and popularity of articulated-morphology robots, which have applications ranging from simple to complex, have grown, particularly with the decline in computer costs and the rise in feasibility studies. The analytical inverse kinematic problem (AIKP) and the forward kinematic solution with D-H parameters, as well as the development of a comprehensive mathematical model for an industrial, selective compliance articulated (SCARA) robot arm, including its servomotor dynamics and simulation of the dynamics, are presented here. The robot arm is designed to follow specific paths in manufacturing, assembly, handling, etc. In order to simulate the design on MATLAB Version R2012a, the 3D virtual reality (VR) model realizes it builds and receives orders using a MATLAB/Simulink link[14].

xi. A Review of SCARA Robot Control System: This essay offers Because of its great speed and accuracy, the SCARA robot is one of the most well-known robotic arms in the market. The controller is essential to guaranteeing the SCARA robot at such a performance in order to attain adequate control at high speeds. An overview of the controllers and their benefits and drawbacks for use with the SCARA robot is provided in this document. PID control, fuzzy logic control, neural network control, sliding mode control, impedance control, adaptive control, and robust control are among the controllers that have been examined. In addition, as part of the IR 4.0 movement, the paper discusses the application of artificial intelligence (AI) and the Internet of Things (IoT) on the SCARA robot[15].

xii. Optimizing Deep Learning Models For Raspberry Pi: This essay offers For a variety of uses, such as speech recognition, computer vision, and natural language processing, deep learning models are growing in popularity. But these models usually need a lot of processing power, which makes it hard to operate them on low-power devices like the Raspberry Pi. Reducing the size of the deep learning models with pruning approaches is one way to tackle this difficulty. Pruning makes a model smaller and more effective by eliminating connections and weights that aren't necessary. Either during or after the model has been trained, pruning can be carried out[16].

xiii. Deep Learning on a Raspberry Pi for Real Time Face Recognition: In this paper, we describe a convolutional neural network (CNN) based real-time face recognition pipeline that is quick and accurate, requiring only moderate processing resources. Following the CNN's training on a desktop computer, we used a Raspberry Pi model B for the classification process. Here, we were able to achieve over 97% recognition accuracy and a performance of about 2 frames per second[17].

xiv. Machine Learning with the Raspberry Pi: : This document details our data and computer vision experiments:In this essay, we outline Expanding upon the ideas initially introduced in Starting Artificial Intelligence with the Raspberry Pi, you'll go beyond merely comprehending AI concepts

to actually working with real machine learning experiments and putting useful deep learning ideas to use in computer vision and Pi board experiments. You may then apply what you learn about machine learning with the Raspberry Pi to advance your personal or professional projects by using it on other platforms[18].

xv. Deep Learning for Inexpensive Image Classification of Wildlife on the Raspberry Pi: In this work, we discuss the need for non-intrusive ways for wildlife observers and researchers to observe and study species in remote locations. In distant areas, a lot of commercial solutions for wildlife monitoring are pricy, intrusive, or not the best. In this study, we investigate the feasibility of finding wildlife of interest using a deep learning image recognition model added to a Raspberry Pi-based camera system. Localized image recognition makes it possible to provide the user only the images of the requested animals, as opposed to typical sensor nodes that would have to send every image that was collected. For this research, we build a convolutional neural network using a Raspberry Pi 3B+ using TensorFlow and Keras.

The existing conventional methods include,

i. **Area Scan Imaging:** In area scan imaging, a camera captures a single, static image of an entire field of view. It is suitable for inspecting stationary objects, flat surfaces, and objects within a fixed frame. This method is commonly used for print inspection, label verification, and defect detection on static objects.
ii. **Line Scan Imaging:** When an object passes in front of the camera, line scan cameras record images one line at a time. This is particularly useful for inspecting objects on conveyor belts or production lines where continuous inspection is required. Line scan imaging is often used for applications like web inspection in printing, packaging, and textiles.
iii. **High-Speed Imaging:** High-speed cameras are designed to capture images at extremely fast frame rates. They are used in applications where high-speed events or fast-moving objects need to be observed and analyzed, such as in sports analysis, automotive crash testing, or manufacturing processes with rapid movements.

3.2 Relevant technologies-Techniques and sensor technology used

The review of existing technologies help us to statergically plan the architectural layout, sensors, actuators, software tools that can be used to develop our solution.

3.2.1 Modelling : Initial stage of the prototype building involves modelling using toolchains like MATLAB and Solidworks, where we can build, integrate and simulate various parts and assemblies of our prototype. This phase helps in analyzing the behaviour of the model in a real world environment, optimizing and also customizing the model with respect to various aspects.

3.2.2 Hardware and software requirements : Hardware components include microcontroller or microprocessor to control the entire mechanism, standard connectors that help in flow of control, sensors to extract data, a suitable prototype body etc. Software toolchains like MATLAB, Solidworks, Proteus are required to implement variousconcepts like Image Processing, Signal Processing, Neural Networks, Statistics and Machine Learning to achieve the required conditions of the client.

3.2.3 Neural Network: Both shallow and deep neural networks can be effectively created, trained, visualized, and simulated using neural network software, algorithms, and pretrained models. Dynamic system modeling and control, time-series forecasting, regression, clustering, and dimensionality reduction are all possible. Convolutional neural networks (CNNs), directed acyclic graph (DAG) network topologies, and autoencoders for feature learning, image classification, and regression are examples of deep learning networks. The toolkit offers deep learning networks with long short-term memory (LSTM) for time-series classification and regression. The network design can be changed, intermediary layers and activations can be seen, and training progress can be tracked.

3.2.4 Deep Learning: Machine learning concepts include techniques like reinforcement learning that help to excels in tasks that involve visual recognition and pattern detection. Deep learning is utilized in the quality inspection of medicine boxes due to its proficiency in recognizing intricate visual patterns, adapting to diverse packaging designs, and automating the inspection process. With the ability to efficiently detect defects such as damaged packaging and incorrect labeling, deep learning enhances accuracy, reduces false positives, and ensures regulatory compliance in pharmaceutical manufacturing. Its data-driven decision-making, adaptability to changes, and high-speed capabilities make it a valuable technology for maintaining the integrity and safety of medicine packaging, ultimately contributing to enhanced quality control in the industry.

3.2.5 Sensor Technology:

Camera – Microcontroller powered cameras can be helpful in capturing and transmitting data continuously and help various algorithms to perform actions. This sensor can be efficient in building an appropriate application for the need statement.

CHAPTER 4

Needs and capabilities

4.1 Needs and capabilities

Operational Needs refer to the essential requirements and resources that an organization or system must have to function effectively and achieve its objectives. These needs encompass a range of elements, including personnel, equipment, technology, and logistical support, tailored to meet the specific demands of the operational environment. Table 4.1 lists the operational needs for the robot.

Table 4.1: Operational Needs

ON1_1	Robot should have 3 degrees of freedom.
ON1_2	Robot should table mounted.
ON1_3	The work volume of the robot must be 400mm*400mm*360mm
ON1_4	Total weight of the robot should be 6-7kg.
ON2_1	Robot should able to integrate with other systems.
ON2_2	Robot should have least vibration.
ON2_3	Robot should operate safely and minimize the risk of accidents.
ON2_4	The optimization of robot should reduce the maintenance costs.
ON3_1	The input power to the robot should be __W.
ON3_2	The robot should equipped with camera module and other sensors.
ON4_1	The camera module is able to envelop the entire work volume.
ON4_2	The camera module should able to recognize any size of boxes within the volume of __mm*__mm*__mm.
ON4_3	The robot system must be able to detect objects quickly and efficient real-time object detection.
ON5_1	The robot should pick the boxes and place on the pallet.
ON5_2	The most boxes that can fit on the pallet at a time is 1.
ON5_3	The robot should able to count the how many boxes were stacked on the pallet.
ON6_1	The maximum weight of the boxes must be 200 g.
ON6_2	Robot should use the least time to pick the box even in the presence of obstacles.
ON6_3	The robot must produce a specific path in order to reach the box and pick it without interference with other components.

ON6_5	The system should always keep the count of the box , whenever the box is picked up.
ON7_1	The gripper should handle the box without damage.
ON7_2	The system should notify that the box is placed on the pallet.
ON7_3	Robot should use the least time to place the box on the pallet even in the presence of obstacles.
ON7_4	The total number of boxes that are placed on the pallet should always be counted by the system.
ON8_1	The system should indicate whether the pallet is empty or loaded.
ON8_2	The system should detect and indicate if any faults or the malfunctions.
ON8_3	The system should give high product performance
ON8_4	The cycle time for pick and place must be less.
ON8_5	The system should stop automatically if no box is detected.
ON8_6	The system should stop when the box is out of specified dimensions.
ON9_1	The system should have emergency stop button if any error.
ON9_2	The operator must able to pause and resume the process through user interface.
ON9_3	The system should provide the history of previous data storage for the improvement.
ON10_1	Robot should be Cost efficient.
ON10_2	Assembly of robot should be easy to operate and inspect.
ON10_3	Robot should have Less maintenance.
ON10_4	Robot should have Scalability.
ON10_5	Robot should have modular architecture.
ON10_6	The components of the robot must be easily replaceable.
ON11_1	Reset button for the home position of robot during the shutdown process .
ON11_2	The user interface should provide the indication of proper functioning of all components before shutdown process.

4.1.2 Operational capabilities

In table 4.2 Operational Capabilities, on the other hand, denote the organization's ability to deploy and employ the necessary resources to fulfill its operational needs. This involves the integration of personnel, technology, and processes to achieve a desired level of performance and efficiency in executing tasks and missions.

