Industrial Management Advances 2023; 1(1): 144.
Original Research Article

Towards Industry 4.0: Color-based object sorting using a robot arm and real-time object detection

Ze Chern Ong[1], Kok Hoe Ho[1,*], Wen-Shyan Chua[2]

[1] *Department of Electrical and Robotics, School of Engineering, Monash University Malaysia, Jalan Lagoon Selatan, Bandar Sunway, Subang Jaya 47500, Selangor, Malaysia*

[2] *Selangor Human Resource Development Centre (SHRDC), Jalan Tinju 13/50, Seksyen 13, Shah Alam 40100, Selangor, Malaysia*

*** Corresponding author:** Kok Hoe Ho, ho.kokhoe@monash.edu

ARTICLE INFO

Received: 26 July 2023
Accepted: 18 September 2023
Available online: 25 December 2023

doi: 10.59400/ima.v1i1.144

Abstract: With the introduction of Industry 4.0, automation and robotics have made great strides, enabling enterprises to improve their manufacturing processes for increased productivity and efficiency. This project introduces a novel method for implementing Industry 4.0 concepts through color-based object sorting employing a robot arm with real-time object identification capabilities. Creating a reliable and effective system that can automatically categorize items based on their color properties is the main goal of this project. To enable seamless object recognition and manipulation in real time, the suggested system integrates robotic manipulation with computer vision algorithms. The system makes use of a convolutional neural network (CNN) for precise object detection, using recent advancements in deep learning and image processing, allowing the robot arm to interact with a variety of items effectively. The training phase and the sorting phase are the two key phases of the approach. The CNN model is trained on a sizable dataset of labeled objects during the training phase to recognize various colors and forms. In order for the robotic arm to recognize things as they go along the conveyor belt and sort them into predetermined bins according to their respective colors, the trained model must be integrated with the robotic arm during the sorting phase. Several experiments are carried out with various lighting setups and object arrangements to evaluate the performance of the suggested system. The outcomes show how well the system performs in terms of exact object detection and reliable sorting. The system's capacity to effectively handle a variety of objects and adapt to changing environmental conditions further emphasizes its suitability for use in actual industrial scenarios. This project has important ramifications for the manufacturing sector, enabling improved automation capabilities and cost-efficiency. An important step towards implementing Industry 4.0 principles is the seamless integration of color-based object sorting and real-time object detection using a robotic arm. This will allow industries to optimize their production processes, minimize human intervention, and increase overall productivity. Further developments in robotics and computer vision are anticipated to push the limits of automation and open the door for more advanced and intelligent industrial systems as

technology develops.

KEYWORDS: Industry 4.0; color-based sorting; robotic arm; real-time object detection; convolutional neural network (CNN); Internet of things (IoT)

1. Introduction

The robotic arm has gained popularity recently in a number of industries, including manufacturing lines, assembly lines, the construction business, the medical field, and even residences and the food service sector[1,2]. This is because robotic arms offer safety, increased production and efficiency, higher precision, and flexibility in carrying out various jobs. The automobile industry is where the robotic arm has the biggest impact; by 2020, 28% of robot installations will be used in this industry, according to the International Federation of Robots (IFR)[3]. The conventional method of commanding an industrial robotic arm through a typical teaching procedure utilizing a teach pendant needs to be altered because the robotic arm is attracting a lot of research and development from many industries. Therefore, innovative and understandable techniques for controlling and programming robots are needed to make way for new opportunities.

Active research has been done on the robotic arm powered by the Internet of things (IoT)[4]. The Internet of things (IoT) is the interconnection of physical objects that are integrated with electronics, software, sensors, actuators, and network connectivity to allow these things to talk to each other and share data. We could use IoT to remotely control the robotic arm from any location using the already-existing network infrastructure. Different approaches to integrating IoT with robotic arm control have been put out, although the more prevalent/familiar ones are operated by a smartphone application[5].

As a result, this project suggests modeling a theoretical notion for a remote system to control the robotic arm via IoT. Using a smartphone application as a control input to the cloud for computation, the robotic arm would be remotely directed to carry out a pick-and-place action. Through the graphical user interface (GUI) offered in smartphone applications, sensors, and cameras could give the user feedback and the information they need. A "cyber-physical system" is created when the robotic arm's microcontroller and smartphone can communicate over the Internet.

2. Problem statement

Product sorting is essential in the manufacturing sector for maintaining product quality, consistency, and effective material management. Items are regularly classified according to different attributes to satisfy particular client requirements, sustain brand standards, or adhere to legal requirements. The manual sorting procedure, however, presents major difficulties that reduce operational effectiveness.

First and foremost, the accuracy of hand-sorting procedures is a crucial concern. Fatigue among human operators may cause mistakes in product identification and sorting. Even with training, results can be inconsistent and unreliable due to subjective interpretations and differences in human perception. Serious repercussions could result from these errors, including inaccurate product shipments, lax quality controls, and tarnished brand reputation.

Second, the procedure is frequently cumbersome and ineffective. Each product must be physically handled, visually inspected, and sorted into the proper categories by human operators. Especially when dealing with large product numbers or intricate color distinctions, this process can be cumbersome. As a result, bottlenecks, delays, and decreased throughput may occur on production lines, which would be

detrimental to overall operational effectiveness.

