Next generation Industrial Robotics advanced by AI

By

Saddam Hossain

A dissertation submitted in partial satisfaction of the

Requirement for the degree of

Master of Science

In

Engineering – Robotics and Automation

In the

Graduate Division

Of the

University of Salford, Manchester

Committee in charge:

Professor Steve Davis

Summer 2021

Declaration

I declare that I studied at the Department of Robotics and Automation on my dissertation, titled "Next generation Industrial Robotics advanced by AI." The literature has been properly acknowledged in the text, and a list of references has been given.

I certify that:

1. This work was completed entirely or mostly while pursuing a research degree at this university;
2. Any portion of this dissertation that has been previously submitted for a degree or other qualification at this University or another institution has been explicitly stated;
3. When I reference other people's published work, I always make sure to credit them;
4. I always mention the source when I quote from another people's work. This thesis is completely my own work, with the exception of such citations.
5. I've acknowledged all major sources of assistance;
6. In cases when the thesis is based on collaborative effort by myself, I have specified precisely what was contributed;

Signed: Saddam

Date: 30-09-2021

Acknowledgements

This dissertation is the result of my work at the University of Salford during the period from June 2021 till September 2021.

First and foremost, I would like to thank my supervisor Prof. Steve Davis. His continuous guidance and the level of trust he granted me made the journey all the more exciting and joyful. The dissertation conducted during 3 months wouldn’t have been possible without the enormous support of the following classmates. I thank you all wholeheartedly and sincerely wish your future success in your private and professional life.

Saddam Hossain

September 2021

Abstract

[1] A collaborative robot is something that is increasingly interesting for manufacturers. But in today's market, a collaborative robot is relatively new. The aim of this project was to examine how an implementation process of collaborative robots in Grocery companies could look like. there will be design a collaborative robot and integrate in the simulation system which will do work for a Grocery products distribution system. where will be used deep learning and vision technology to inspect the product and put the product specific place. e. The goals of this dissertation by cobot and AI are based on the identification and diagrammatical process of a collaborative robot, and the identifying of actors and abilities required for effective cobot integration.

Table of Contents

List of figures

List of tables

Abbreviations

ROI = Return on Investment

AI = Artificial Intelligence

Cobot = Collaborative Robot

DL = Deep Learning

Chapter 1: Introduction

1.1 Motivation

New robotics applications involve artificial intelligence. In industrial robotics, AI-enabled Cobots, self-driving cars, and non-piloted drone operations are just a few of the many new applications that are being developed for the technology. [2] According to Fortune Business Insights, industrial robots combined with AI is expected to drive market growth by more than 15 percent in the coming years, reaching USD 66.48 billion by 2027. New cooperation and workflow models are bringing people and robots together in industry as boundaries between human tasks and robotic capabilities decrease.[3] The majority of applications, from materials handling to assembly, may now be automated. Sensors such as touch and vision systems have also been developed to expand the number of robot applications and to make them easier to deploy. In addition, the processing capacity of robots has increased, giving them artificial intelligence (AI) capabilities. Identifying and evaluating development methods for highly AI competent industrial robots in which topic knowledge is constantly changing is the goal of this dissertation. This chapter will begin by describing the study's background and context, followed by the contribution, the dissertation objectives, the justification and finally, the limitations.

1.2 Background

General Motors' first industrial robot was initially deployed at a plant in 1954, and seven years later was used for spot welding and die-casting. It has been utilised in sectors ranging from manufacturing to agriculture as a way of increasing efficiency and decreasing costs, as well as increasing income. [4] In 1967, Svenska Metallverken in Upplands Väsby installed the first robots in Sweden, and they were also the first in Europe. The robots carried out mundane tasks such as selecting in and out. General Motors purchased the Unimate #001 prototype in 1961 and put it to work in their New Jersey car plant.

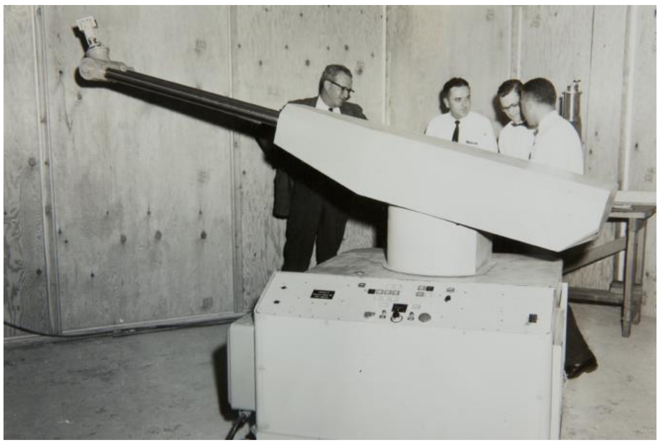


Figure 1. 1 The first industrial robot (Unimate). Image: robotics.kawasaki

This was followed in 1969 by the announcement that GM'a newly renovated facility in Ohio was producing 110 vehicles per hour with the help of a fleet of Unimate 1900 robots. As a result of these unprecedented speeds, no other automobile factory at the time could match them. [4] In 1972, Europe followed suit, with Fiat establishing a spot-welding line using robots. When ASEA (later ABB) bought Trallfa in 1985, the painting robots and ASEA's industrial robots worked nicely together. During this time, improved sensors and basic machine vision systems were introduced, laying the groundwork for the modern industrial robot.

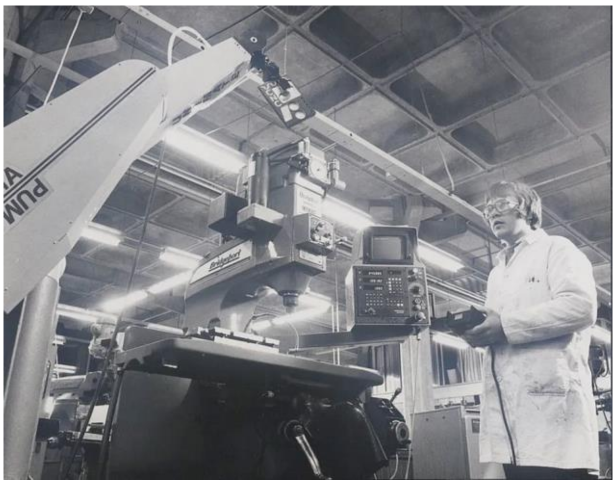


Figure 1. 2 The Unimate 500 and Puma 560 Industrial robots in 1986. Image: Wiki Commons

[5] Industrial robots now have the capacity to identify and track manufactured components through assembly lines using force sensors and lasers. A human-like sensation of touch was given to the robots by the lasers. This radically changed the way robots interacted in an industrial setting. This led to robots evolving from basic mechanical devices programmed to do repetitive tasks into more complex machines with "limited intelligence," according to some. In 1992, FANUC robotics built the first intelligent robot prototype.[5] Advances in software have been a major driving force in industrial robots since the mid-2000s. Recent advances in machine learning (ML), artificial intelligence (AI), and other related areas have enabled robots to learn, improve and make choices independently of humans.

[6] At Linatex, the first cobot was deployed in 2008. Rather of putting the robot behind a safety fence, this Danish plastics and rubber company opted to place it on the floor. It was possible to configure the robot using a touchscreen instead of employing a programmer, which saved the company money. COBOTS, a relatively new kind of robot, are intended for safe interaction with people, whether near by or in direct touch. Computer vision and force-limited joints allow them to identify people in their surroundings. It is common for cobots to be smaller and lighter than conventional industrial robots, and to be readily moved and trained for particular jobs. [5] Recently, the usage of collaborative robots has exploded in sectors such as manufacturing, construction, and healthcare. According to ABI Research, the market for cobots is projected to reach $1 billion by 2020, with an estimated 40,000 robots in operation across the world.

1.3 Contribution

AI-enabled collaborative robots are on the increase, as are a growing number of creative application cases. [3] Individual industrial robots, on the other hand, need the presence of safety equipment – such as fences with gates linked to the system for safety – and function entirely without direct interaction with humans.However, Use cases are developing in which people and robots engage together on tasks as boundaries between human activities and robotic capabilities decrease — going beyond the gated activities of last-generation industrial robots. In contrast, [7] collaborative robots do not need safety barriers for safe operation, they are more common than stand-alone ones.

[8] When it comes to automating business processes and automating workflows across hybrid network settings and different technologies, there are still basic difficulties.[9] [7] Slowly but surely, the traditional industrial sector is moving towards the intelligent era of the future. Today, machine vision technology has taken on a prominent role due to its visual benefits. [9] Intelligent industrial growth is significantly aided by it As with human vision, machine vision relies on visual sensors to operate. In the beginning, instead of using human eyes, it collects pictures. Transmit the pictures to the computer after that, and it’s done work. Finally, relevant information is utilised in real detection, measurement, and control once it is processed and analysed.

As a result,[8] Assuming a moderate pace of knowledge growth, the current research is insufficient for Cobot in which critical information and abilities are continuously and quickly developing so that robotic activities with or near human employees are more safe and productive

1.4 Objectives

This dissertation seeks to learn how AI is utilised in next-generation industrial robots, as well as to develop a suitable control system using the robot simulator CoppeliaSim software. This job entails a variety of duties, including designing, testing, and building a robot, as well as modelling and controlling the robot system.