Table 4.2: Operational Capabilities

Capability	Description
C1 Execute the task autonomously	It should be possible for the robotic system to choose and arrange boxes on a pallet on its own.
C1.1 Detect boxes	Pick and place procedures for the boxes in the order should be broken down by the system based on the dimensions of the boxes.
C1.2 Pick box	The robot is able to select a box from the pile.
C1.3 Handle box	Any box can be moved by the robot while it is holding it without falling.
C1.4 place box	The boxes can be positioned by the robot by setting them on the pallet.
C1.5 Pallet management	Pallet stacking will be automated by the robot, which will also notify the operator when a stack is empty or loaded. The robot will also know where to select a box from the stack.
C1.6 Operation control	The system should be able to manage and optimize the robot's overall operation .This also includes tasks such as troubleshooting, repairing .
C2 Harmless	A particular kind of safety is called "harmless," since the robot shouldn't injure anyone. The sophisticated capability of being harmless necessitates the execution of particular features.
C2.1 Move safety	While avoiding unintentional collisions with the operator, the boxes, or any other aspect of the work cell, the robot selects the box.
C2.2 Handle box safely	The robot moves while grasping a box, not letting go of it or running into any obstructions.
C2.3 Place box safely	Without causing any harm to the boxes, the bin, the pallets, or the operator, the robot places the boxes on the pallet.
C3 Increased availability	The robot has to available maximum for the task without any interruptions in the components and the software.
C3.1 Manage faults	The operator is able to understand the occurred problem in the system and able to fix them.

C3.2 Manage contingencies	The system minimize the impact of contingencies and confirm that normal operations can be restored as quickly as possible.
C4 Quick integration	Robot is able to quickly and easily integrate with other systems or components, allowing for efficient task.
C4.1 Calibrate	Configuring the robot for the desired task, setting up the calibration environment, performing calibration tests for the good accuracy.
C5 Support Human-robot interaction	An interface for human operators to use in system management will be provided by the system.
C6 Feedback control	The software and interface must continuously monitor the robot's performance and adjust its movements to stays on track and performs its tasks accurately.

4.2 Operational Scenarios

It includes different phases

I. Adaptation phase

In Table 4.3 The Adaptive Phase is a stage in the life cycle of a system or organization where it undergoes adjustments and enhancements to adapt to changing conditions and requirements. This phase is crucial for ensuring flexibility and responsiveness, allowing the entity to evolve and address emerging challenges effectively.

Table 4.3: Adaptive Phase

S1 system configuration and calibration	The operator install the necessary components of Scara robot such as the robot arm, controller, sensors, and end effector. Calibration of the SCARA robot involves setting the robot's coordinate system, calibrating its sensors and actuators and gripper position to reach the all positions.
S2 Testing and optimization	Once the robot has been calibrated and programmed, it undergoes a series of tests to ensure that it can perform its tasks accurately and consistently. Optimization involves adjusting the robot's parameters, such as its velocity or acceleration, or optimizing its algorithms to reduce errors and improve performance(accuracy).

II. Operation phase

Table 4.4:Operational Phase

S3 System Start	The user switch on the robot. Check status and communication with other systems . Robot moves to the home position. System should check and notify whether the sensors and actuators are working properly.
S4 Object detection	The sensor or the vision system detects the objects to be picked up. The vision system determines the size of the boxes and send the data to the controller. The controller determines the co-ordinates and orientation of the objects.

S5 *Manage order*	The movements needed to pick and position an object are carried out by the robot arm. The destination location on the pallet is determined by the system using the coordinates.

S6 Pick the object	By adjusting the robot's position and orientation to align the end-effector to picking location .End-effector moves to the location of box and verify that the object is securely held.
S7 Place the object	By adjusting the robot's position and orientation to align the gripper with the placement location. The robot end-effector moves to the desired placement location and release the object from the gripper . Controller verifies that the object has been placed correctly and notify that it is placed successfully. Robot moves to the home or safe position and prepare for next pick and place operation.
S8 Repeat the process	The robot should repeat same task without any interrupts until the user's termination.

In Table 4.4 The Operational Phase is the stage during which the organization or system is actively engaged in its core activities, executing missions, and achieving its objectives. It involves the utilization of operational capabilities to carry out tasks efficiently, often in a dynamic and challenging environment.

III. Maintenance Phase

Table 4.5: Maintenance Phase

S9 Monitoring and Emergency stop	The operator monitors the robot's position, speed, torque and other parameters through the interface. The robot's controller triggers an alert and stop the robot if there are any issues or if the robot start behaving anomalously. Once the emergency situation has been resolved, the robot's controller initiates a recovery procedure to resume operation.
S10 Maintenance	The operator should perform regular maintenance tasks such as cleaning and lubricating the robot's joints and inspect the robot's components including motors, sensors and vision system. He should replace damaged parts as needed and verify that the robot is operating correctly.

S11 Shut down	The operator has to send a stop command to the controller. The operator has to check the position of the robot whether it is in safe or unsafe state .Once it confirmed that it is in safe position ,the operator powers-off the whole robot system.

In Table 4.5 Maintenance Phase involves the ongoing support, upkeep, and optimization of the organization or system after it has been deployed and is in active operation. This phase includes routine maintenance, updates, and improvements to sustain performance levels, address issues, and ensure the longevity and reliability of the operational infrastructure.

4.3 Design – Mechanical

4.3.1 Assembly

Figure 4.1:Assembly of quality inspection system.

The mechanical assembly of the system involves a well-coordinated combination of various components, incorporating acrylic sheets, 3D printed parts, aluminum extrusion rods for base support, and a lead screw for the end effector. The structural foundation comprises sturdy acrylic sheets serving as the primary framework. These sheets are strategically cut and assembled to form the main body of the system.

To enhance stability and structural integrity, aluminum extrusion rods are employed as the base support, providing a robust and lightweight framework. These rods are meticulously integrated into the assembly, ensuring a stable foundation for the entire system.

4.3.2 Mass Properties

Mass properties of SCARA

Mass = 4251.90 grams

Total weld mass = 0.00 grams

Volume = 2157353.60 cubic millimeters

Surface area = 739672.37 square millimeters

Center of mass: (millimeters)

X = 83.33

Y = 136.39

Z = 7.08

Principal axes of inertia and principal moments of inertia: (grams * square millimeters)

Taken at the center of mass.

Ix = (0.86, 0.51, 0.01) Px = 36927898.00

Iy = (-0.51, 0.86, -0.06) Py = 181647664.88

Iz = (-0.04, 0.05, 1.00) Pz = 200723999.33

Moments of inertia: (grams * square millimeters)

Taken at the center of mass and aligned with the output coordinate system. (Using positive tensor notation.)

Lxx = 75311898.22 Lxy = 63892685.65 Lxz = 2139227.01

Lyx = 63892685.65 Lyy = 143355459.85 Lyz = -81701.63

Lzx = 2139227.01 Lzy = -81701.63 Lzz = 200632204.14

Moments of inertia: (grams * square millimeters)

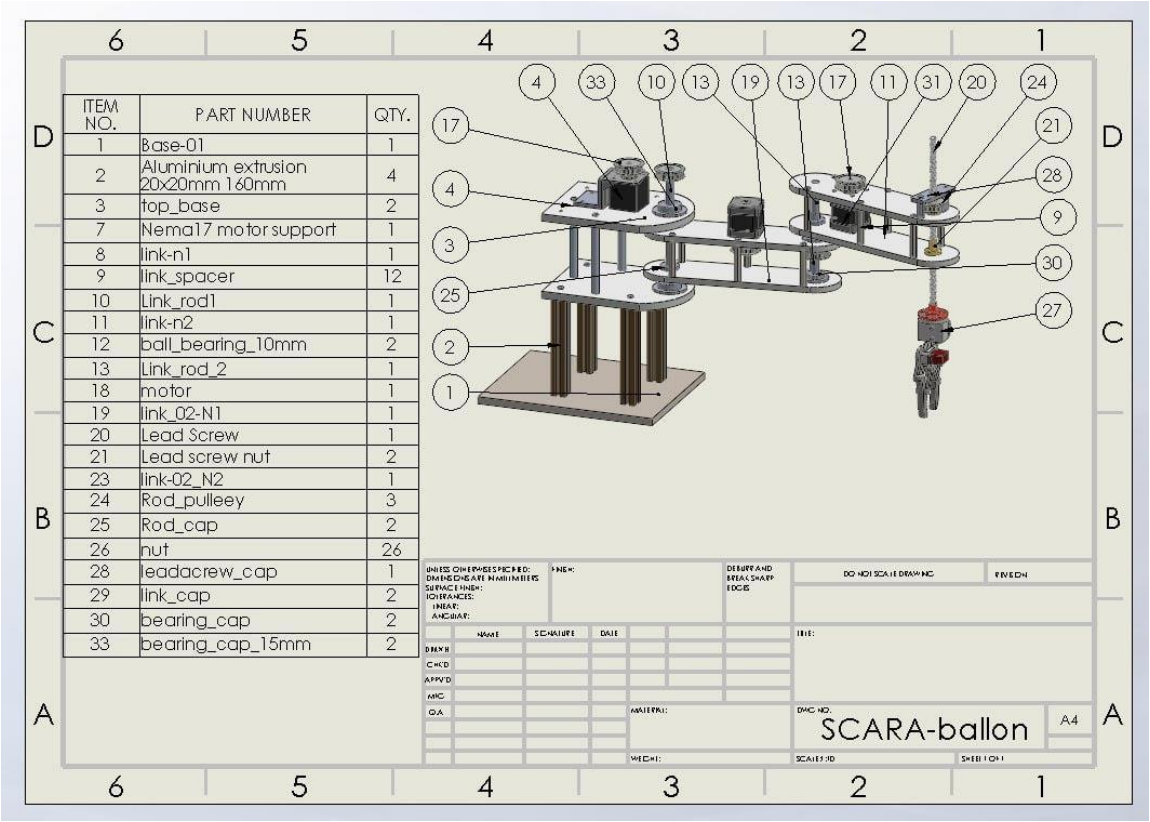
Taken at the output coordinate system. (Using positive tensor notation.)