The inefficient utilization of a sizable labor force, which led to high labor expenses and a greater dependency on human resources, is still a concern. The management and training of numerous operators can be time- and money-consuming. Additionally, manual labor is subject to performance variances and absenteeism, which further jeopardizes the manufacturing process' uniformity and dependability.

There is a chance to completely automate the color sorting process in order to address these issues and take advantage of the room for growth. Manufacturers can accomplish precise, effective, and economical sorting processes by adopting cutting-edge technology like computer vision systems, machine learning algorithms, and robotics.

The implementation of a sorting mechanism based on color attributes will be the main goal of the sorting operation in this project. Cameras and sensors can be used by automated color sorting systems to precisely detect and group products according to their color characteristics. In order to achieve a high level of accuracy and consistency, machine learning algorithms can be trained to distinguish various color variations, hues, and patterns. These systems can work quickly, allowing quick product sorting without sacrificing output or quality. Manufacturers may considerably reduce their dependency on manual labor by automating color sorting, which will result in cost savings, increased operational effectiveness, and better resource allocation.

As a result of embracing automation in color sorting, production processes will be optimized while also freeing up human resources to concentrate on higher-value jobs, encouraging innovation and promoting corporate success.

3. Objectives

The project's goal is to fully automate product sorting, specifically color-based classification, by employing cutting-edge technology like industrial robotics and real-time object identification. The project's goal is to create a fully integrated system that integrates deep learning algorithms, a 6-DOF industrial robot arm, and IoT connectivity to provide highly precise and effective real-time color identification and sorting. Not only will this simplify the sorting process, but it will also decrease labor intensity and increase overall productivity. By putting this automated solution into practice, it shows how cutting-edge technologies may transform conventional industrial processes and encourage the adoption of smart manufacturing methods.

The aim of this project is to increase the real-time color identification and sorting process' accuracy, efficiency, labor intensity, and IoT connectivity.

Objective 1: Improve accuracy

A strong color-detecting system is created utilizing deep learning methods to attain improved accuracy. To evaluate the real-time video feed and precisely identify the colors of the objects, an object detection model will be used.

Objective 2: Enhance efficiency

Utilizing a 6-DOF (degree of freedom) industrial robot arm that can mimic human arm movements would increase efficiency. The replacement of manual labor with this sophisticated robotic system will allow for quick and accurate pick-and-place operations. Path planning strategies will also be used to optimize the robot's mobility and shorten the duration of the entire procedure.

Objective 3: Reduce labor intensity

The goal is to automate the color sorting process as little as possible to reduce manual intervention. To obtain color data from the object detection model, a Python coding mechanism will be created. This algorithm will carry out path planning and decision-making, delivering precise instructions to the robot arm to efficiently sort the objects to their intended positions.

Objective 4: Enable IoT connectivity

For systems and devices to function effectively, networking and communication must be seamless. The Internet of things (IoT) is included into the system to do this. using in particular the Message Queuing Telemetry Transport (MQTT) messaging protocol, which is renowned for its easy-to-use and effective communication features. A reliable and expandable communication infrastructure is created by utilizing MQTT, allowing for real-time data transfer and analysis for better decision-making and control.

4. Literature review

4.1. Industrial robotic arm

A typical industrial robotic arm comprises two significant components and six degrees of freedom. The robot may be programmed by the operator using the controller, a computer that powers its motor, and the teach pendant[6]. The teach pendant is a wired or wireless portable human-machine interface (HMI) used to manage input/output signals, develop modules to carry out a series of tasks, control the joints, regulate the location of the end-effector, and control the speed of the robotic arm[7]. A teach pendant that is based on software is required to realize a remote model. This has been the subject of numerous studies and developments, and they are frequently referred to as soft teach pendants or virtual teach pendants since they use software to represent physical teach pendants on personal computers[8,9]. A more straightforward method of controlling a robotic arm would be used as these programs are not open-sourced and the design of the teach pendant software user interface is outside the scope of the project.

4.2. Internet of Things (IoT) and its application

These days, a lot of IoT applications for robots are built around mobile devices and web-based software. Mobile phones are becoming the preferred user interface due to their portability, availability, flexibility for application creation, and ability to operate devices through wireless communication. Three layers could be used to decompose an IoT architecture[10]. Sensors are utilized in the physical layer of the perceptive layer to detect physical parameters in the outside world. The transport layer known as the network layer is in charge of information exchange and transmission. The user interface for the application layer, which contains calculation processes to deliver particular services to the user, gives the user control. With this fundamental design, many researchers have accomplished wonderful things, such as the Internet of things (IoT) assisted smart home[11], web-based application-controlled mobile robots[12], and scanning and inspection robots for the mining industry[13], which reduces human casualties in dangerous workplaces.