The main Objectives of this dissertation are:

1. Study the performance of the Cobot and AI technology.
2. An overview of collaborative industrial scenarios.
3. CoppeliaSim simulator Creation As a testbed for created algorithms and scientific ideas, this virtual world will be utilised. It will also be utilised as a soft copy for the final product.
4. Vision sensors feedback analysis technique and possible application of information obtained from digital image processing.
5. Proximity sensors are used to detect the presence of Inquiring into and developing a technique for object identification and recognition.
6. Implementation of Cobot Inverse Kinematics, which is capable of executing the guiding system based on vision sensor feedback.
7. To compare and contrast these products identification and selection in terms of their output from deep learning classification and integrated vision system.

1.5 Justification

This study will aid in the development of a warehouse product distribution system that will use a cobot equipped with vision and proximity sensors. We'll also look at how convolutional neural networks may help with the functioning of a cobot product distribution system. The cobot system's primary function is to classify items delivered on a conveyor belt into appropriate categories for optimal product distribution. Finally, This will contribute to the existing research deficit in this sector and offer real-world benefit to warehouse-related businesses.

1.6 Limitations

This project cannot be physically tested since it was created using simulation software. We also utilised certain design and integration stages, as well as an API connection with the CoppeliaSim programme, which aids in the system's development.

1.7 Structure

In Chapter One, the study's background has been described. In addition to identifying the study's aims and questions, the usefulness of such research has also been discussed. We've also spoken about some of the study's shortcomings.

In Chapter Two, as part of the study, important methods and strategies for developing skills in fast-moving sectors of Cobot, particularly those that are technology heavy, will be identified.

In Chapter Three, a theoretical framework will be introduced. We develop a Cobot for Grocery products distributed system where will be discussed about methodology how to make a cobot and integrate in the simulator software also explained all the elements and part about this project.

In Chapter Four, will present the result of vision system and Cobot work output simulation and visualization. In addition, there will be including the experimental verification result resources.

In Chapter Five, discussion part discusses about the output of results those we found in results part. In addition, acknowledge the limitations of study this dissertation. Finally, discuss about the recommendations from implementation and future research.

In Chapter Six, in conclusion part discuss to overall findings in relation to the dissertation aims. In addition, how this study will contribute to the field of this topic.

Chapter 2: Literature review

2.1 Introduction

[10] Artificial intelligence, robots and the like are certainly changing our industry, as well as other sophisticated technology.[11] Furthermore, Advanced robots have higher sensing, systems are integrated, adaptability and mobility compared to ordinary robots. [10] Increasing industrial competitiveness will be driven by collaborative robotics and AI. Human-machine systems really offer many new possibilities which raise the need for new qualified experts. Likewise, Manufacturers use sophisticated robotics as an important feature of advanced automation that allows the smart factory to be self-controlled.[11] The BCG Focus, December 2016 research showed that the use of robots that have more advanced capabilities than the industry standard is capable of reducing conversion costs by up to 15%, while in combination with other technologies, process improvements, and structural layout modifications, these savings reach 40%.However, [10] As collaborative robots and AI become more prevalent in industry, their use will help companies maintain their competitive edge. Due to the fact that human-machine systems provide new job possibilities, the need for new qualified workers will grow. The basis of artificial intelligence is the collection of data, which have been carefully chosen and compiled by humans.[12] We may expect a significant effect on robots due to the development of industrial IoT and artificial intelligence. These new maintenance methods, more advanced robotics, and new collaborative robotics technologies are enabled by Industrial revolution 4.0 technology. Besides, [13] Using advanced vision technologies, the robot can recognise objects regardless of their form or arrangement, which eliminates this issue. [11] Software and hardware advancements allow for the use of sophisticated robots with exceptional characteristics. Enhanced human-machine interfacing, smart sensors, and the capacity to handle huge data volumes on the cloud are just

a few examples of technologies that have been significantly improved by cloud computing. Therefore, Even the most difficult programming jobs may be accomplished by frontline workers using interactive or linked interfaces, which means companies are less reliant on specialist suppliers and technical departments.

2.2 Robots in Industry

[14] While it is relatively recent that robots should be included into manufacturing technologies, evidence indicates that there is currently a fast rise in robots used in industry. Robots with a broad range of functions may be found in a number of environments. There are various reasons for use and acceptance:

1. Labour expenses are lower.
2. Boosted productivity.
3. Elimination of hazardous or unappealing occupations.
4. Product quality has improved.
5. Enhanced manufacturing flexibility
6. Waters of less substance
7. Inventory turnover is more controlled and quicker.
8. Reduce material waste while increasing output.

[10][15] This interest comes from robots' contribution to improving job safety, increasing productivity and reducing manufacturing costs. There are industrial robots, medical robots, food production robots, agricultural and construction sector robots. They are useful for a variety of diverse applications, including as welding in the automotive sector, painting the various components of a vehicle, and inspection. The industrial robots have greater efficiency while doing physical labour, and they are often ready to work around the clock. For

businesses, this means that they don't have to pay a team of employees to perform the same task, which saves money in the long run.

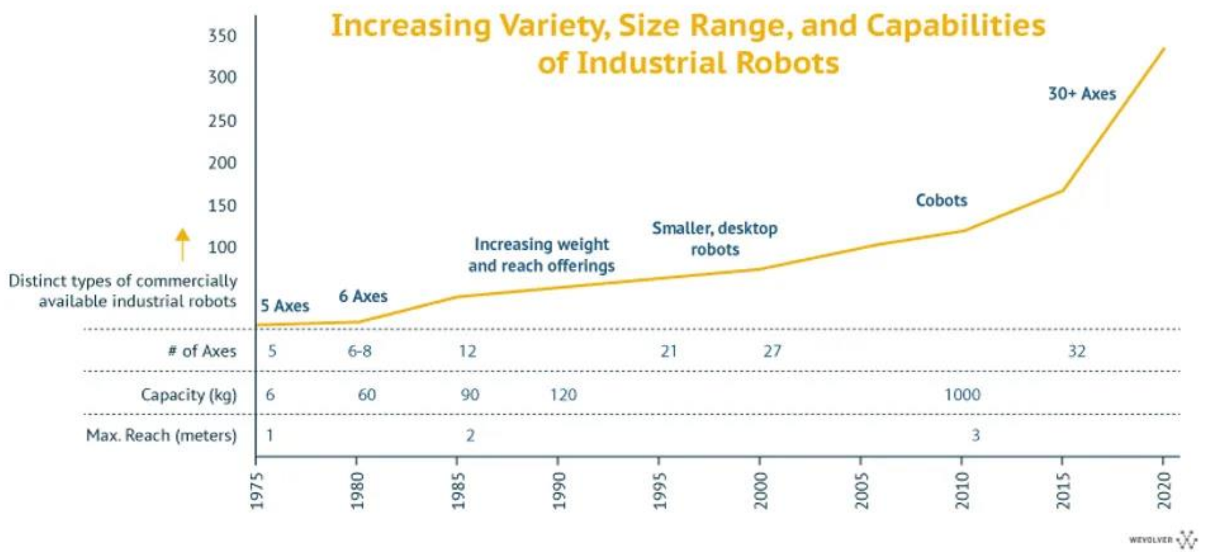


Figure 2. 1 The increase in axes, capacity of robotic arms Source: McKinsey

The increased range and capacities of robots are driving the development of the market for industrial robotics.

2.3 Traditional Industrial Robot

[16] For the last decade, the need for more consistent and higher-volume robot manufacturing has been increasing for industrial robotics, particularly in the aerospace industry (Summers, 2005). Recently, the robot calibration services provided by most industrial robot manufacturers and a few service providers (for example, Dynalog of USA). However, Dechow observed that traditional robots are not intended to work with people directly.[17] Today's robots include safety measures that allow people to work alongside robots without having to actually interact with the robots. Dechow said: "You are allowed to work near any robot, provided that it is accompanied by safety devices that stop it when you come close."

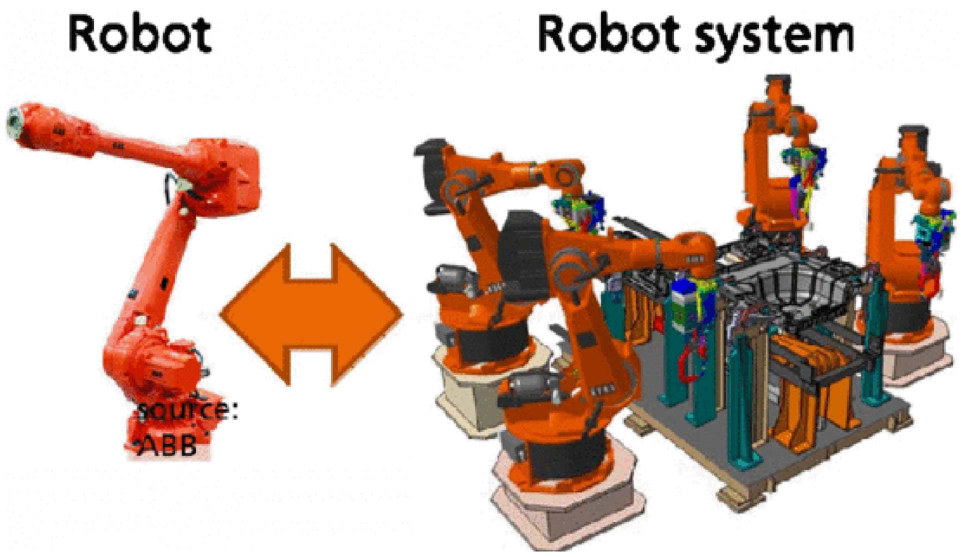


Figure 2. 2 Robot vs robot System Source: [18]