Ixx = 154622473.13 Ixy = 112215885.99 Ixz = 4646892.54

Iyx = 112215885.99 Iyy = 173090632.94 Iyz = 4022957.14

Izx = 4646892.54 Izy = 4022957.14 Izz = 309251940.97

4.3.3 Bill of Material



ITEM NO.	PART NUMBER	QTY.
1	Base-01	1
2	Aluminium extrusion 20x20mm 160mm	4
3	top_base	2
7	Nema17 motor support	1
8	link-n1	1
9	link_spacer	12
10	Link_rod1	1
11	link-n2	1
12	ball_bearing_10mm	2
13	Link_rod_2	1
18	motor	1
19	link_02-N1	1
20	Lead Screw	1
21	Lead screw nut	2
23	link-02_N2	1
24	Rod_pulleey	3
25	Rod_cap	2
26	nut	26
28	leadacrew_cap	1
29	link_cap	2
30	bearing_cap	2
33	bearing_cap_15mm	2

Figure 4.2: Bill of Material of SCARA robot

4.4 Design – Electrical & Electronics

4.4.1 Torque Calculations

Known:

Length : L1=200mm ,L2=200mm

Spacers length: L1=17*7,L2=17*6

Weights:M1=457.5g,M2=437.65g

Stepper weight=288g

Servo motor = 36gm

payload =200g

lead screw weight=80gm

Torque Calculation:

Torque on M3

F= 360g = 0.36 kg
L= 25cm =0.25kg
E= 0.8
Torque= 0.36*0.25/2*3.142*0.8
Torque= 0.1825kg-cm

Torque on M2

Torque = (0.0595)*(0.288)+(0.0200)*(0.360)+(0.100)*(0.43765)
= 0.1329 Nm
= 1.89kg-cm

Torque on M3

Torque= (0.400)*(0.360)+(0.200+0.0595)*(0.200)*(0.200+0.100)*(0.288)
= 0.4259Nm
= 4.25kg-cm

4.4.2 Circuit Diagram

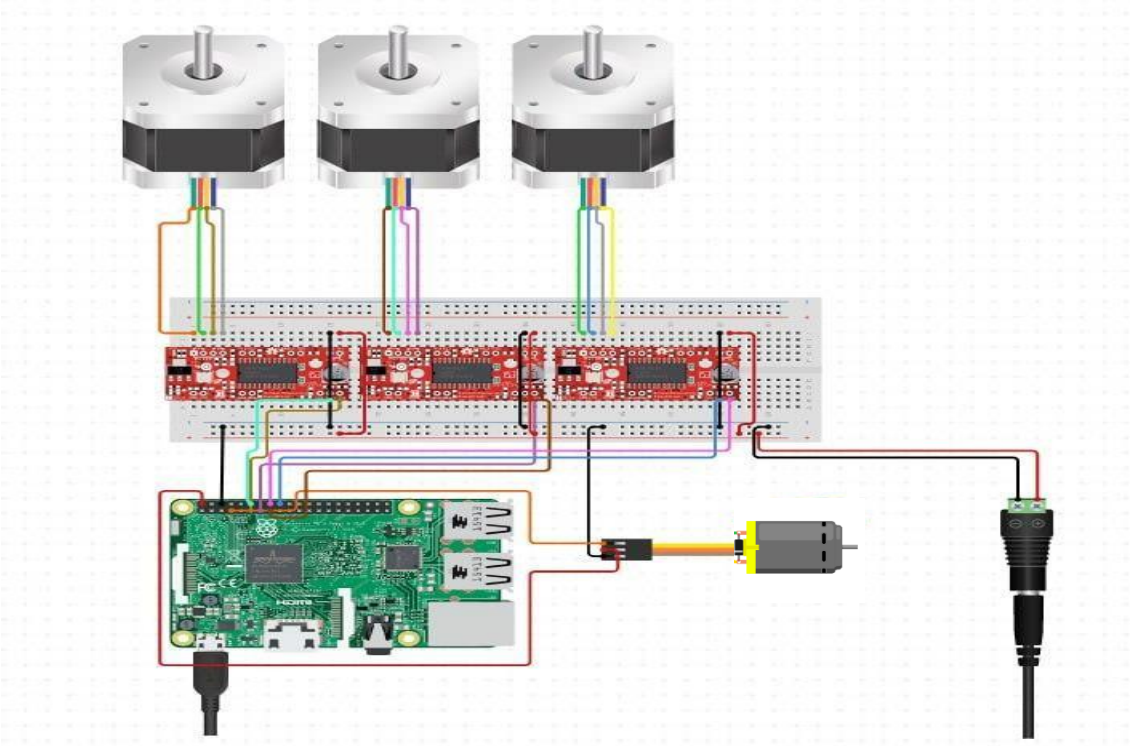


Figure 4.3: Circuit Diagram of Actuation system.

In Figure 4.3 The circuit for Vision-Based Quality Inspection of medicine packages integrates a Raspberry Pi 4 with a camera module for image capture, a NEMA17 stepper motor controlled by a stepper motor driver, and a DC motor managed by a motor driver for end effector movement. The Raspberry Pi processes captured images using computer vision algorithms, potentially employing deep learning models trained on a dataset of medicine box images. Decision logic within the software determines the quality of medicine packages, triggering controlled movements of the stepper and DC motors to perform quality inspections with precision. A careful consideration of power supplies, protective measures, and programming ensures seamless integration, allowing for intelligent quality inspection in pharmaceutical manufacturing.

4.5 Design – Software

Robot Control System: The robot control system is in charge of directing the movements and actions of the robot. It is made up of the following elements:

- ✓ Kinematics Solver: The kinematics solver calculates the robot's joint angles based on the location and orientation of its end effector. This section considers the physical constraints and limitations of the robot, including its joint range and workspace.
- ✓ Motion Planner: Based on the required task, the motion planner creates a trajectory for the robot's end effector. It creates a smooth and collision-free path while taking into consideration the robot's physical limitations, such as joint velocities and accelerations.
- ✓ Controller: After receiving the motion planner's generated trajectory, the controller instructs the robot's motors and actuators to carry out the required motion. To ensure precise and secure movements, it also keeps an eye on the robot's sensors, including encoders and force sensors.

4.5.1 Different ML model used in Matlab:

AlexNet :- AlexNet a pioneering deep convolutional neural network, revolutionized the field of computer vision, especially in image classification. Developed by Alex Krizhevsky and colleagues, it gained fame by winning the 2012 ImageNet competition. AlexNet comprises eight layers, featuring innovative design choices like ReLU activation functions and dropout to mitigate overfitting, leading to a top-5 error rate of 15.3%. In MATLAB, the Deep Learning Toolbox simplifies the implementation of AlexNet, offering pre-trained models for a range of computer vision tasks, further solidifying its significance in the development of advanced convolutional neural networks and the widespread adoption of deep learning in the field.

Google net Model:- Google created GoogleNet, commonly referred to as Inception V1, a deep convolutional neural network in 2014. It is well-known for its Inception modules, which use varying kernel sizes to capture information at different scales. Because of these modules, GoogleNet is efficient despite being relatively deep. To lower parameters and enhance training, it makes use of auxiliary classifiers and global average pooling. High accuracy was attained by GoogleNet, which had an impact on CNN architectures and later Inception models for computer vision tasks.
Resnet 50 Model: A deep neural network called ResNet-50 is well-known for its remarkable picture classification abilities. It combines residual blocks with skip connections to get around the problems deep

networks have with training. It has fifty layers. The bottleneck architecture of ResNet-50 lowers computing costs, and transfer learning frequently makes advantage of ImageNet's pre-trained models.

Mobilenetv2 model: MobileNetV2 is a neural network designed for mobile and embedded devices. It uses depthwise separable convolutions to reduce model size and computational cost while maintaining performance. It's highly efficient, offers flexibility through width and resolution adjustments, and is suitable for various computer vision tasks. MobileNetV2 is particularly popular for mobile and edge computing applications

4.5.2 Models used in Google Collab:

Resnet Architecture: ResNet is a deep neural network architecture introduced in 2015. It's known for using residual blocks with shortcut connections, making it possible to train very deep networks. ResNets are widely used in computer vision due to their ability to capture complex features. They often employ batch normalization, global average pooling, and are suitable for transfer learning. ResNet models have achieved state-of-the-art accuracy on various image recognition tasks.

YOLOv5: You Only Look Once version 5, or YOLOv5, is a cutting-edge real-time object identification technology renowned for its precision and quickness. It uses a single neural network to predict bounding boxes and class probabilities for several objects in an image at the same time. Improvements brought about by YOLOv5 include an effective inference pipeline, a new backbone architecture, and an emphasis on model scalability for better performance. Because it can provide quick and reliable results for object detection, it is widely utilized for a variety of computer vision applications, including autonomous cars, surveillance, and object recognition in a variety of situations.

Reason for selecting YOLOV5

The choice of YOLOv5 for quality inspection is often preferred over models like ResNet50, GoogleNet, MobileNet, and AlexNet due to its superior performance in real-time object detection tasks. YOLOv5, short for "You Only Look Once," is renowned for its efficiency, speed, and accuracy in detecting objects within images. Its architecture allows for simultaneous detection of multiple objects in a single pass, making it particularly well-suited for quality inspection scenarios where quick and precise identification of defects or anomalies is crucial. This efficiency, combined with the model's robustness and ability to handle various object scales, makes YOLOv5 a compelling choice for applications demanding high-quality inspection and rapid decision-making based on visual data.