4.3. Internet of Things (IoT) based robotic arm control

Existing initiatives that use mobile applications to remotely control robotic arms are comparable[14–17]. Signals would be delivered or received via a web server such as a web server hosted by the microcontroller itself (e.g., ESP32) through the soft access point technique (Access Point - AP) or an

IoT development platform such as Blynk[15,18]. Microcontrollers are used to control the signal input/output. Robotic arms would be controlled using a user interface on mobile devices with internet access. The operator must be present in person in front of the robotic arm for the majority of these projects, though. Instead of eliminating the requirement that operators be physically present at the workplace, these ideas advocate the idea of remote control as the removal of the necessity to be connected to a fixed terminal. Given that it has been demonstrated that a robotic arm can be controlled using IoT, this is a nice place to start.

4.4. Color-based sorting object

The use of color-based item sorting in several sectors has increased recently. This calls for a system that proposes a way to identify the color of the object and sort such things using image processing. Once the object has been detected, signals will be sent to the sorting system, and depending on the application, the system will decide what to do next. A typical vision system used for object or color detection typically consists of hardware for picture collection and processing as well as software that contains the images and algorithms for analysis[19]. These days, artificial intelligence is widely used, and to distinguish between different colors of objects, software algorithms frequently use deep learning neural networks like the convolutional neural network (CNN).

5. Methodology

A systematic and structured methodology was used to develop the color-based object sorting system using a robot arm and real-time object detection. This part provides an overview of the essential procedures and techniques used throughout the project. It covers hardware configuration, the creation of object detection models, real-time color detection, Python code integration, and dashboard creation. **Figure 1** shows the workspace setup of the project.

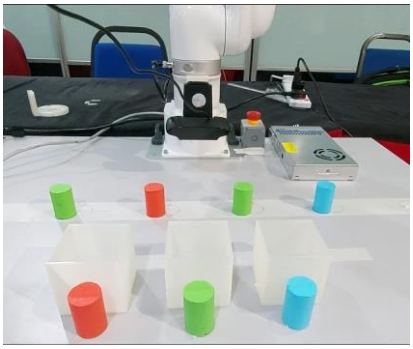

Figure 1. Project workspace setup.

5.1. Hardware setup

A flexible robotic system capable of accurate and sophisticated movements in six different axes is a 6-DOF (degrees of freedom) industrial robot arm. In the instance of this project, it will be used to mimic the movement of a human arm to perform a pick-and-place operation to sort the cylinders into their appropriate locations. Its versatility and agility make it suited for a wide range of industrial applications. The Selangor Human Resource Development Centre's (SHRDC) UFACTORY Lite 6 collaborative robot arm will be employed in this project since it meets industrial standards.

5.2. Robot arm control

The UFACTORY Lite 6 SDK, a Python software development kit made available on GitHub by

UFACTORY, is used to control the UFACTORY Lite 6 robot arm. Its movements and functionalities were programmed using this SDK (Software Development Kit), which was a potent tool. A Python script imports the Python API, and libraries are used to program the robot arms' movements.

5.3. Object detection model development

Object detection identifies and separates different color cylinder variants and produces results in real-time. In this project, the workflow for creating an object detection model for color sorting is finished using the open-source software Integrated-Vision-Inspection-System (IVIS) created by SHRDC (Selangor Human Resource Development Center). Model training, model deployment, and dataset preparation are some of the crucial steps in this approach.

Creating sets of photos exhibiting the various cylinder colors is a step in the dataset preparation process. There are three different labels for each image: red, blue, and green. The prepared dataset is used to train the object detection model using the IVIS program during the model training phase. In order to increase the model's accuracy in classifying the various colors represented in the dataset, this method entails adjusting the hyperparameters of the model.

After model training is finished, the deployed trained model is used to detect colors in real-time on the cylinders. The model receives real-time video input from the robot arm's camera, analyzes the incoming frames, and determines the color of each cylinder it detects. After that, the output is shown in real time, allowing the robot arm to sort the cylinders appropriately.

5.4. Real-time color sorting

Real-time object identification and real-time data transfer are what enable real-time color sorting. Real-time color detection results are published and subscribed to using the MQTT protocol.

5.5. Dashboard development

A simple Node-RED dashboard is developed using the dashboard UI (User Interface) node as shown in **Figure 2**.

The robot arm positions itself above the cylinder to position the camera and start the workflow. The object detection model then processes the live video feed, collecting the objects' color data. MQTT is used to provide seamless delivery of color data. The object detection program serves as the publisher, disseminating the color information under a certain heading. The Python code simultaneously assumes the role of the subscriber and subscribes to this subject in order to obtain the color information in real time. The Python method heavily relies on the incoming color information for path planning and decision-making. The program uses this data to generate exact robot orders that allow the robot arm to pick up the cylinder and place it precisely where it needs to be. Data interchange between various devices and systems is gathered throughout the entire operation and published to a dashboard. This dashboard allows for speed optimization, real-time monitoring of the operation's status, and quality control of the sorted items. As long as this automated color sorting mechanism is in place, the operation will be efficient and accurate with little need for manual intervention. The methodology described provides a fluid workflow by fusing dashboard monitoring, object identification, robot arm control, and color sorting into a single, integrated system.