Robot are commonly grouped by consideration of the following characteristics:

1. Manipulator Configuration
2. Actuator types
3. Method of programming
4. Sensor System
5. Controller design
6. End-effector and task

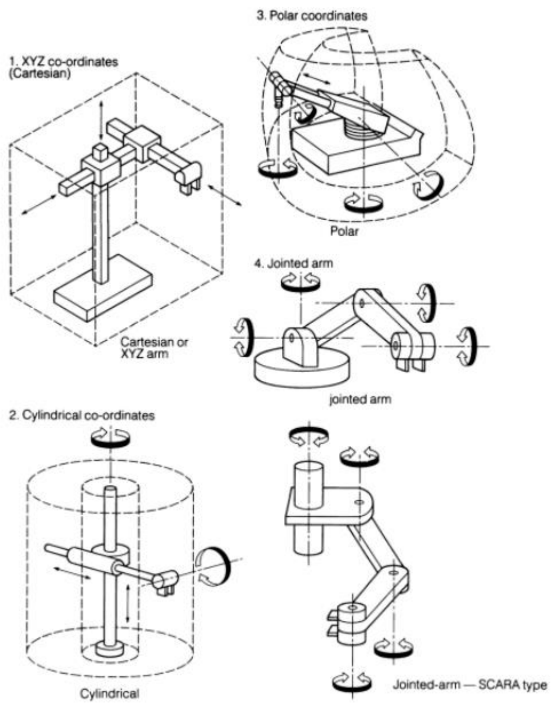


Figure 2. 3 Common robot configuration Source: google

[19] Classically, improved quality and coherence, improved production and performance as well as improved protection and reduced labour costs were the benefits of industrial robots. On the other hand, their expensive initial investment, the shortage of robotic know-how and maintenance expenses are drawbacks.

We'll first dig into the past, to show how robots are moving forward in order to meet demand.

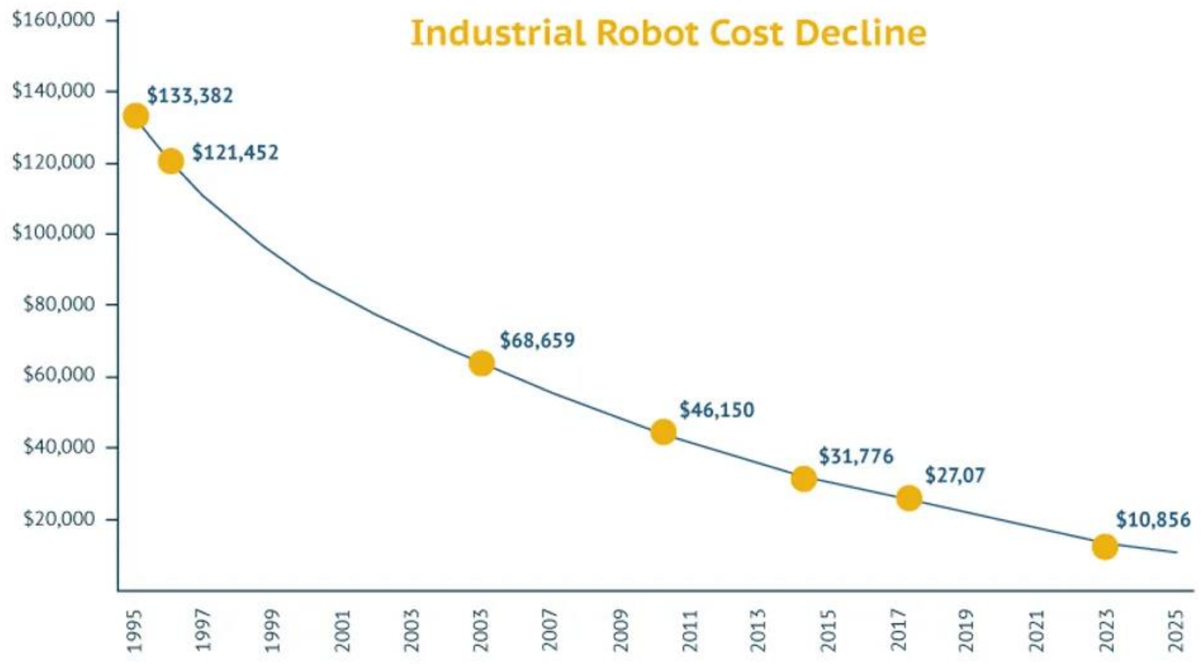


Figure 2. 4 Industrial Robot Cost Decline Source: ARK Invest

2.4 Collaborative Robot

[7] According to MarketsandMarkets research, the worldwide market for collaborative robots, or cobots, is projected to expand at a compound annual growth rate (CAGR) of 41.8% between now and 2026. [20] As Whitton notes, collaborative robots, as opposed to conventional robots, have a considerably lower power consumption, and, thus, are less likely to fail when handing over or colliding.

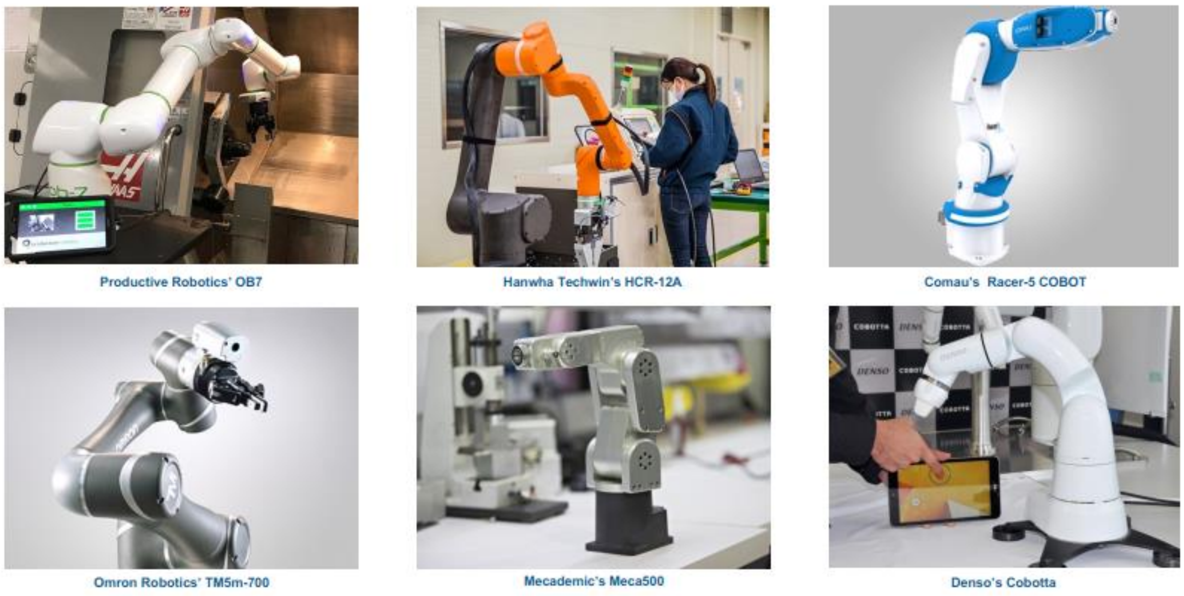


Figure 2. 5 New Collaborative Robot System Source: robotic business review

[21] Improved safety features for working close with the operator and simpler programming for flexible application enables easy deployment and re-deployment in the workplace will be key characteristics that differentiate cobot from conventional industrial-based robot.

There are some features of Cobot:

1. Sharing a Workspace
2. Interactive Collaboration
3. Independent Movement
4. Physical contact Presumed

[22] Key design goals may be stated as creating collaborating robots:

- Safe contact with personnel and sensitive equipment for assembly.
- Reduced expenses to justify the employment of robotic labour.
- Robot activities at a pace consistent with human ability.
- Clean operations and minimal noise.
- Light and compact shape considerations.

- Non-expert programming to deal with high-mixed output is simple and quick.