4.5.3 Deploying YOLOv5 on a Raspberry Pi

Deploying YOLOv5 on a Raspberry Pi 4 involves several steps to optimize the model for the resource-constrained environment of the Raspberry Pi. Here's a theoretical overview of the process:

1. **Model Optimization:** Begin by optimizing the YOLOv5 model for the Raspberry Pi's limited computational resources. This may involve reducing the model size, quantizing the model's weights, or using model architectures that are more suitable for edge devices.
2. **Environment Setup:** Set up the Raspberry Pi environment by ensuring that the required dependencies, including Python, OpenCV, and any other libraries needed for YOLOv5, are installed. Consider creating a virtual environment to manage dependencies efficiently.
3. **Compilation for ARM Architecture:** Compile any necessary components or libraries for YOLOv5 to run on the ARM architecture of the Raspberry Pi. This may include compiling custom versions of libraries like OpenCV with optimizations for the Pi's hardware.
4. **Inference Pipeline:** Develop an efficient inference pipeline tailored for the Raspberry Pi. Optimize the code to take advantage of the Pi's hardware capabilities and minimize computational load during real-time object detection.
5. **Input Stream Handling:** Implement mechanisms for handling input streams, such as camera feeds or video files, on the Raspberry Pi. Ensure that the input resolution aligns with the model's expectations for optimal performance.
6. **Real-Time Inference:** Design the deployment to support real-time object detection on the Raspberry Pi, considering factors like frame rate, accuracy, and latency. Fine-tune the model and parameters for a balance between performance and detection accuracy.
7. **Power and Memory Management:** Consider power and memory constraints on the Raspberry Pi. Implement strategies to manage memory efficiently and optimize power consumption during continuous inference.
8. **Integration with Hardware:** Integrate YOLOv5 with any additional hardware components, such as cameras or sensors, that may be connected to the Raspberry Pi for comprehensive real-world applications.
9. **User Interface (Optional):** Develop a user interface or visualization system to display the results of the YOLOv5 object detection on the Raspberry Pi. This could involve integrating with a display or providing output through a web interface.
10. **Deployment Considerations:** Finally, consider deployment considerations, such as whether the Raspberry Pi will operate in a standalone mode or as part of a larger system. Address any security considerations and implement measures to ensure the stability of the deployment.

User Interface: The user interface serves as the conduit between the robot and its human operator. It enables the operator to communicate with the robot, issue commands, and keep tabs on its condition. The following elements make up the user interface:

- ✓ Graphical User Interface (GUI): The GUI shows the operator a visual depiction of the robot's workspace and enables command sending and status monitoring. Python, Java, or C# are just a few of the computer languages that can be used to create the GUI.
- ✓ Devices for input and output (I/O): These include input devices like joysticks, keyboards, and touch screens as well as output devices like displays, LEDs, and speakers. With the help of these gadgets, the operator can communicate with the robot and get feedback.
- ✓ Communication Protocol: The communication protocol enables communication between the user interface and the robot control system. Standard protocols like Ethernet/IP, Modbus, or CAN can be used to implement the protocol.

CHAPTER 5

ARCHITECTURE OF THE SYSTEM

5.1 System Composer

In Figure 5.1 the comprehensive process of detecting and placing medicine boxes, the initial step involves situating the boxes in a predetermined location, where a high-resolution camera captures detailed images. These images undergo preprocessing to enhance quality and mitigate noise. Subsequently, a trained deep learning model is employed to scrutinize the images, swiftly and accurately identifying any defects or irregularities in the medicine packaging. The decision-making process, guided by the deep learning model, distinguishes between open and closed medicine packages. Throughout this operation, external factors such as noise, energy, and heat may impact the process. The culmination of these steps, combined with the precision of a 3 DOF SCARA Robot, facilitates the placement of boxes into predefined positions, contributing to a seamless and robust pharmaceutical quality inspection process.

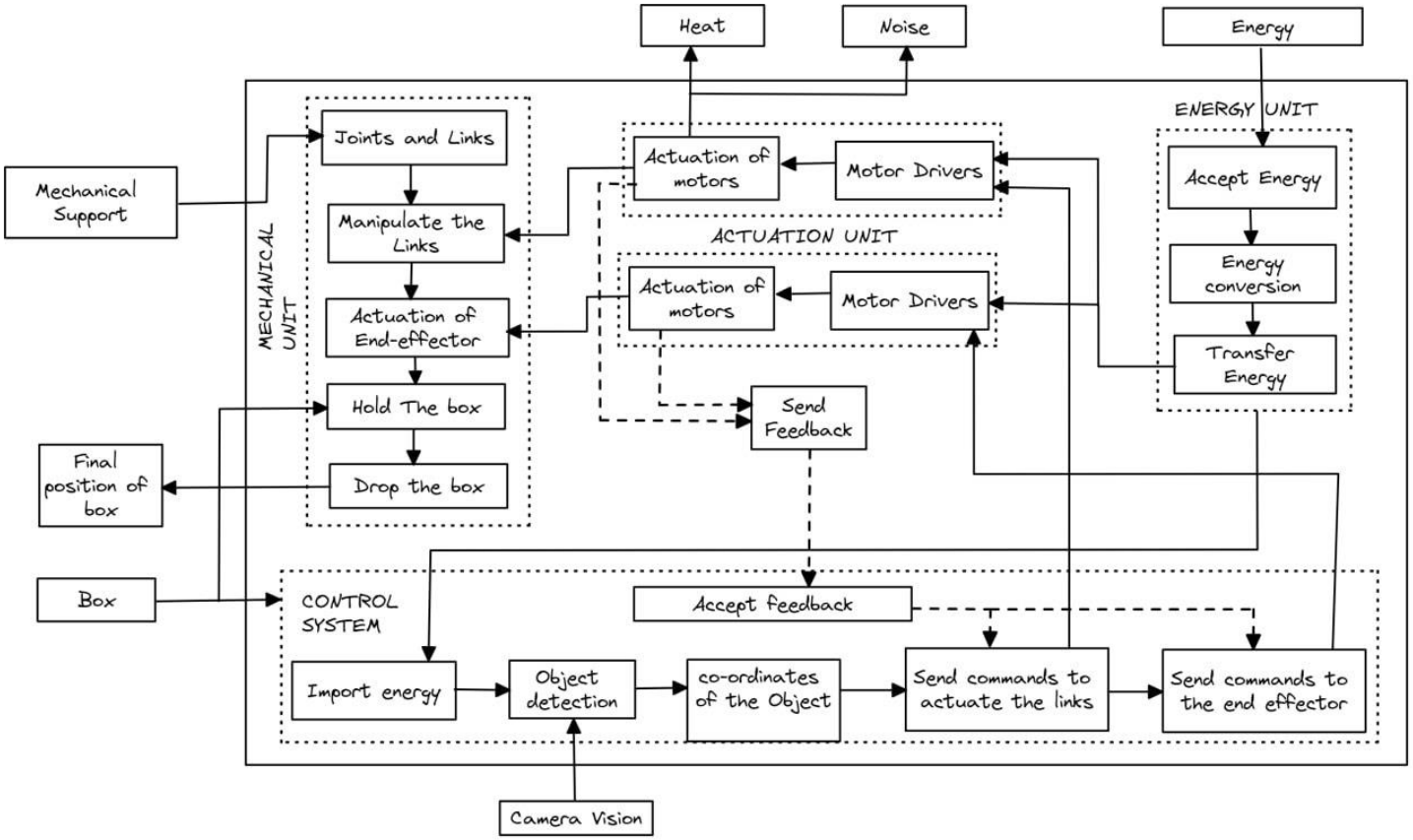


Figure 5.1: Architecture of quality inspection system.

CHAPTER 6

METHODOLOGY

The precise procedure used from the beginning of the project to its conclusion is known as the methodology. The project's technique includes multiple stages that are each very important. The procedure begins in a highly speculative manner. The specifications required force us to select the right technology and software as we work through the analysis to create the finished product.

6.1 HARDWARE

Hardware involves all the necessary electrical components, mechanisms, characteristics of components, energy, and all other necessary specifications that are needed to be considered to build the solution.

6.1.1 Black Box

In figure 6.1the first abstract structure in our model is called the black box. It shows the anticipated result of the solution while taking into account every input into the model. The inputs and outputs of the quality inspection system are shown by the black box below. Power supply, user input, signal, and box are taken in as inputs, and the outputs include placing the box at the goal, producing heat energy and sound energy.

Figure 6.1 Black box of quality inspection system.

6.1.2 Sys-ML Function diagram

In Figure 6.2 A general-purpose modeling language for systems engineering applications is called the Systems Modeling Language. Functions and their sub-functions for identifying boxes and positioning them in their proper locations are included in the Systems Modeling Language for the quality inspection of pharmaceutical boxes. Detect the boxes is a function for detecting medicine packages using camera.sys-ML function diagram represent the sequence of the SCARA operations.