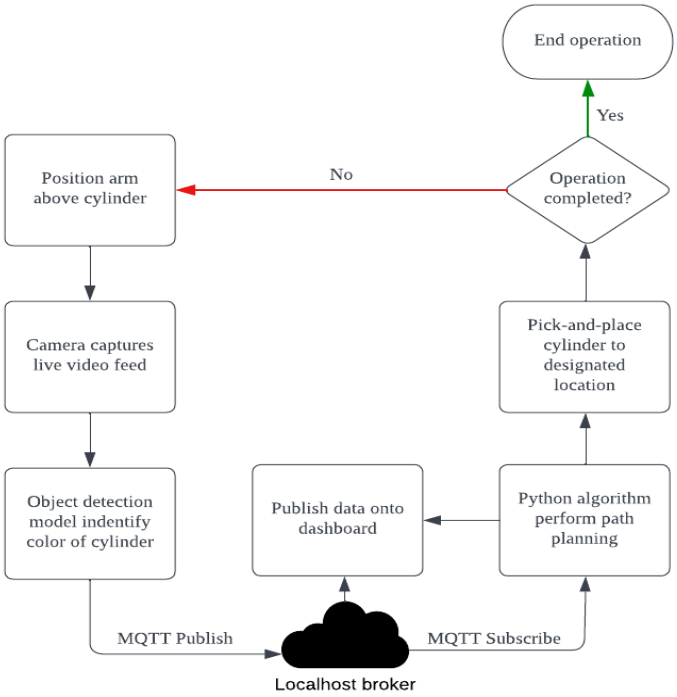

Figure 2. Workflow diagram of color sorting system.

6. Result and discussion

6.1. Robot arm control

A software platform called UFACTORY Studio offers a user-friendly interface for directing the robot arm. For the cylinder sorting task, this is used to establish the robot arm trajectories and waypoints to the cylinder locations. This is accomplished by using the base coordinate control buttons in combination with the manual instruction capability in "Manual mode". **Figure 3** illustrates the UFACTORY control UI using software (UFACTORY Studio)

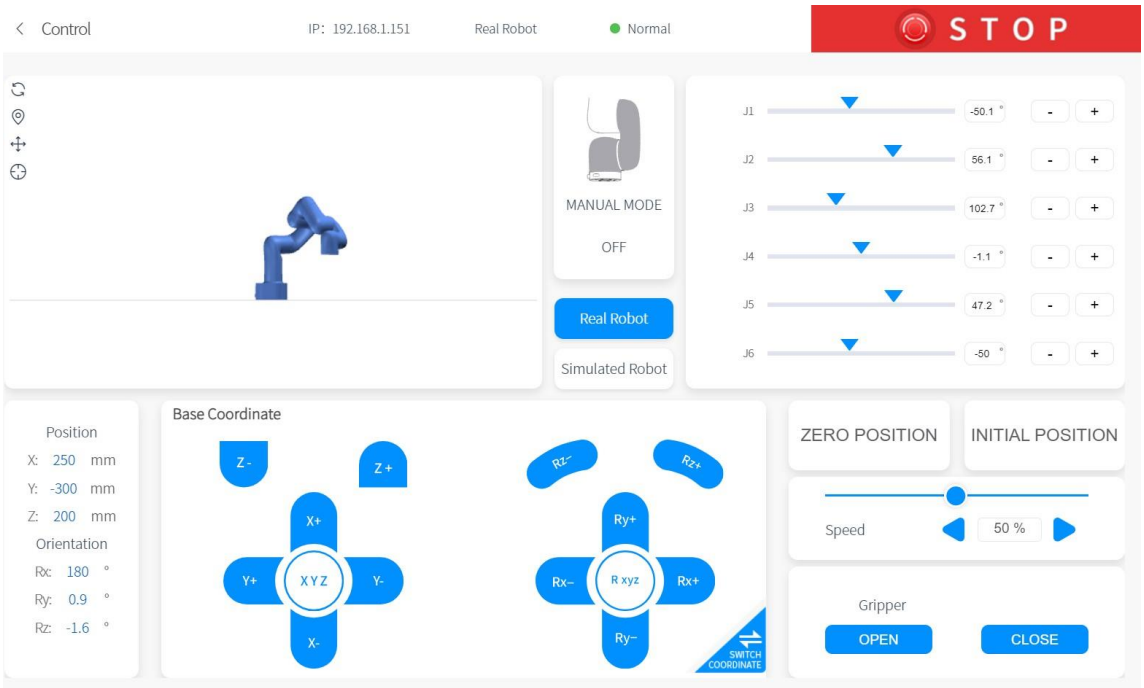

Figure 3. UFACTORY control UI.

To produce robot arm commands in a Python script, coordinates and trajectories are recorded throughout the procedure. MoveL and MoveJ are two of the instructions that can be utilized with conventional industrial robot arms. The appropriate robot arm movements are exported as .py files after being defined in UFACTORY Studio.

As long as the xArm Python SDK is appropriately imported, this exported script can be used in any Python-supporting IDE. The script can now be integrated with other systems using this technique. The exported Python script is then further altered by the addition of functions that process the results of the object detection and determine where to put the cylinder.

6.2. Camera mount design

The Logitech C920 Pro HD (High Defination) webcam was used, which has a relatively large body. To accommodate its large size, a custom camera mount was designed, taking inspiration from the UFACTORY xArm camera stand[20], featuring a 2 mm thin camera mounting plate positioned between the end-effector and gripper of the robot arm. This 3D-printed camera mount ensures a secure attachment of the webcam and is positioned to capture a clear view of the cylinder during object sorting operations. **Figure 4** shows the 3D CAD drawing of camera mounting plate.