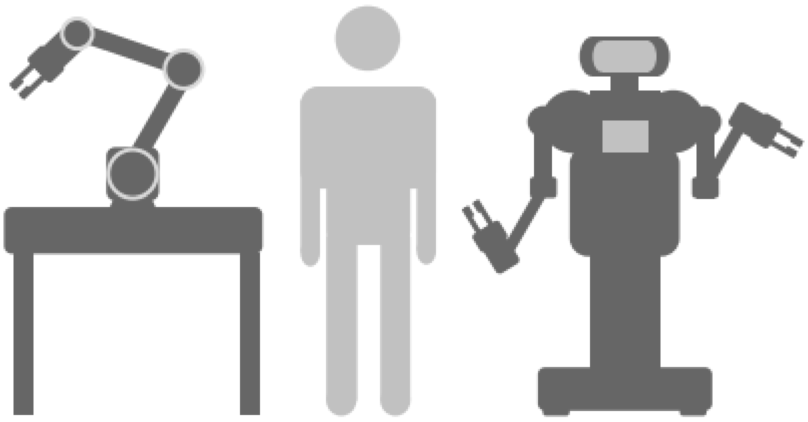


Figure 2. 6 Collaborative Robots Source: [23]

Operational Features	Representative Technologies
Relative low cost/rapid ROI	6 or 7 DoF Articulated arms
Programmed easily/quickly	Force sensing/force control
Fast deployment and integration	No pinch points/sharp edges
Lightweight systems/small footprint	Programming by demonstration
Human scale size and operating range	Lightweight construction/ new materials
Work safely and effectively with humans	Non-Aggressive Appearance/ Cool Colours
Lower power/maintenance requirements	Integrated sensors(torque, vision, sonar etc.)
Versatile, Supports multiple automation tasks	Software Centricity/ intelligence/ “perception”
Operated by workforce with range of education levels	Compliant / gravity compensated / backdrivable arms

Table 2. 0-1 Features of Collaborative Robot

	Safety-Rated Monitored Stop	Speed and Separation Monitoring	Hand Guiding	Power and Force Limiting
Safety Standards	(ISO 10218-1,5.10.2, ISO/TS 15066)	(ISO 10218-1, 5.10.4, ISO/TS 15066)	(ISO 10218-1, 5.10.3, ISO/TS 15066)	(ISO 10218-1, 5.10.5, ISO/TS 15066)
Ensuring Safety	Robot halts when human enters collaborative workspace.	Distance between human and robot monitored in collaborative workspace, slowing or stopping if minimum violated.	Human physically maintains control of robot at all time in collaborative workspace.	Assumes contact, but static and dynamic forces are limited. Materials and construction designed to limit injury.
	~~Shared Workspace~~	Shared Workspace	Shared Workspace	Shared Workspace
	~~Interactive Collaboration~~	~~Interactive Collaboration~~	Interactive Collaboration	Interactive Collaboration
	~~Independent Movement~~	Independent Movement	~~Independent Movement~~	Independent Movement
	~~Physical Contact Presumed~~	~~Physical Contact Presumed~~	Physical Contact Presumed	Physical Contact Presumed

Table 2. 0-2 Classes of Collaborative Operation

[23] Cobots are simple (setup), but exploring the potential offered by cobots is a challenge - finding new methods of sharing work among people and cobots. New and increased skills at all levels are needed, particularly for operators who must be able to modify their cobots on their own.

2.4.1 Collaborative System

[7] We have witnessed enormous strides in collaborative systems and safe robotics over the past several years. Additionally, we have seen significant development in developing interfaces that make programming robots for specific jobs simple and fast. Collaborative systems both for software creation and learning by demonstration have made great improvement over the last several years. The perfect collaborative work environment is where humans do what they are excellent at while robots carry out routine, hazardous, and potentially tedious tasks. [24] categorization of human and robot physical closeness and consider a cooperative robot to operate closer to a person than a collaborative robot. Further, the kind of contact permitted between a person and a robot is categorised. A cooperative robot may be contacted by humans if it is static, while the human may even contact the collective robot while it moves. A collaborative robot is any robot that works with a person without a fence.

2.4.2 Sensors

[25] Mainstream security systems utilise specialised sensors in order to detect presence, trace or distances. In addition to the collaboration of linked sensor, the individual IoT devices adopt transformational or opportunistic sensing to support sophisticated tasks well beyond their initial designs.

Radar Sensors:

In human-robotic cooperation, radar sensors provide a dependable, extremely accurate method to prevent collisions. Radar sensors provide the necessary dependability of human-robot cooperation, even with dirt, smoke, fire or steam in the most unfavourable conditions.

Smart Motion Sensor:

Cobots programmed with artificial intelligence algorithms are used to record human motions in the manufacturing area using smart motion sensors. Network-based smart motion radar sensors, augmented by artificial intelligence and networking technologies, are able to sense human motions in industrial settings and collaborate with cobots with micrometer-level measurement accuracy.

Vision Sensors: Many vision sensing systems are simple and affordable, or sophisticated and costly.[26] Included in this concept are several approaches for handling and interpreting pictures in order to get information that represents human sight. Despite the complexity and computing cost, the fullness of this feeling is one-of-a-kind. Understanding the surroundings and human purpose is critical to provide real-time feedback on behaviour, enabling robots to respond appropriately.

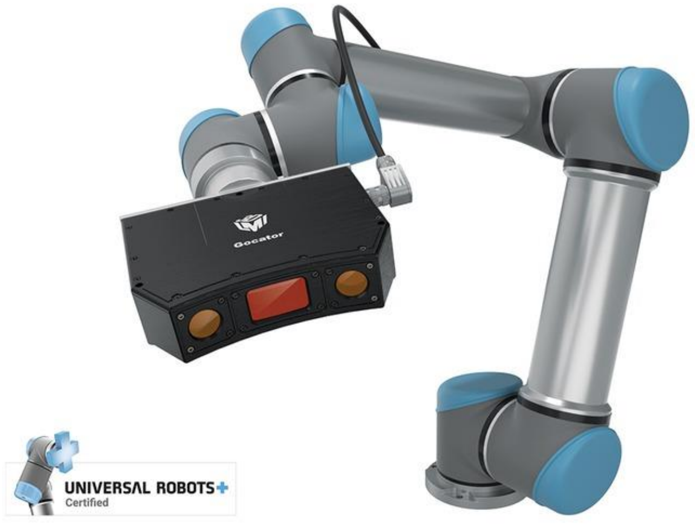


Figure 2. 7 3D vision-guided cobot Source: LMI Technologies

Force Sensors: As a reaction to this force, the force sensor recognises the forces that are being applied. Using force control, you can tell the robot how hard to push components. Instead of following a predetermined route, the robot will deliver a continuous force. Tolerances and placement tolerances both need to be taken into consideration in order for the robot to perform a better buffing or a consistent glue line. [26] In the final analysis, the robot may measure it using joint position error measurement devices or torque sensor technology implanted in the joints; and after ascertaining human intents, it will leverage force control to adapt to the intentions of the human operators.

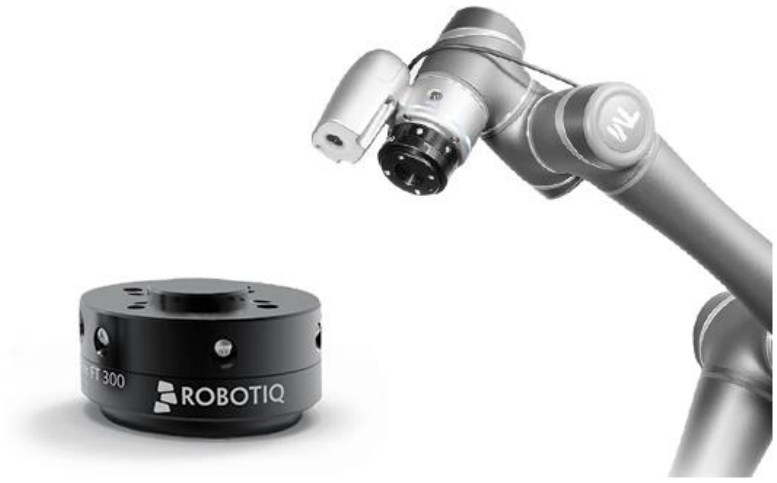


Figure 2. 8 Force and Torque Sensor Source: reeco.co.uk

Safety Sensors: the next generation of human-robot cooperation with improved precision and speed is provided by a functioning safety torque sensor. Light curtains are considerably less invasive than fence, and as a result, they offer a lot of freedom and flexibility when it comes to creating safe zones. The robot may be trained to react appropriately when a human worker walks through the light curtain. An additional feature to safety sensors is the ability to set safe zones that, when someone enters them, activate various behaviours, such a safety perimeter where the robot just slows down, and an inner zone where the robot comes to a full stop.

SAFETY
TORQUE SENSOR
CUSTOM TORQUE SENSOR FOR INTEGRATED
AND IMPROVED FUNCTIONAL SAFETY
IN COLLABORATIVE ROBOTS

Figure 2. 9 Safety Torque Sensor Source: TE Connectivity

Distance Sensor:

[26] The only measurement that humans are unable to make is a true physical measurement. On the other hand, there are many instances within the animal world, for example in the form of echolocation, such as bats and whales. Optical (e.g., infrared or lidar), ultrasonic, or capacitive sensors are used to measure distance. This “sense” is important because there is a clear correlation between the distance from obstructions (i.e., humans) and safety.

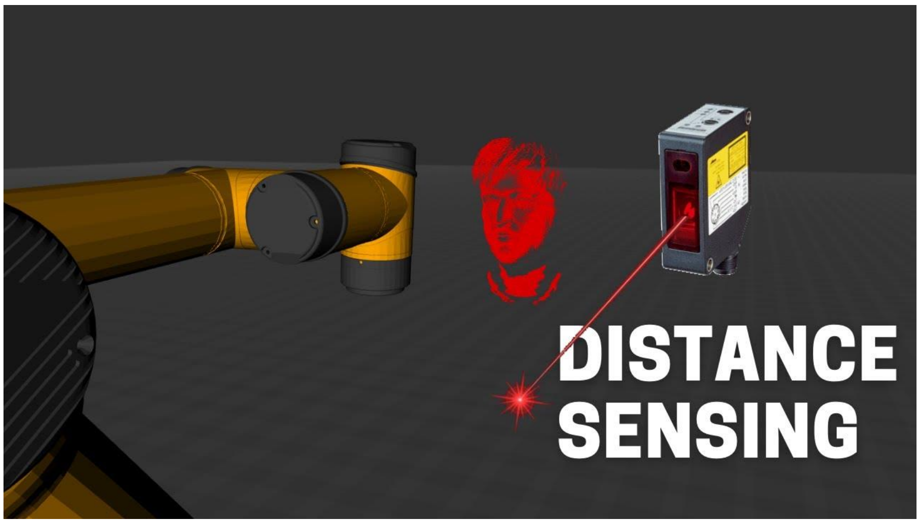


Figure 2. 10 Distance Sensor Source: sick

2.5 Key Difference between Collaborative Robot and Traditional Robots

[27] A cobot may help workers with tasks which may be too hazardous, exhausting or tedious to do by themselves, to provide a safer, more efficient workplace without replacing employment in factories that actually produce a product. Typically, big, solid equipment for high volume, high-precision and high-speed manufacturing are industrial robots. Due to the rapidity of arm movements of the robot, industrial robots may pose safety hazards to human employees, therefore safety precautions such as a cage are often required to prevent people from working in the robot.