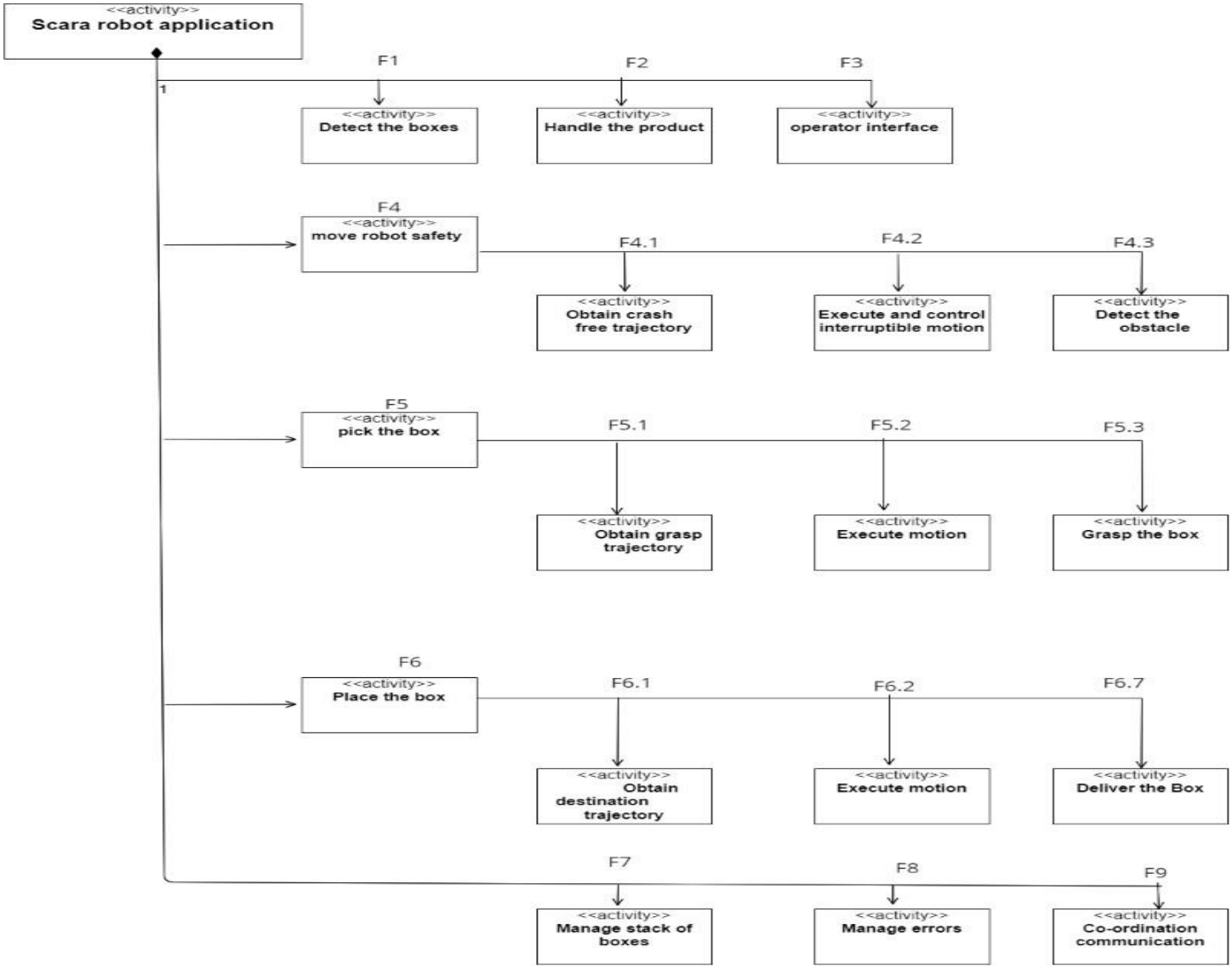


Figure 6.2: SYML of functions of SCARA Robot

6.1.3 Use Case Diagram

In Figure 6.3 A use case diagram of a quality inspection system using a camera and a SCARA robot would show the main functions and interactions associated with quality control procedures. In the diagram, the operator and technician might be viewed as performers. Start, calibrate, emergency stop, link, move, monitor, and configure are examples of use cases. Either circles or ellipses are used to symbolize the use cases. The operator has access to start, emergency stop, link movement. Technician calibrate, monitor and configure.

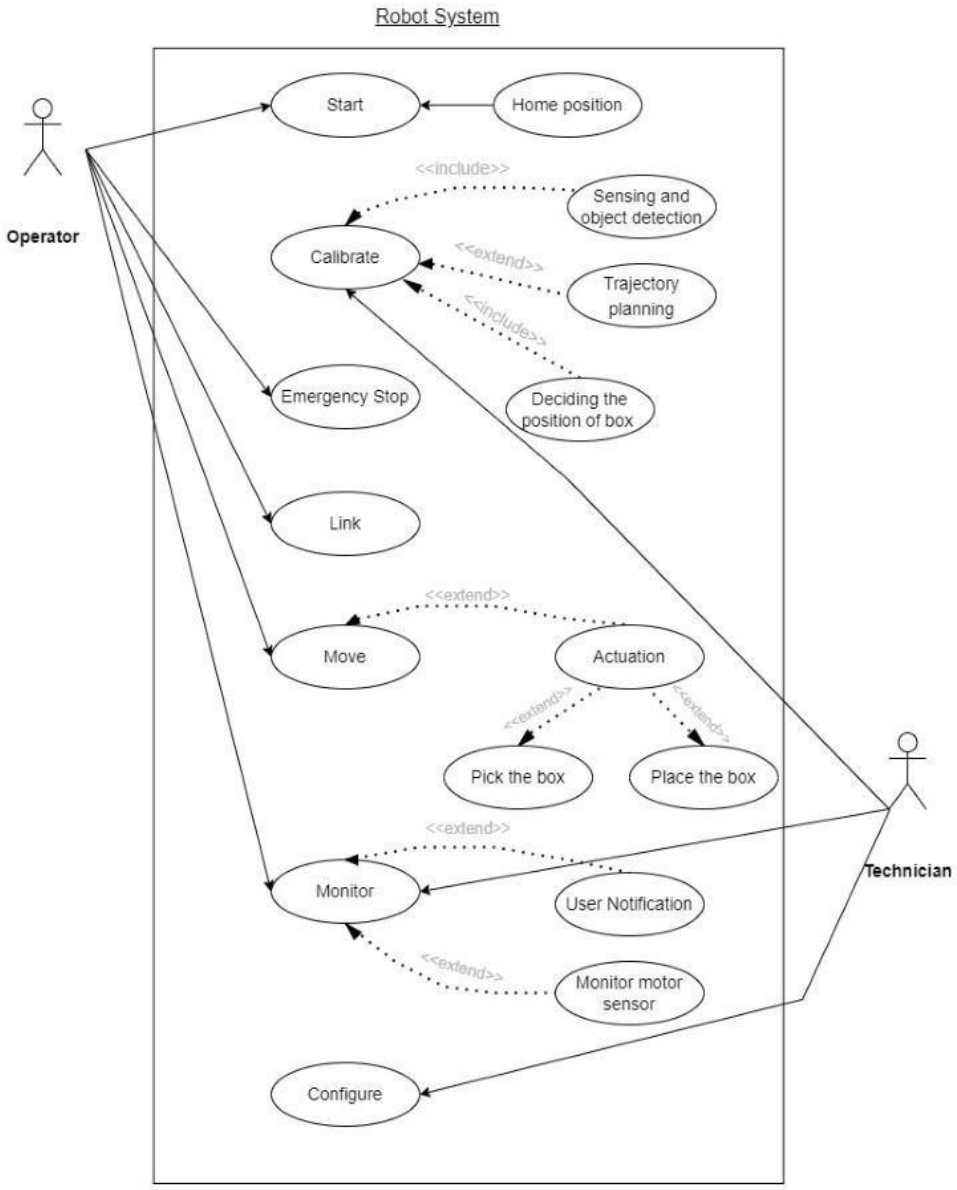


Figure 6.3 Use Case Diagram scara robot system.

6.2 SOFTWARE

Software programs and auxiliary services are what engineering design tools are basically used for while building 3 DOF SCARA robots. Included are all stages of engineering work employed in the design, building, handover, first operations, and maintenance of industrial infrastructure and facilities.

Software packages including 2D and 3D CAD, engineering analysis, project software and services, collaborative engineering software, and asset information management are examples of specific segmentations. These tools are utilized in infrastructure and industry not just for creating assets but also for managing data and information during the course of the physical asset lifespan. They give owners documents detailing facility upgrades as well as as-built data for operations and maintenance needs.