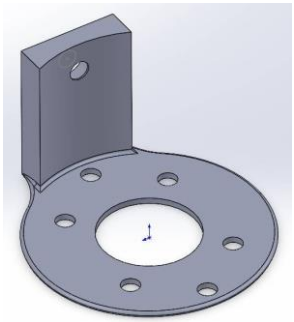

Figure 4. 3D CAD drawing of camera mounting plate.

Figure 5 shows the installation of the camera mounting plate on end-effector.

Figure 5. Installation of camera mounting plate on end-effector.

6.3. Object detection model

6.3.1. Dataset preparation

Three distinct methods were utilized to collect the cylinder dataset, as shown in **Figure 6**: (1) Top image of the cylinder as seen from the robot arm right above it, (2) a close-up photo of the cylinder taken with a handheld camera, and (3) a partial top view of the cylinder as seen from the robot arm as it approaches. Images of (1) and (3) were first gathered since they show the views that were photographed during the real operation. However, it was noted that there were considerable shape and scale differences between these two perspectives, making it difficult for the object detection model to

precisely recognize the shape of the cylinder. While the image (3) shows the side view of a more rectangular shape, image (1) shows a more circular and compact cylinder. As a solution, image (2) was added as a reference point to help the model be more resilient to changes in size and shape. These methods guaranteed thorough coverage and a variety of viewpoints during the dataset collection procedure.

441 datasets were collected. The dataset was equally divided among the three colors, with red, blue, and green cylinders having a ratio of 164:148:129. Due of their greater sensitivity to changes in lighting, the red-colored cylinders were given extra attention, which led to more obvious variations in their color look. Following the best labeling procedures, these photos are subsequently labeled using Label Studio[21].

Figure 6. Example of datasets collected.

6.3.2. Model training

It takes time to train an object detection model, and the dataset's quality has a big impact on how well it works. Several measures and design considerations were used to address dataset quality and speed up the training process in recognition of the importance of both of these variables.

The base model used during the training of the object identification model is displayed in **Table 1** and was taken from the TensorFlow Object Identification Model Zoo[22]. Three important criteria model architecture, model correctness, and computational effectiveness were used to choose the basic model. The models were assessed using two metrics: COCO mAP (mean average precision) (an accuracy metric on benchmark datasets, where higher is better) and latency (speed during deployment in milliseconds, where lower is better).

Table 1. Object detection base model speed vs. performance.

	Model name	Speed (ms)	COCO mAP (mean average precision)
1	CenterNet HourGlass104 512 × 512	70	41.9
2	CenterNet Resnet50 V1 FPN 512 × 512	27	31.2
3	CenterNet MobileNetV2 FPN 512 × 512	6	23.4
4	EfficientDet D1 640 × 640	54	38.4
5	Faster R-CNN ResNet50 V1 640 × 640	53	29.3
6	SSD MobileNet V2 FPN Lite 640 × 640	39	28.2

In this project, SSD (Single Shot Detector) MobileNet V2 FPN (Feature Pyramid Network) Lite 640x640 has been used as the basis model. With this paradigm, accuracy, and computing complexity are balanced. It is important to note that CenterNet ResNet50 V1 FPN 512x512 performs better. Despite this, SSD MobileNet is chosen because it has a one-stage detector as opposed to CenterNet's two-stage detector, which drastically cuts down on training time.

The same data set was used to train both SSD MobileNet (a one-stage detector) and Faster R-CNN (a two-stage detector) to see if there would be a significant gain in model accuracy justifying the longer training time. After testing, the two models' deployment performance did not significantly differ from one another. The training procedure was finished in an additional 20 min by Faster R-CNN. The trade-off between training time and model accuracy leads to the conclusion that SSD MobileNet is still the better option before moving on to more computationally intensive models like EfficientNet and CenterNet.

The object detection model's training setting is shown in **Figure 7**. The dataset is trained without prior augmentation because the dataset gathering takes many characteristics into consideration. When used judiciously, dataset augmentation approaches can improve the object detection model's robustness. When specific qualities and limits weren't taken into account, it can, however, inject unrealistic or incorrect data into the dataset. For instance, color information may be distorted and the color of the cylinders may be misrepresented by transformations such as hue shifts or white exposure modifications. IVIS is not used during training because of its restricted controllability over augmentation functions. **Figure 8** indicates the result prediction through validation and test data sets.

Training Session Name: train_1

Training Description:

Dataset List: FYP_cylinders_1

Partition Ratio: training : validation : test -> 0.8 : 0.1 : 0.1

Partition Size: training : validation : test -> 353 : 44 : 44

Selected Model Name: SSD MobileNet V2 FPNLite 640x640

Model Name: model_1

Batch size: 4
Is changing default lr: False
Num train steps: 5000

Loss/classification loss: 0.065963
Loss/localization loss: 0.024571
Loss/regularization loss: 0.12443
Loss/total loss: 0.21496

Figure 7. Object detection training information.