Consider some of the main variations between a traditional robot and a cobot:

Features	Traditional Robot	Cobot
Easy to use	To use this need skill of knowledge about this.	COBOTs are designed to work with everyone—not just programmers.
Vision as Standard	Robot vision systems have been available for a long, and their integration is usually more complex.	The vision system expands the variety of applications of Cobot considerably.
Safety	While traditional robots are typically self-sufficient, they are often isolated. Industrial robots do their job blindly until they have been instructed to stop.	Cobots may have pre-set speeds and torques, so they can stop if they detect anything. This isn't just an accident reduction.
Incredible versatile	Robots which are designed either for a specific job or are difficult to adjust to new ones.	That's so simple to set up and operate it can be installed and deployed rapidly and painlessly wherever it is needed.
Applications everywhere	This one is not possible to apply any place.	It is lightweight, transportable and simple to put up anywhere.
Collaboration with Humans	It can’t collaborate with humans to work in the space.	A COBOT has an understanding of it, so that it may learn about its environment and its

		changing environment and work closely with human labourers.
Easy Programmability	It does need a high programming expertise.	There are no sophisticated programming capabilities needed, and staff may reprogram COBOT for various duties.
Fast Setup	It is very difficult to setup also need lot of space.	Often it takes only a few hours to set up COBOTs.
Increased Productivity	Robot can't increase the productivity because it can't complete the different types of task.	The COBOT receives all the usual advantages of a robot in production sector.
Cost-Effective	This one is need numerous additional costs	COBOTs are smaller and can perform more jobs than many industrial robots.

2.6 Application of Cobot

In all sectors there are many collaborative robotic applications.[28] The collaborative robot industry is projected to expand to around $2 billion and 150,000 units by 2021 according to BIS Research. Many sectors look to cobots as a means of introducing the future of automation.

2.6.1 Assembly

Assemblies are getting more computerized. Produce repetitive, precise installation operations such as the drive of the screw, nut driving, fitting and insertion and minimise repeated job accidents. In addition, Cobots may also be utilised throughout the manufacturing process for quality assurance. [29] You may now utilise a robot for your assembly line more easily, less riskily and more quickly.

Collaborative robots for mounting applications provide advantages:

1. Improved quality, consistency and speed of manufacturing.
2. Fast redeployment to new installation configurations may be done easily.
3. Lightweight space saving design for outstanding flexibility in production.
4. Ability to adjust assembly performance to satisfy peak seasons and changing customer requirements.
5. Quick payback without conventional programming, configuration and specialised working cells expenses for robotics.

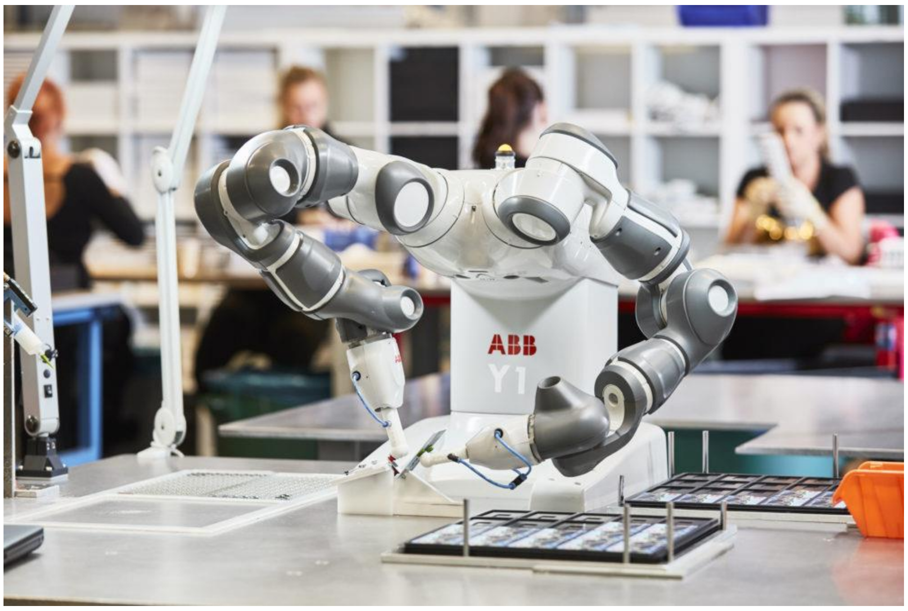


Figure 2. 11 ABB YuMi dual-armed cobot at assembly job Source: cobottrends

2.6.2 Finishing

Manual tools and a significant amount of force are necessary for the completion of human operators' duties. The tool's vibration may damage the operator. A cobot can offer the strength, repeatability and precision essential for completing tasks. These works may be polishing, grinding and deburring. It is possible to teach the robot manually or through computer programming. Force control cobots may assist the robot to increase its robustness.

The robot can thus handle various sized components. This is done either through the end-effector or internally by means of force sensing.

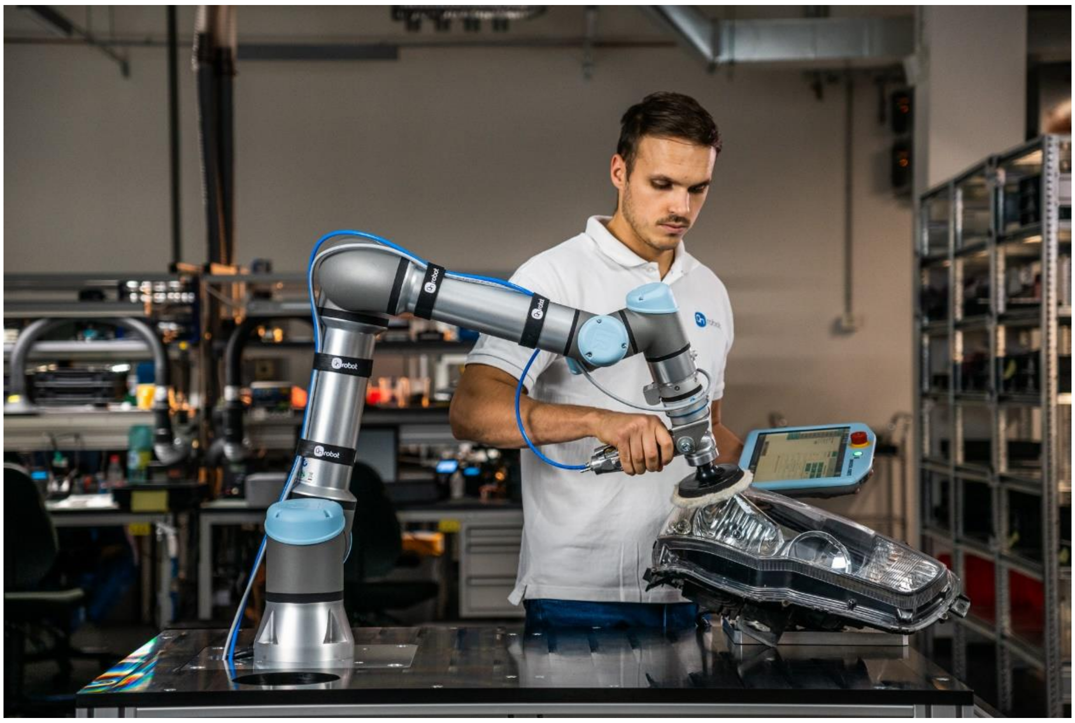


Figure 2. 12 Cobot surface finishing Source: tuliptechs

2.6.3 Machine Tending

[30] Machine care requires a person to stand before a CNC machine, injection modeller or other similar equipment for extended hours and cater to their operating requirements. It may be changing instruments or substituting raw materials. For the human operator, the procedure is lengthy and difficult. Cobots not only release the operator but also a single cobot may use many machines, resulting in improved production. The cobot may need this kind of cobot to have machine-specific input/output interface hardware. The I/O hardware signals the next cycle for the robot or when the material has to be filled in.