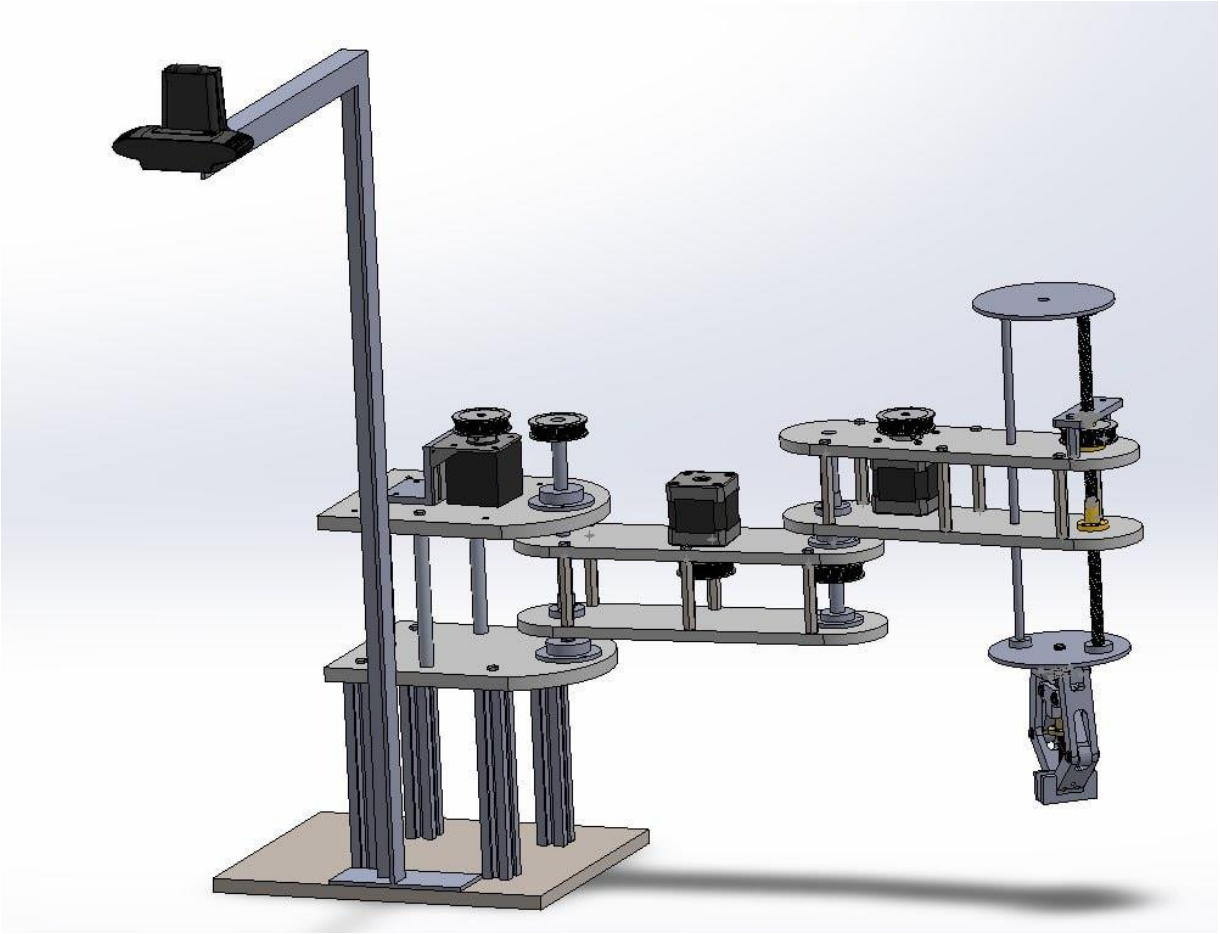


Figure 6.4 SolidWorks model of SCARA Robot

In Figure 6.4 the image illustrates a SolidWorks model of a SCARA (Selective Compliance Assembly Robot Arm) robot, showcasing its detailed design and spatial configuration. This visual representation provides insights into the physical structure and engineering considerations of the SCARA robot for potential applications in manufacturing or automation.

```python
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

# Load the saved model
model = load_model('image_classification_model.h5')

def predict_image(filename):
    img = image.load_img(filename, target_size=( 480, 640))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img = img / 255.0  # Normalize the image
    prediction = model.predict(img)

    if prediction >= 0.5:
        return "Defective"
    else:
        return "Non-Defective"

# Usage:
image_filename = r"C:\Users\Hp\OneDrive\Desktop\defective_4.jpg"
result = predict_image(image_filename)
print("Prediction:", result)
```

Figure 6.5 Building Deep learning model

In figure 6.6 An image featuring code for building a deep learning model showcases a script, likely in Python using frameworks like TensorFlow or PyTorch. The code covers importing libraries, defining the neural network architecture, training the model, and evaluating its performance, encapsulating the essential steps in constructing a deep learning model.

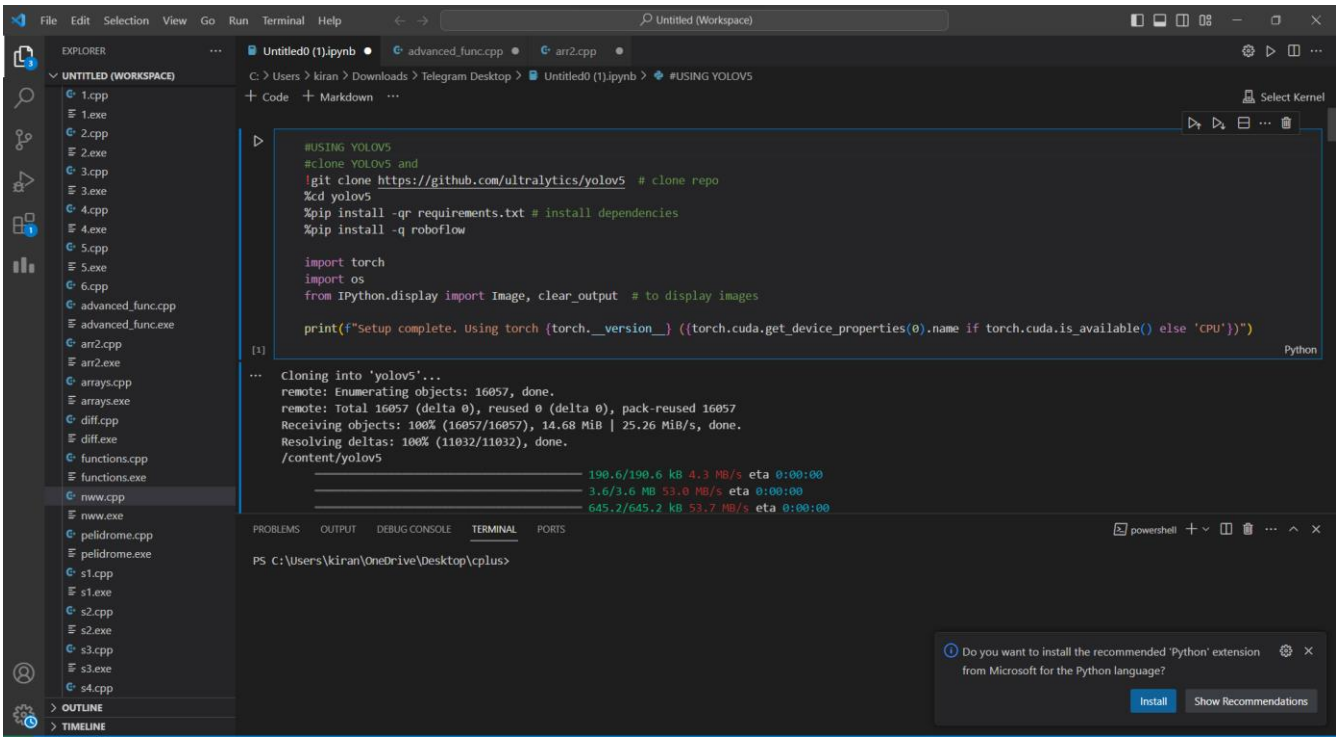


Figure 6.6 Deep learning model using YOLO V5

CHAPTER 7

RESULTS

The objectives of quality inspection for medicine boxes using a SCARA robot can be further advanced through the integration of deep learning techniques. By incorporating deep learning algorithms, the SCARA robot gains the capability to perform sophisticated image recognition and analysis. This encompass ensuring the integrity, accuracy, and safety of pharmaceutical products. The primary goal is to employ the SCARA robot's precision and speed to systematically examine various aspects of the medicine boxes. This includes verifying the correctness of packaging information, inspect for any physical defects, ensuring the integrity of the packaging, and seals. Additionally, the SCARA robot can play a crucial role in detecting anomalies in the quantity. By automating these inspection processes, the objectives are to enhance efficiency, reduce human error, and ultimately guarantee the delivery of high-quality and compliant pharmaceutical products to consumers. This contributes to maintaining regulatory compliance, safeguarding patient safety, and upholding the reputation of pharmaceutical manufacturers.

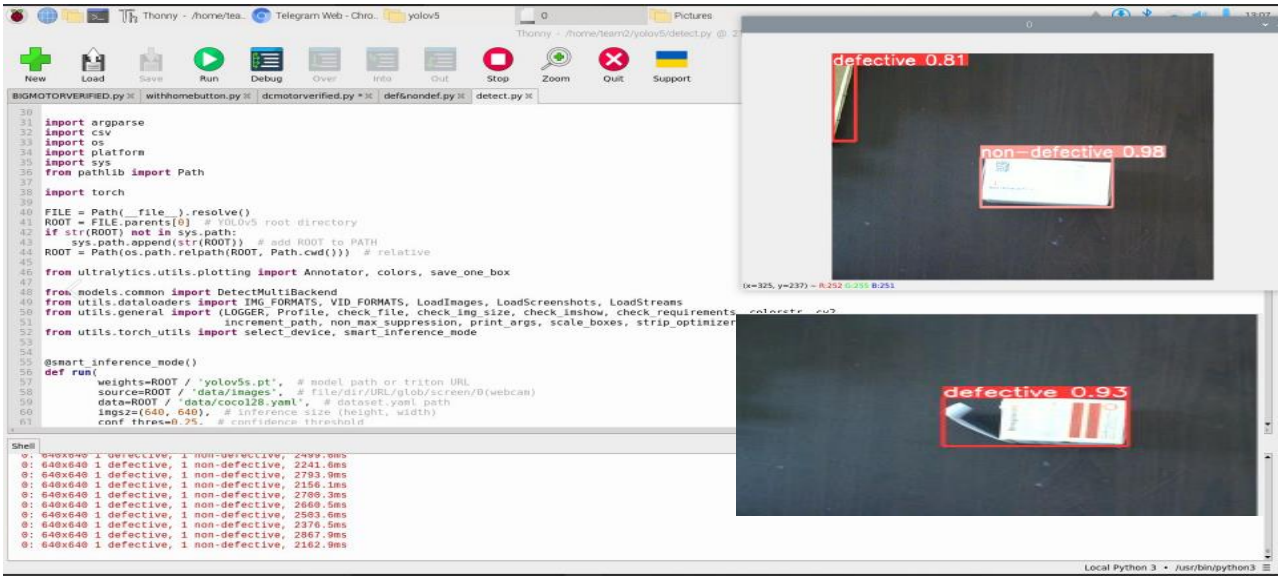


Figure 7.1 Real-time input and output analysis

In figure 7.1 The image illustrates a setup where medicine boxes are captured by a camera for quality inspection, showcasing the integration of visual data into a computer vision system, employing deep learning algorithms for automated assessment in quality inspection. Deep learning model classify medicine boxes with accuracy around 95%.