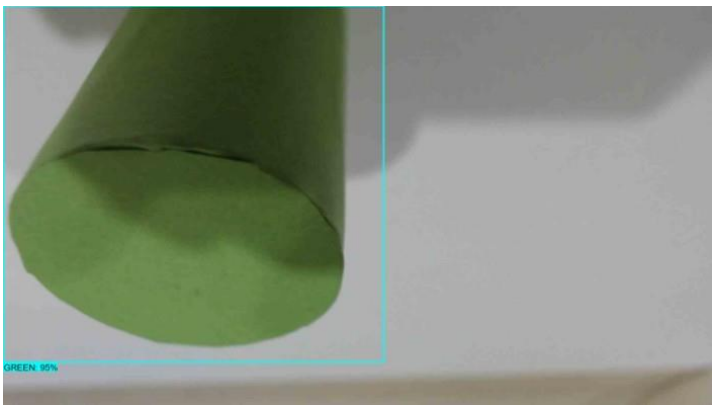

Figure 8. Result prediction through validation and test data sets.

The cyclical process of training, reviewing findings, and improving the dataset is sped up by cutting training time. The performance of the model can be gradually improved with this method.

When tested against 88 validation and test set photos, the model consistently produced correct answers using this iterative methodology, with an average confidence level of about 92%. This high degree of assurance is essential for precise and dependable color sorting. To get the model to where it is now, 441 labeled datasets were combined. Furthermore, a low total loss of 0.21496 is attained. This shows that the model has little inaccuracy and is resistant to changing factors such as variations in the brightness of the environment, cylinder size, and the position of the cylinder within the collected image. It's crucial to remember that other metrics other than total loss should be used to evaluate the success of the model. A model that tends to overfit the input data may actually be more appropriate in some situations where the variables can be controlled and are consistent. This is so that an overfitting model, which ensures accurate and reliable results, can capture the precise patterns of the controlled variables. Because of this, even if the current model has demonstrated good performance, it is crucial to take into account the deployment strategies for the color sorting system when training the object detection model. To make sure the model is appropriate and to maximize its performance, factors including system configuration, operating circumstances, and desired outcomes should be carefully taken into account.

6.3.3. Model deployment

The IVIS software is utilized to deploy the model. The robot arm-mounted USB (universal serial bus) camera must be connected to the PC and input settings must be changed to recognize it as a USB Camera Video camera while supplying the appropriate camera port number in order to make the necessary connection. The model built in the preceding stage is utilized to perform real-time object detection once deployment has begun.

The MQTT protocol's topic "cvsystem/main/publish_frame" is used to effortlessly publish the object detection output. The MQTT broker runs on port 1883 and is locally hosted on the system as "localhost". For message transport, MQTT quality of service (QoS) level 0 is chosen. For real-time communication, where message delivery speed is more crucial than dependability, QoS 0 assures "best-effort" delivery without any acknowledgments[23]. The reliability of message transfer is considered secure since communication is restricted to the local network, which negates the need for higher QoS levels and guarantees effective and timely data delivery. **Figure 9** shows the object-detection model deployment using IVIS software.

Figure 9. Object detection model deployment.

6.4. MQTT connection

The Eclipse Paho MQTT Python client module is used to create the MQTT connection. With the help of a client class provided by this code, applications are now able to connect to an MQTT broker and publish and subscribe to messages via topics. It is used in this project to import color detection data from IVIS into a Python script and produce commands for the robot arm to pick and place the cylinder in the desired area.

The loop() function is essential for starting and maintaining communication with an MQTT broker when using the Paho Python client. The underlying network connectivity, message receipt and publication, and maintaining the MQTT client's connection are all handled by this built-in function. The loop() method, on the other hand, is a blocking function, which means it will continuously watch for incoming messages and respond to network events. This can prevent the main program, which is used to control the robot arm, from running. Although this problem can be solved by putting the loop() function in a different thread, mqtt_listen_once() has been created as a special function for ease of use.

By setting up the MQTT client, connecting to the localhost broker at port 1883, subscribing to the IVIS result subject, and waiting for a single message on that topic, the mqtt_listen_once() code accomplishes the above tasks. After receiving a message, the function cuts off communication with the broker. With this method, information is only received when it is required, removing the need for constant listening and ensuring that the main program can keep running. The program's workflow is made more streamlined and effective thanks to this feature.

6.5. System integration

Python code created for controlling a robot arm and subscribing to MQTT messages was integrated into a single piece of code. The python shell node is used to execute this Python script. The results of the sorting procedure are output onto the console terminal by the python script as it runs. **Figure 10** indicates the architecture of the Node-RED flow used in the project.