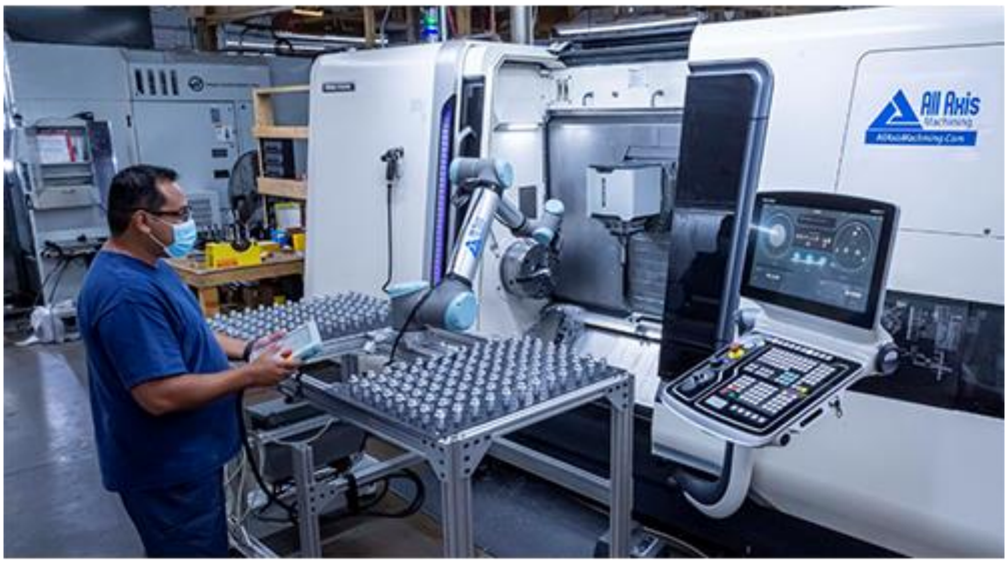


Figure 2. 13 Collaborative Robot automating machine tending task Source: automation

2.6.4 Material Handling

[15] The handling of materials is one of the most hazardous production tasks. The workforce may be exposed to a significant danger from materials such as metal, plastic and other substances. In addition, many jobs of material handling are repeated and may cause repetitive strain damage. Industries that employ robots in production have much less accidents in their job. Heavy items can be readily lifted and moved on moving robot platforms throughout manufacturing floors. Machines tending tools are, meantime, also within the capacity of universal robots, including jobs that require heavy-duty CNC machines.

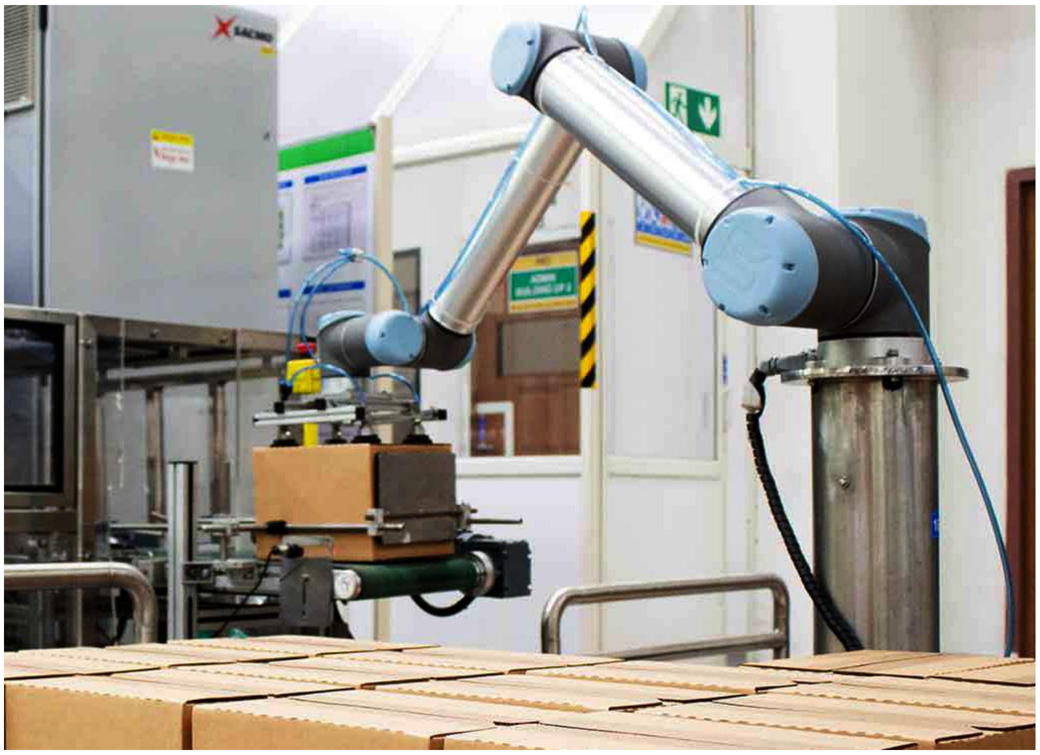


Figure 2. 14 Material handling cobot Source: robot27

2.6.5 Inspection for Quality

[31] Enterprises and customers need goods to be delivered without manufacturing faults, yet inspection by employees is sometimes difficult. In addition, The procedure typically consists of complete examination of whole components, high-resolution photographs and the part check against CAD models for precision machined parts. Multiple high quality cameras may be mounted into cobots to automate the procedure for quicker results. Furthermore, [15] Cobots may lead to better inspections, which lead to more precise batches in manufacture. High resolution cameras end-effectors and vision systems and software may be needed for examination.

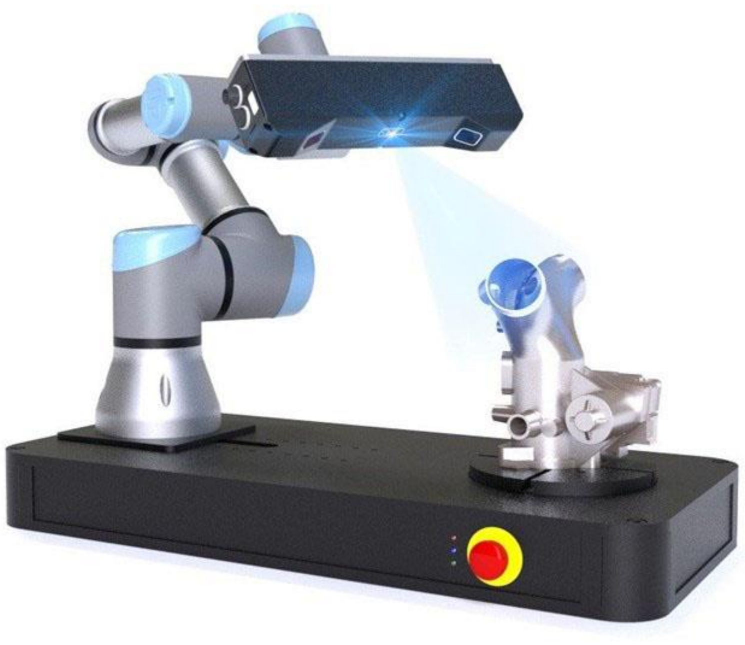


Figure 2. 15 Universal cobot inspectioning for quality Source: Universal Robot

2.6.6 Welding

[15] In the automobile sector in particular the technique of robotic welding has gained momentum throughout the years. The robots' polyvalence of welding has been used in arc soldering, tungsten inert gas welding and spot sounding.

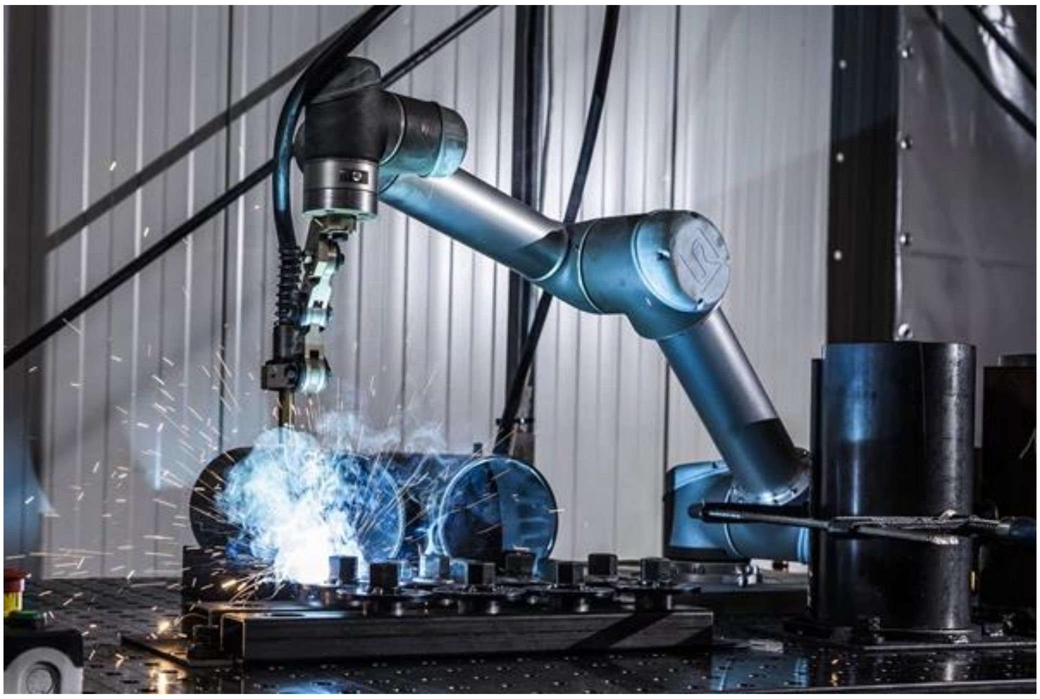


Figure 2. 16 Cobot welding Source: migatronic

2.6.7 Picking and Placing

[31] Manual pick and placement tasks involve a lot of repeat, frequently found worldly by employees. [28] The first time cobot users start with the pick and place task. One job is to choose and put a workpiece and to position it in another area. [31] When coupled with sophisticated vision systems, cobots can more effectively execute jobs, enabling people to concentrate on the aspects of their work that need critical thinking.