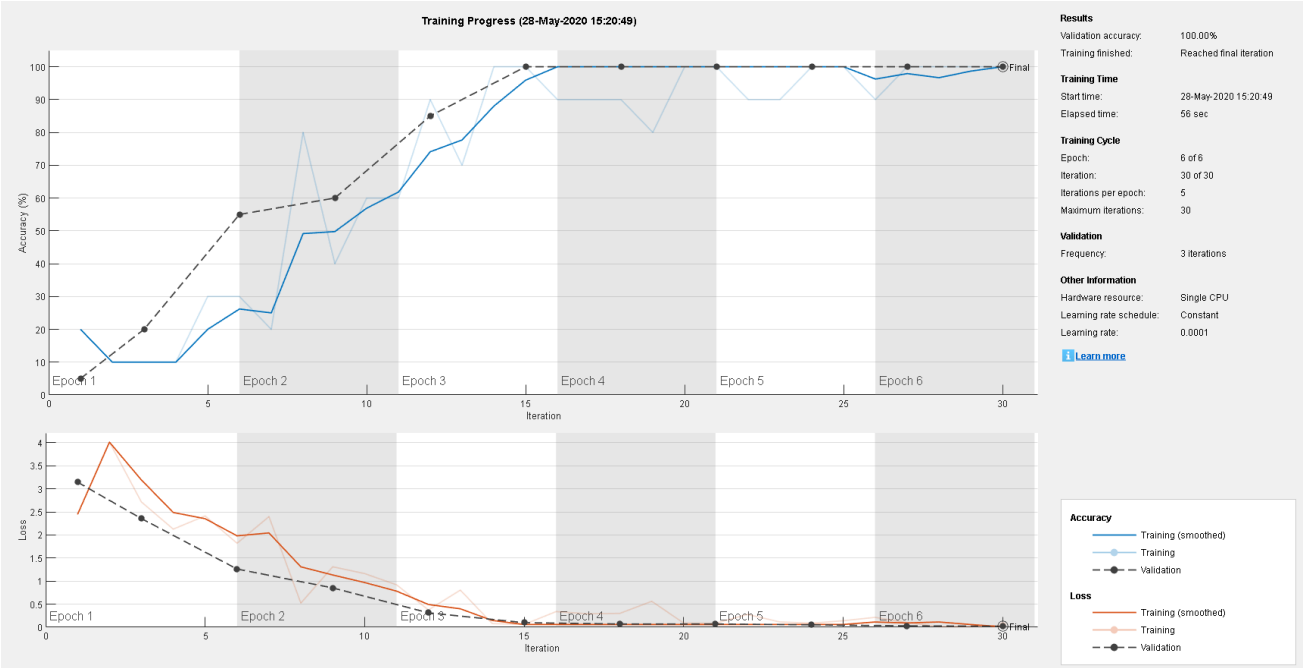


Figure 7.2 Accuracy of google net in MATLAB

In figure 7.2 the image portrays accuracy and loss of google net model that is trained in MATLAB.In the context of deep learning, accuracy and loss are commonly used metrics to assess the model's capability to correctly classify and minimize errors during training, respectively.

The accuracy graph typically shows the percentage of correctly classified instances over training epochs. As the training progresses, you expect to see the accuracy increase, indicating that the model is learning and improving its predictive capabilities. However, fluctuations or plateaus may occur, and it is essential to monitor these trends.

The loss graph illustrates how well the model is minimizing errors during training. It represents the difference between the predicted outputs and the actual labels. In an ideal scenario, the loss should decrease over epochs, reflecting the model's improved ability to make accurate predictions. Sharp increases in loss may indicate issues such as overfitting, while a stagnant or slow decrease may suggest the need for model adjustments.

The average accuracy of Google Net, on benchmark datasets like ImageNet is typically reported to be around 74.8% for top-1 accuracy and 93.3%.For our medicine package datasets accuracy of google net is around 89.2%.

CHAPTER 8

CONCLUSIONS

8.1 Conclusion

In conclusion, the Vision-Based Quality Inspection of medicine packages using a 3 DOF SCARA Robot presents a forward-thinking solution that integrates cutting-edge technology, including deep learning, to revolutionize pharmaceutical quality control. By harnessing the power of computer vision, the precision of a 3 DOF SCARA Robot, and the capabilities of a deep learning model, this system aims to elevate standards of quality assurance in medicine packaging. The outlined goals, including the reduction of manual labor, intelligent detection of defects, and optimization of production quality, collectively contribute to ensuring the delivery of safe and high-quality pharmaceutical products to end-users. The proposed vision-based inspection system, with its integration of deep learning, intelligent automation, and meticulous defect detection capabilities, is poised to set new benchmarks in pharmaceutical quality control, benefiting both manufacturers and end-consumers alike.

In the future, research and development in the field of SCARA robots with 3-DOF will focus on enhancing manipulation capabilities, enabling multi-robot collaboration, incorporating adaptive and learning behaviors, improving human-robot interaction, developing agile and dynamic control strategies, integrating advanced technologies, optimizing task planning, and addressing safety and ethical considerations. These advancements will drive the further improvement and wider range of applications for SCARA robots.

APPENDIX

IEEE Code of Ethics:

Here is an abridged version of the IEEE Code of Ethics:

1. Act with integrity and professionalism, upholding the highest ethical standards.

2.Exercise judgment and competence with a commitment to quality, safety, and the environment.

3.Serve the public interest, disclosing any potential conflicts of interest and avoiding actions that could harm the public.

4.Strive to enhance individual and collective knowledge, skills, and abilities in the field of technology.

5.Respect and uphold the dignity and rights of all individuals, treating them fairly and without discrimination.

6.Foster an inclusive working environment that embraces diversity and equal opportunity.

7.Respect and protect the privacy and confidentiality of others, ensuring the security of personal information.

8.Engage in responsible use of technology, considering its impact on society and the environment.

These principles guide engineers and technologists in their professional conduct, emphasizing the importance of ethical decision-making, responsibility, and the well- being of society. The complete IEEE Code of Ethics can be found on the IEEE website.