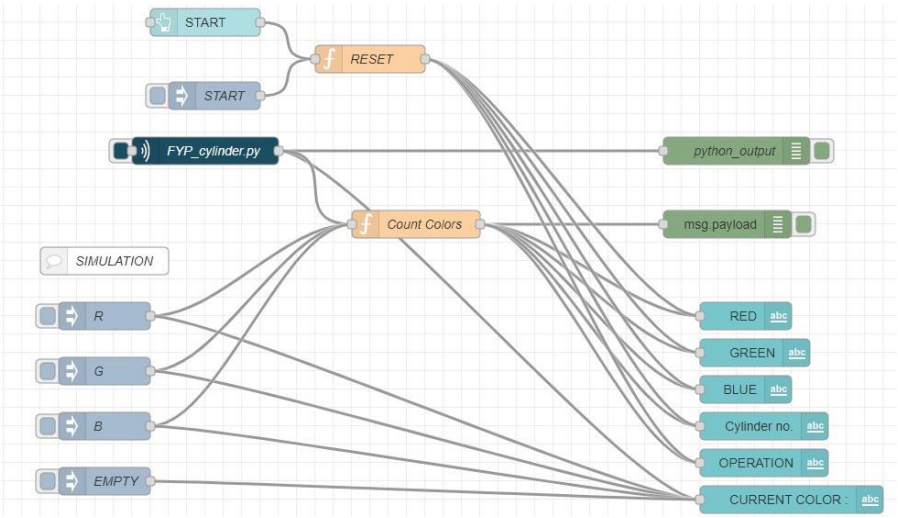

Figure 10. Node-RED flow.

Figure 11 shows the Node-RED dashboard for control and monitoring of the robotic arm.

Figure 11. Node-RED dashboard for control and monitoring.

The 'count colors' function node processes this result to update the flow variables value, which is subsequently transmitted to the 'text UI' nodes for display on the dashboard. The dashboard has two buttons: "start" and "RESET" that can be used to reset all results to zero and get ready for new operations, respectively. A user-friendly dashboard and smooth IoT connectivity are two major advantages of integrating Node-RED.

6.6. Findings

It was observed that the gripper functions offered in the xArm Python API (application programming interface) could not be used, leading to a timeout error and program termination, while developing a Python script for controlling a robot arm. A number of gripper-related functions, including "get_gripper_version()" and "set_gripper_enable()," were found to be either missing or incomplete when the library's source code was examined. But utilizing its integrated Python IDE, the UFACTORY Studio software effortlessly controlled the gripper.

Further research found that there were differences between the xArm Python API contained in the robot arm and the one that was made available on GitHub, including different function names. The library files were attempted to be accessed by SSHing (secure socket shell) into the robot arm's IP address, but access was password-protected, necessitating communication with the Chinese manufacturer.

In order to solve these problems, every function present in the library's source code was examined until a solution was discovered, enabling control of the vacuum gripper through functions created for the gripper. This strategy, nevertheless, has several drawbacks. First off, until the power source was shut off, the gripper was always in active mode because no sleep or stop capabilities were available. Second, because the functions were not intended for this use, the robot arm's auto collision detection would not be activated if the gripper ran into an obstruction. In order to prevent harm to the robot arm, the operator had to exercise caution and be ready to hit the emergency stop button.

6.7. Future work

At the moment, the robot arm is manually instructed to find the cylinder's location, which includes putting the object in the right spot and orienting it correctly. The procedure could be made more logical, but it is not. Automating the position-detecting method to do away with the necessity for manual education is one potential future development.

In real-world applications where products need to be sorted and are frequently kept in containers or put on moving conveyor belts, automating location detection would be especially helpful. A real-time automatic location detection method is required in these situations to dynamically update and

adapt to environmental changes.

To do this, an overhead camera that offers a top-down view of all the products can be put directly above the workspace. To identify and pinpoint the center point of each product, the recorded photos can then be analyzed using image processing techniques, such as a blob-detection model.

Blob detection is a well-liked method of extracting and identifying regions of interest in an image. Blobs are areas with comparable characteristics, such as color or intensity, that stand out from the backdrop while an item is being detected. The unique properties of the products can be recognized as individual blobs by applying a blob-detection model to the images taken by the overhead camera.

The centers of the blobs that stand in for the items can then be identified. This data gives the robot arm the precise geographical coordinates it needs to find and organize things. The robot arm can autonomously carry out sorting activities without the need for manual teaching of coordinates by applying blob detection and examining the center points of the items.

The system may be made much more efficient and flexible by adding an overhead camera and using blob detection for location detection. In handling scenarios where products are held in containers or moved on conveyor belts, the camera's continuous capture of the top view enables real-time updates of the object's orientation and location information.

Although the idea seems feasible, the execution of automating location identification would be subject to specific needs and process limitations. One solution is the offered method, which makes use of an above camera and blob recognition, however, it might not work in all circumstances. To achieve the intended automation, different scenarios could call for different methods or tools.

As an illustration, under some circumstances, different kinds of sensors like proximity sensors, laser scanners, or depth cameras may be better suitable for determining the position and orientation of objects. These sensors can offer accurate distance readings or 3D data that can be used for automated object detection and localization.

Additionally, a different strategy can be necessary depending on the type of objects being sorted or the operational setting. A vision system that can read ARUCO (Augmented Reality on OpenCV) markers and determine the distance of each object from them, for example, might be used. ARUCO markers could be set all around the region. In order to achieve successful and efficient automation, a personalized method must be devised while taking into account the requirements and restrictions.