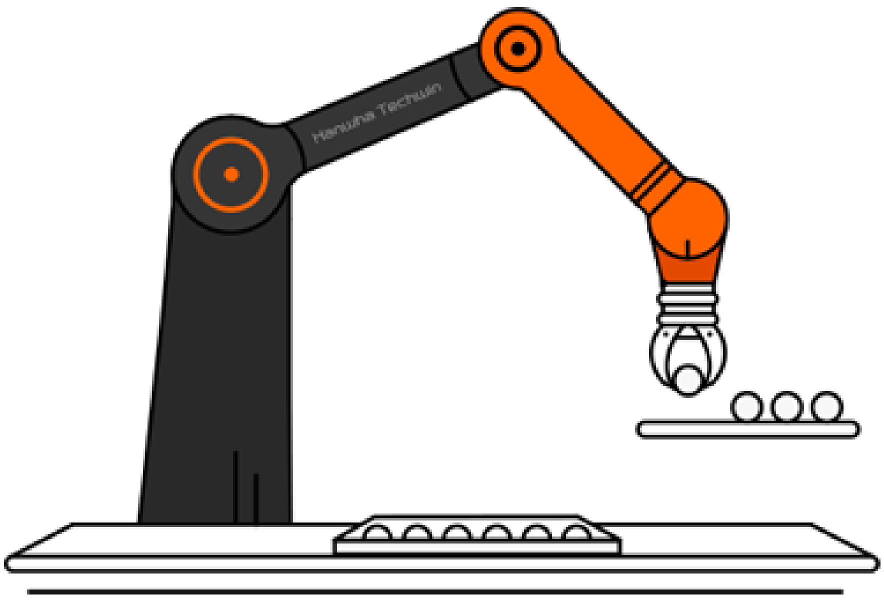


Figure 2. 17 Automatic Pick and Place Cobot Source: cobots.gr

2.7 Human – Robot Collaboration

Industrial competitiveness will be significantly enhanced by collaborative robotics and AI. Human-machine systems have the potential to provide a great deal of new career possibilities and industries that generate new demands for new workers with new skills. [32] As long as the job is shared properly, the combination of people and robots may significantly increase performance. The degree of automation and human involvement that can be implemented via human-robot cooperation varies greatly. If a completely automated solution is neither affordable or too complicated, tasks may be partly automated.

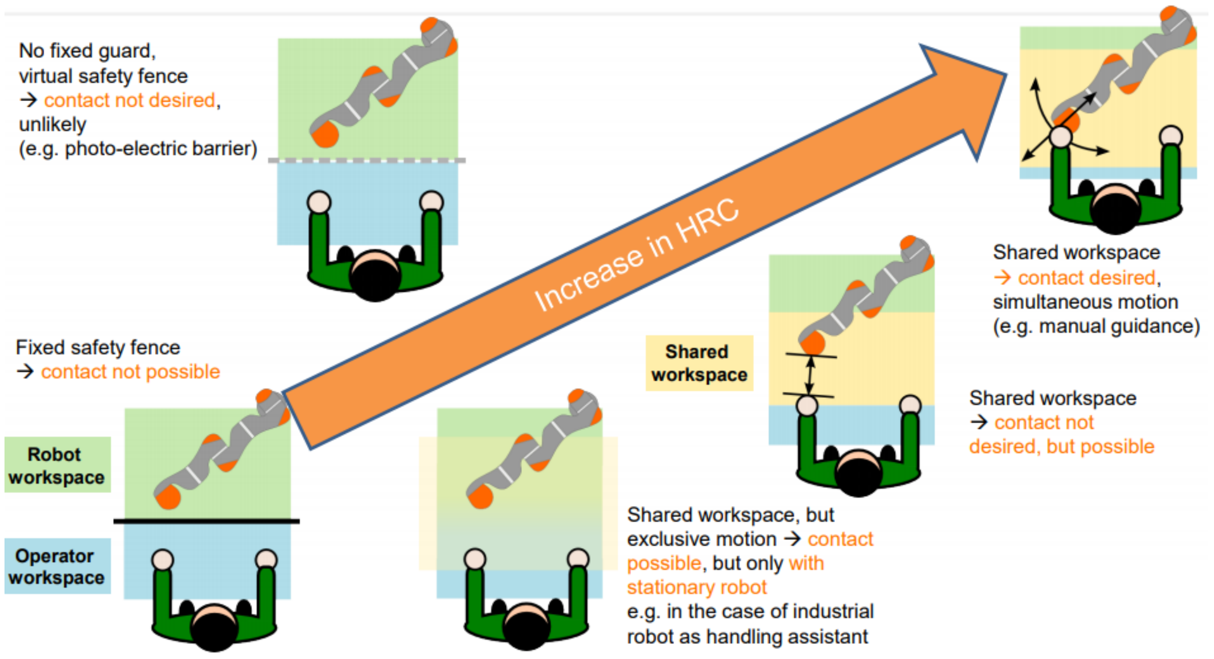


Figure 2. 18 Forms Of Human-robot Collaboration Source: robotics.org

2.8 Safety Mechanism

[26] Collaborative robots include safety as their number one priority. Despite the recent increase in standardised robot safety (such as the ISO 13482:2014, 2014), we are still in the early phases of implementation.

Relevant safety elements to reduce risk of HRC requests in line with the relevant requirements (ISO 10218-1:2011)

- Monitoring of safe speed
- Safe working areas and protected areas
- Safe detection of collisions (free collisions possible)
- Monitoring of safe force (avoidance of pinching or crushing)
- Safe detection of tools
- Secure State Switching (i.e. safe protection zones)

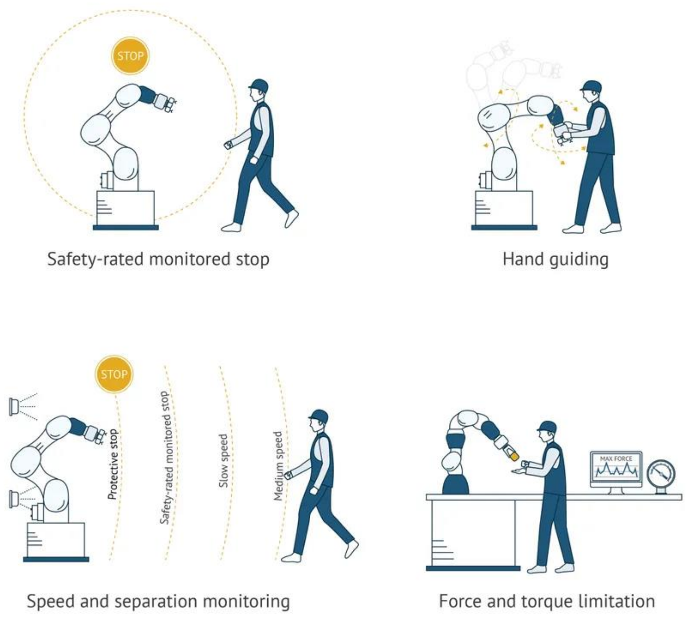

STOP
Safety-rated monitored stop
Hand guiding
STOP
Protective stop
Safety-rated monitored stop
Slow speed
Medium speed
Speed and separation monitoring
MAX FORCE
Force and torque limitation

Figure 2. 19 Design collaborative mechanism for safety robot Source: Wevovlver

2.8.1 Safety Rated Monitored Stop

[33] This may be done via safety features, such as velocity limitation and detection of collisions, which are built into cobots, or by using distance sensors, which enable a cobot to gradually slow down or stop if something or someone is nearby. [24] A cobot usually functions in a well-defined workplace on a specific work item. Cobots are programmed to immediately cease activities when an operator enters the workplace to make it easier for the operator to execute actions on the work piece. The technology guarantees that the robot and human do not move simultaneously and is used mostly when the robot moves heavy components quickly around the workplace.

2.8.2 Hand-Guided Operation

[22] Prior to the commencement of a hand-guided operation, a robot must complete a safety-rated monitored halt. A worker is directly engaged with the robot arm throughout the process,

and may use hand controls to guide it. This technique is often employed for lift assistance or applications in which “tool” features are extremely changeable. [24] This makes it simpler to teach intuitive paths. It facilitates sophisticated and interactive cooperation.[34] Additionally, the control system helps to keep the cobot in accessible settings by compensating for its gravity. To this purpose, a cobot job was described as a number of competencies, and the control programme included the collection of subprograms produced through kinesthetic education.

2.8.3 Speed and Separation Monitoring

[24] The working area of a Cobot is split into zones. The narrower the guy is, the slower the Cobot travels. At a certain level, the Cobot achieves a full stop. This improves the safety of the operator and Cobot. Various surveillance systems mainly use vision to monitor the safety zone. The safety zone may be of any size and shape, the user will build up multiple zones to ensure the robot does not damage the worker under all circumstances, and will link various acceleration and speed adjustations with those zones. This may happen if the human-robot cooperation is not continuous and the robot is operative in full speed most of the time alone.[35] Multilevel speeds or distance limitations are set to create the compensation for productivity so that motion speeds may be changed seamlessly.