CODE FOR QUALITY INSPECTION :-

```
import argparse
import csv
import os
import platform
import sys
from pathlib import Path
import time
import torch
from classnfun import motor_actuation


motor = motor_actuation()

FILE = Path(__file__).resolve()
ROOT = FILE.parents[0]  # YOLOv5 root directory
if str(ROOT) not in sys.path:
    sys.path.append(str(ROOT))  # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd()))  # relative
```

```
from ultralytics.utils.plotting import Annotator, colors, save_one_box

from models.common import DetectMultiBackend
from utils.dataloaders import IMG_FORMATS, VID_FORMATS, LoadImages, LoadScreenshots,
LoadStreams
from utils.general import (LOGGER, Profile, check_file, check_img_size, check_imshow,
check_requirements, colorstr, cv2,
                           increment_path, non_max_suppression, print_args, scale_boxes, strip_optimizer,
xyxy2xywh)
from utils.torch_utils import select_device, smart_inference_mode


@smart_inference_mode()
def run(
        weights=ROOT / 'yolov5s.pt',  # model path or triton URL
        source=ROOT / 'data/images',  # file/dir/URL/glob/screen/0(webcam)
        data=ROOT / 'data/coco128.yaml',  # dataset.yaml path
        imgsz=(640, 640),  # inference size (height, width)
        conf_thres=0.25,  # confidence threshold
        iou_thres=0.45,  # NMS IOU threshold
        max_det=1000,  # maximum detections per image
        device='',  # cuda device, i.e. 0 or 0,1,2,3 or cpu
        view_img=False,  # show results
        save_txt=False,  # save results to *.txt
        save_csv=False,  # save results in CSV format
        save_conf=False,  # save confidences in --save-txt labels
        save_crop=False,  # save cropped prediction boxes
        nosave=False,  # do not save images/videos
        classes=None,  # filter by class: --class 0, or --class 0 2 3
        agnostic_nms=False,  # class-agnostic NMS
        augment=False,  # augmented inference
        visualize=False,  # visualize features
        update=False,  # update all models
        project=ROOT / 'runs/detect',  # save results to project/name
        name='exp',  # save results to project/name
        exist_ok=False,  # existing project/name ok, do not increment
        line_thickness=3,  # bounding box thickness (pixels)
        hide_labels=False,  # hide labels
        hide_conf=False,  # hide confidences
        half=False,  # use FP16 half-precision inference
        dnn=False,  # use OpenCV DNN for ONNX inference
        vid_stride=1,
        prediction_delay=0.5,# video frame-rate stride
 ):
    global robot
    robot = None
    source = str(source)
```

```
    save_img = not nosave and not source.endswith('.txt')  # save inference images
    is_file = Path(source).suffix[1:] in (IMG_FORMATS + VID_FORMATS)
    is_url = source.lower().startswith(('rtsp://', 'rtmp://', 'http://', 'https://'))
    webcam = source.isnumeric() or source.endswith('.streams') or (is_url and not is_file)
    screenshot = source.lower().startswith('screen')
    if is_url and is_file:
        source = check_file(source)  # download

    # Directories
    save_dir = increment_path(Path(project) / name, exist_ok=exist_ok)  # increment run
    (save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True)  # make dir

    # Load model
    device = select_device(device)
    model = DetectMultiBackend(weights, device=device, dnn=dnn, data=data, fp16=half)
    stride, names, pt = model.stride, model.names, model.pt
    imgsz = check_img_size(imgsz, s=stride)  # check image size

    # Dataloader
    bs = 1  # batch_size
    if webcam:
        view_img = check_imshow(warn=True)
        dataset = LoadStreams(source, img_size=imgsz, stride=stride, auto=pt,
vid_stride=vid_stride)
        bs = len(dataset)
    elif screenshot:
        dataset = LoadScreenshots(source, img_size=imgsz, stride=stride, auto=pt)
    else:
        dataset = LoadImages(source, img_size=imgsz, stride=stride, auto=pt, vid_stride=vid_stride)
    vid_path, vid_writer = [None] * bs, [None] * bs

    # Run inference
    model.warmup(imgsz=(1 if pt or model.triton else bs, 3, *imgsz))  # warmup
    seen, windows, dt = 0, [], (Profile(), Profile(), Profile())
    for path, im, im0s, vid_cap, s in dataset:
        with dt[0]:
            im = torch.from_numpy(im).to(model.device)
            im = im.half() if model.fp16 else im.float()  # uint8 to fp16/32
            im /= 255  # 0 - 255 to 0.0 - 1.0
            if len(im.shape) == 3:
                im = im[None]  # expand for batch dim

        # Inference
        with dt[1]:
            visualize = increment_path(save_dir / Path(path).stem, mkdir=True) if visualize else False
            pred = model(im, augment=augment, visualize=visualize)
```

```
        # NMS
        with dt[2]:
            pred = non_max_suppression(pred, conf_thres, iou_thres, classes, agnostic_nms,
max_det=max_det)

        # Second-stage classifier (optional)

#         mean_confidence = torch.mean(pred[0][:, 4])  # Assuming confidence is at index 4
#         print(f"Mean Confidence: {mean_confidence.item()}")# pred =
utils.general.apply_classifier(pred, classifier_model, im, im0s)
#



        # Define the path for the CSV file
        csv_path = save_dir / 'predictions.csv'

        # Create or append to the CSV file
        def write_to_csv(image_name, prediction, confidence):
            data = {'Image Name': image_name, 'Prediction': prediction, 'Confidence': confidence}
            with open(csv_path, mode='a', newline='') as f:
                writer = csv.DictWriter(f, fieldnames=data.keys())
                if not csv_path.is_file():
                    writer.writeheader()
                writer.writerow(data)

        # Process predictions
        for i, det in enumerate(pred):  # per image
            seen += 1
            if webcam:  # batch_size >= 1
                p, im0, frame = path[i], im0s[i].copy(), dataset.count
                s += f'{i}: '
            else:
                p, im0, frame = path, im0s.copy(), getattr(dataset, 'frame', 0)

            p = Path(p)  # to Path
            save_path = str(save_dir / p.name)  # im.jpg
            txt_path = str(save_dir / 'labels' / p.stem) + ('' if dataset.mode == 'image' else f'_{frame}')  #
im.txt
            s += '%gx%g ' % im.shape[2:]  # print string
            gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
            imc = im0.copy() if save_crop else im0  # for save_crop
            annotator = Annotator(im0, line_width=line_thickness, example=str(names))
            if len(det):
                # Rescale boxes from img_size to im0 size
                det[:, :4] = scale_boxes(im.shape[2:], det[:, :4], im0.shape).round()
```

```
            # Print results

            for c in det[:, 5].unique():
                n = (det[:, 5] == c).sum()  # detections per class

                s += f"{names[int(c)]} ,"# add to string
                global a
                a=f"{names[int(c)]}"
                print(a)
                if a=='defective':
                    robot = 1
#                       print('pick')
#                       motor.home_position()
#                       motor.home_to_pick()
#                       motor.place_to_defective()
#                       motor.home_position()
#
                elif a=='non-defective':
                    robot = 0
#                       print('place')
#                       motor.home_position()
#                       motor.home_to_pick()
#                       motor.place_to_nonDefective()
#                       motor.home_position()
#



            # Write results
            for *xyxy, conf, cls in reversed(det):
                c = int(cls)  # integer class
                label = names[c] if hide_conf else f'{names[c]}'
                confidence = float(conf)
                confidence_str = f'{confidence:.2f}'
                d=0

                if save_csv:
                    write_to_csv(p.name, label, confidence_str)
                    cv2.imwrite(os.path.join(save_dir,"frame_%d.jpg"%d), im0)
                    d+=1

                if save_txt:  # Write to file
                    xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist()  #
normalized xywh
                    line = (cls, *xywh, conf) if save_conf else (cls, *xywh)  # label format
                    with open(f'{txt_path}.txt', 'a') as f:
                        f.write(('%g ' * len(line)).rstrip() % line + '\n')
```

```
                if save_img or save_crop or view_img:  # Add bbox to image
                    c = int(cls)  # integer class
                    label = None if hide_labels else (names[c] if hide_conf else f'{names[c]} {conf:.2f}')
                    annotator.box_label(xyxy, label, color=colors(c, True))
                if save_crop:
                    save_one_box(xyxy, imc, file=save_dir / 'crops' / names[c] / f'{p.stem}.jpg',
BGR=True)

            # Stream results
            im0 = annotator.result()
            title = "3-DOF SCARA ROBOT INTERFACE"
            cv2.putText(im0, title, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2,
cv2.LINE_AA)
            if view_img:
                if platform.system() == 'Linux' and p not in windows:
                    windows.append(p)
                    cv2.namedWindow(str(p), cv2.WINDOW_NORMAL | cv2.WINDOW_KEEPRATIO)  #
allow window resize (Linux)
                    cv2.resizeWindow(str(p), im0.shape[1], im0.shape[0])
                cv2.imshow(str(p), im0)
                cv2.waitKey(1)  # 1 millisecond


            if  robot == 1:
        #       print('pick')
                motor.home_position()
                motor.home_to_pick()
                motor.place_to_nonDefective()
                motor.home_position()

            elif  robot ==0:
        #       print('place')
                motor.home_position()
                motor.home_to_pick()
                motor.place_to_defective()
                motor.home_position()




        # Print time (inference-only)
        LOGGER.info(f"{s}{'' if len(det) else '(no detections), '}{dt[1].dt * 1E3:.1f}ms")



#         LOGGER.info(f"{'' if len(det) else '(no detections), '} ")
```

```
#         print(f"{" if len(det) else '(no detections), '} ")

    # Print results
    t = tuple(x.t / seen * 1E3 for x in dt)  # speeds per image
    LOGGER.info(f'Speed: %.1fms pre-process, %.1fms inference, %.1fms NMS per image at
shape {(1, 3, *imgsz)}' % t)

    if save_txt or save_img:
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to {save_dir / 'labels'}" if save_txt
else ''
        LOGGER.info(f"Results saved to {colorstr('bold', save_dir)}{s}")
    if update:
        strip_optimizer(weights[0])  # update model (to fix SourceChangeWarning)



def parse_opt():
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str, default=ROOT / 'best-fp16.tflite',
help='model path or triton URL')
    parser.add_argument('--source', type=str, default=0, help='file/dir/URL/glob/screen/0(webcam)')
    parser.add_argument('--data', type=str, default=ROOT / 'data.yaml', help='(optional)
dataset.yaml path')
    parser.add_argument('--imgsz', '--img', '--img-size', nargs='+', type=int, default=[640],
help='inference size h,w')
    parser.add_argument('--conf-thres', type=float, default=0.25, help='confidence threshold')
    parser.add_argument('--iou-thres', type=float, default=0.45, help='NMS IoU threshold')
    parser.add_argument('--max-det', type=int, default=1000, help='maximum detections per
image')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--view-img', action='store_true', help='show results')
    parser.add_argument('--save-txt', action='store_true', help='save results to *.txt')
    parser.add_argument('--save-csv', action='store_true', help='save results in CSV format')
    parser.add_argument('--save-conf', action='store_true', help='save confidences in --save-txt
labels')
    parser.add_argument('--save-crop', action='store_true', help='save cropped prediction boxes')
    parser.add_argument('--nosave', action='store_true', help='do not save images/videos')
    parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --classes 0, or --
classes 0 2 3')


def main(opt):
    check_requirements(ROOT / 'requirements.txt', exclude=('tensorboard', 'thop'))
    run(**vars(opt))
```

```
if __name__ == '__main__':
    opt = parse_opt()
    main(opt)
```

ISSUES AND TROUBLESHOOTING

1. Overheating of Pi

Solution :Use commercially available heat sinks for Pi fans.

2. Error while importing libraries in Pi

Solution : Examine the Pi's architecture closely. Make sure the installed libraries and Python version are compatible with one another.

3.Error while detecting defective and non defective boxes.Solution- Carefully consider the data set to ensure that the background can be distinguished from the boxes.

REFERENCES

1] Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2005). Robot modeling and control. John Wiley & Sons.
2] Paul, R. P. (1981). Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators. MIT Press.
3] Dini, G., Dufourd, J. F., & Zoppi, M. (2015). Design and control of a SCARA robot for automatic palletizing. Procedia Engineering, 100, 800-807.
4] Pham, C., & Tjahjadi, T. (2017). Vision-based robotic palletizing system using SCARA robot for mixed case palletizing. Robotics and Autonomous Systems, 97, 134- 144.
5] Mahdi Bahaghighat & Leila Akbari,(2018). Designing quality control system based on vision inspection in pharmaceutical product lines
6] N. M. Duong & S.N. Demidenko,(2014). Vision inspection system for pharmaceuticals
7] Aria Hendrawan, Rahmat Gernowo, Oky Dwi Nurhayati(November 2022). Improvement Object Detection Algorithm Based on YoloV5 with BottleneckCSP
8] Jhatial, Muhammad Shaikh, Dr Hussain, Rafaqat(2022). Deep Learning-Based Rice Leaf Diseases Detection Using Yolov5.
9] Marko Horvat ,Gordan Gledec(2022). A comparative study of YOLOv5 models performance for image localization and classification.
10] Nani Kurniati,Ruey Huei Yeh,Jong-Jang Lin(2015). Quality Inspection and Maintenance: The Framework of Interaction.
11] Sarvesh Sundaram,Abe Zeid(2023).Artificial Intelligence-Based Smart Quality Inspection for Manufacturing.
12] Amal Chouchene ,Adriana Carvalho, Tânia M. Lima(2020). Artificial Intelligence for Product Quality Inspection toward Smart Industries: Quality Control of Vehicle Non-Conformities.
13] Morteza Shariatee, Alireza Akbarzadeh, Ali Mousavi Mohammad(2014).Design of an economical SCARA robot for industrial applications.
14] Yousif Ismail Al Mashhadany(2012).SCARA Robot Modeled, Simulated, and Virtual-Reality Verified.
15] See Han Tay,Wai Heng Choong ,Hou Pin Yoong A Review of SCARA Robot Control System.
16] Salem Ameen, Kangaranmulle Siriwardana, Theo Theodoridis(2023). Optimizing Deep Learning Models For Raspberry Pi.
17] Oliver Dürr Diego Browarnik Rebekka Axthelm(2015).Deep Learning on a Raspberry Pi for Real Time Face Recognition
18] Donald J (2020).Machine Learning with the Raspberry Pi: Experiments with Data and Computer Vision.
19] Brian H. Curtin; Suzanne J. Matthews(2019). Deep Learning for Inexpensive Image Classification of Wildlife on the Raspberry