7. Conclusion

In conclusion, by incorporating cutting-edge technologies, the project has effectively met its goals of creating an automated color sorting procedure. The accuracy and efficiency of color sorting operations have improved thanks to the use of a strong color detection model and a 6-DOF industrial robot arm. Through this study, the potential for increased productivity decreased labor intensity, and greater efficiency in industrial settings has been demonstrated. The conceptual idea of achieving a color sorting operation through an industrial robot arm and object detection model has been confirmed. This serves as the starting point for future developments in industrial automation. The successful fusion of these technologies has created new avenues for further study and development in order to develop a more complex strategy for overcoming the difficulties encountered during the color sorting process.

Author contributions

Conceptualization, KHH and WSC; methodology, ZCO; software, ZCO; validation, ZCO, KHH and WSC; formal analysis, ZCO; investigation, ZCO; resources, WSC; data curation, ZCO; writing—original draft preparation, ZCO; writing—review and editing, KHH; visualization, KHH; supervision, KHH and WSC; project administration, KHH; funding acquisition, KHH. All authors have read and agreed to the published version of the manuscript.

Conflict of interest

The authors declare no conflict of interest.

References

1. Harris T, Pollette C. How robots work. Available online: https://science.howstuffworks.com/robot2.htm#:~:text=The%20computer%20controls%20the%20robot,same%20movement%20over%20and%20over (accessed on 10 November 2023).
2. Palmai K, Smale W. The robot chefs that can cook your Christmas dinner. Available online: https://www.bbc.com/news/business-59651334 (accessed on 10 November 2023).
3. Fairchild M. Top industries using robots. Available online: https://www.howtorobot.com/expert-insight/top-industries-using-robots (accessed on 10 November 2023).
4. Fu S, Bhavsar PC. Robotic arm control based on Internet of things. In: Proceedings of the 2019 IEEE Long Island Systems, Applications and Technology Conference (LISAT); 3 May 2019; Farmingdale, NY, USA. pp. 1–6.
5. IOT based robotic arm project using NodeMCU. Available online: https://iotdesignpro.com/projects/iot-based-robotic-arm-using-esp8266 (accessed on 10 November 2023).
6. Available online: https://www.iaasiaonline.com/controlling-robotic-arms-2/ (accessed on 10 November 2023).
7. Lin HI, Lin YH. A novel teaching system for industrial robots. Sensors 2014; 14(4): 6012–6031. doi: 10.3390/s140406012
8. Software pendant. Available online: https://www.yaskawa.eu.com/products/software/productdetail/product/software-pendant_1673 (accessed on 10 November 2023).
9. Jan Y, Hassan S, Sanghun P, Jungwon Y. Smartphone based control architecture of teaching pendant for Industrial manipulators. In: Proceedings of the 2013 4th International Conference on Intelligent Systems, Modelling and Simulation; 29–31 January 2013; Bangkok, Thailand. pp. 370–375.
10. Sethi P, Sarangi SR. Internet of things: Architectures, protocols, and applications. Journal of Electrical and Computer Engineering 2017; 2017: 1–25. doi: 10.1155/2017/9324035
11. Li XQ, Ding X, Zhang Y, et al. IoT family robot based on raspberry Pi. In: Proceedings of the 2016 International Conference on Information System and Artificial Intelligence (ISAI); 24–26 June 2016; Hong Kong, China. pp. 622–625.
12. Jain RK, Saikia BJ, Rai NP, Ray PP. Development of web-based application for mobile robot using IOT platform. In: Proceedings of the 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT); 1–3 July 2020; Kharagpur, India. pp. 1–6.
13. Qadri I, Muneer A, Fati SM. Automatic robotic scanning and inspection mechanism for mines using IoT. IOP Conference Series: Materials Science and Engineering 2021; 1045(1): 012001. doi: 10.1088/1757-899x/1045/1/012001
14. How to make robot arm Nodemcu Esp8266 (access point) IOT project. Available online: https://youtu.be/r7712qCkngs (accessed on 10 November 2023).
15. IoT based robotic arm using NodeMCU ESP8266. Available online: https://youtu.be/ZG2Ek-Z5i1Q (accessed on 10 November 2023).
16. How to mechatronics. DIY Arduino robot arm with smartphone control. Available online: https://youtu.be/_B3gWd3A_SI (accessed on 10 November 2023).
17. Robot arm using ESP32 and smartphone | Complete robot arm assembly. Available online: (https://youtu.be/cVSvg6VQhGU (accessed on 10 November 2023).
18. Getting started with the ESP32. Available online: https://dronebotworkshop.com/esp32-intro/ (accessed on 10 November 2023).
19. Shatwell DG, Murray V, Barton A. Real-time ore sorting using color and texture analysis. International Journal of Mining Science and Technology 2023; 33(6): 659–674. doi: 10.1016/j.ijmst.2023.03.004

20. We make affordable robots for all. Available online: https://store-ufactory-cc.myshopify.com/products/xarm-camera-module-2020 (accessed on 10 November 2023).
21. Nelson J. How to label images for computer vision models. Available online: https://blog.roboflow.com/tips-for-how-to-label-images/ (accessed on 10 November 2023).
22. TensorFlow 2 detection model zoo Available online: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md (accessed on 10 November 2023).
23. T. H. Team, "Try hivemq," MQTT Quality of Service (QoS) 0,1, & 2 – MQTT Essentials: Part 6. Available online: https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/ (accessed on 18 November 2023).