2.8.4 Power and Force Limiting

[22] In situations where the collaborative robot (or any workpiece) may be intentionally or unintentionally contact full to a worker, the power and strength limitation are needed. In addition, [24] You don't need to use safety sensors since the operator may be as near to the Cobot as required. [34] In ISO/TS 15,066, the permissible cooperation levels of both power and force are clearly defined. The data regarding allowed biomechanical loads are included in the instance if a person has a collision with an item. To restrict the force or power applied to

the contact event, if the motor brakes are engaged, or to regulate the force or power applied, if the torque control mode with gravity compensation is activated.

2.9 Recent Progress

Intelligent robots with more sensing and dexterity capabilities are becoming more common. Initially, we are seeing an increase in the usage of computer-controlled robots with better robot command languages, which are programmed offline. Equipment, such as sensors that use vision and touch, will be made available to use in this programming.[36] Most robots nowadays have been very well able to combine vocal and visual acknowledgement to help people.

2.10 Current Research Gaps

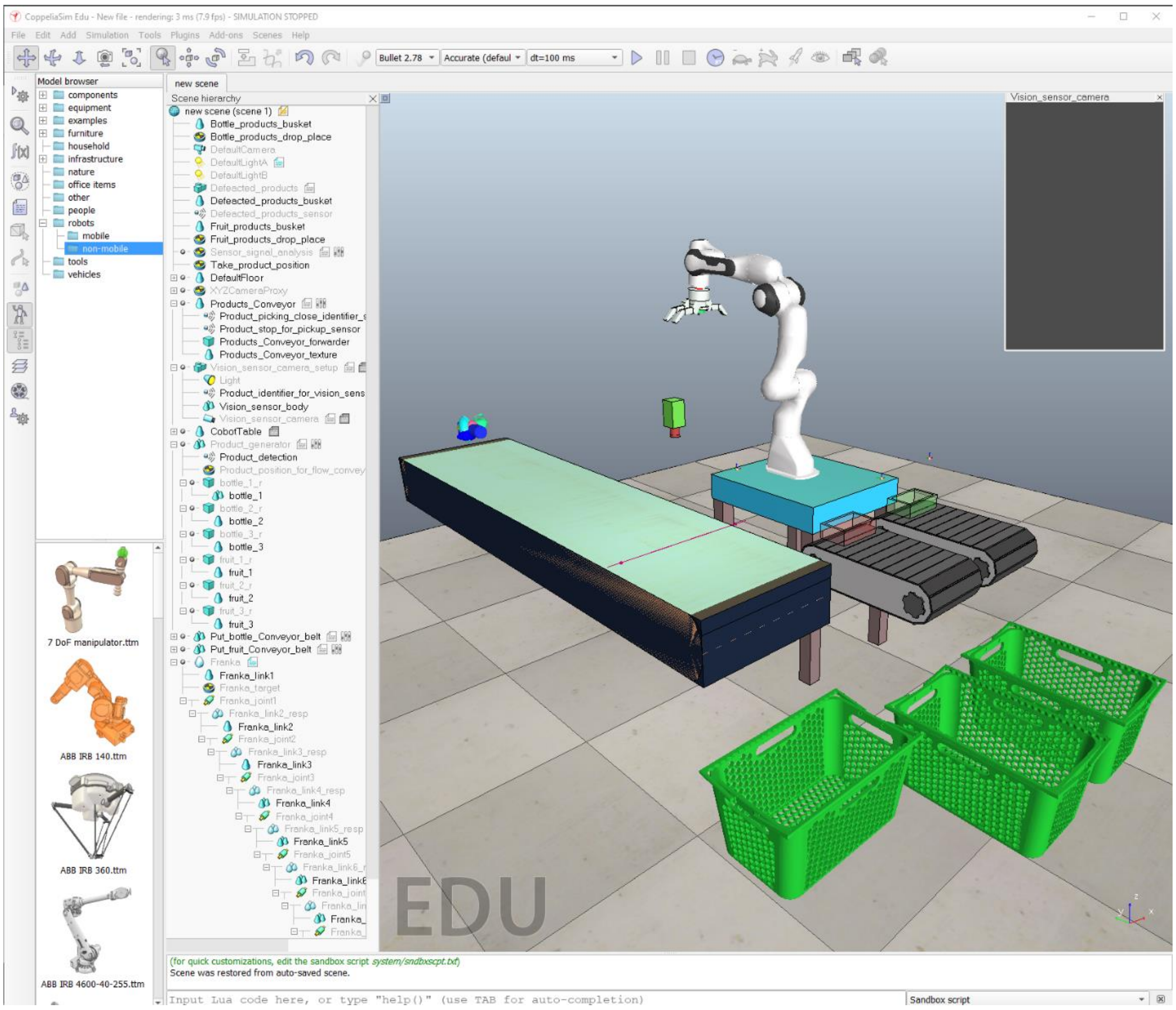
Cobots are not always Autonomous, which still necessitates the need of human operators and certain safety measures are needed. [36] But the capacity of Cobot to comprehend context and react to complicated circumstances is missing.[37] Maybe the next big Cobot will come much more quickly, which will be thanked by the industry.[38] A Cobot can operate quicker than a human being by installing additional safety measures like safety matts or light curtains and halt or decrease speed whenever a person enters the workpiece.

Chapter 3: methodology

Robotics and computer vision have both benefited greatly from advances in artificial intelligence (AI). Robotic assembly, packaging, product distribution, and even customer support may all benefit from the use of machine learning. As a result of their ability to provide real-time course correction in assembly operations, machine learning algorithms are commonly used by industrial robotics companies to improve packaging systems by lowering costs while also improving packaging accuracy, and to boost the efficiency of various processes in general. The usage of convolutional neural networks will be put to the test in order to aid a robotics grocery products distribution system. The robot system's primary function is to classify goods into appropriate types for proper distribution and distribute those types on a conveyor belt.

3.1 CoppeliaSim

[39] CoppeliaSim is a simulation and industrial robotics application programme that runs on many platforms. Moreover, Using this programme, to build simulations for a variety of different situations that are portable, scalable, and low-maintenance.



(for quick customizations, edit the sandbox script *system/sndbxscpt.txt*)
Scene was restored from auto-saved scene.

```
Input Lua code here, or type "help()" (use TAB for auto-completion)
```

Figure 3. 1 CoppeliaSim Software simulator view

The CoppeliaSim Robot Simulator uses a distributed control architectural approach in which the individual control of each object or model necessitates the use of a high-performance embedded script. [39] As a result of these capabilities, CoppeliaSim may be used to simulate a wide variety of robotic systems.

[40] Features include:

1. ROS2 interface.
2. PyRep, a Python toolkit for robot learning research.
3. XML scene/model format import/export.
4. With shadows, OpenGL3 renderer.
5. Cross-platform and open source (Windows, Ubuntu, MacOS).

6. C/C++, Lua, Python, Java, Matlab, and Octave API/ bindings.
7. For every kind of mechanism, a forward/inverse kinematics solver is available.
8. Collision, minimum distance, and proximity sensor module that works with meshes, point clouds, and OC-trees.
9. OMPL plugin for path/motion planning.
10. Embedded distributed scripting provides advanced control and customization.
11. RRS-1, Reflexxes motion library, haptic devices, SDF/URDF, image processing, script commander (read-eval-print loop), and other features are supported.

3.1.1 Architecture and Used Features

There are many features we have been used in our projects those going to described those helps to complete the project of work in the simulator.

Conveyor Belt: We'll create a conveyor belt that works precisely like a genuine one, down to the individual conveyor belt pads that are dynamically recreated. Smaller items, for example, may get caught between two neighbouring pads if this is the case. This kind of simulation is computationally demanding, which causes a delay in the simulation's overall progress.

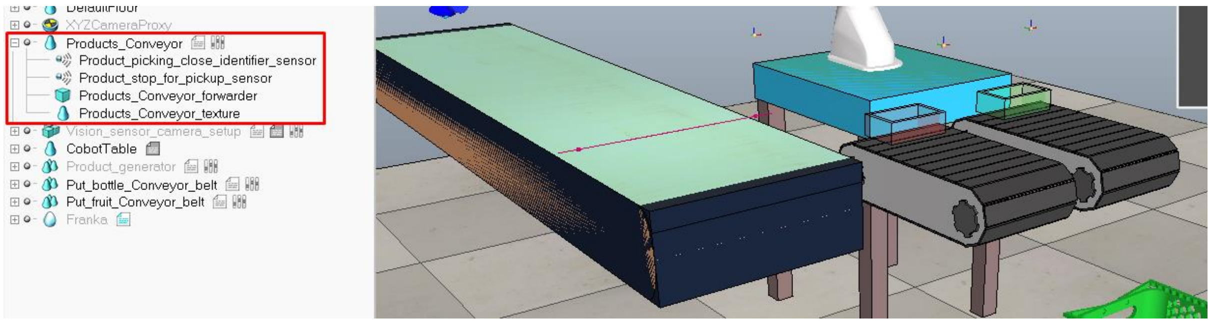

DefaultFloor
XYZCameraProxy
Products_Conveyor
Product_picking_close_identifier_sensor
Product_stop_for_pickup_sensor
Products_Conveyor_forwarder
Products_Conveyor_texture
Vision_sensor_camera_setup
CobotTable
Product_generator
Put_bottle_Conveyor_belt
Put_fruit_Conveyor_belt
Franka

Figure 3. 2 CoppeliaSim Conveyor belt

FRANKA Cobot: Incorporating traditional stiff industrial robot characteristics, Franka Emika Panda has posture repeatability of +/-0.1mm and minimal route deviation even at speeds of up to 2 m/s, making it an ideal cobot. With seven degrees of freedom (DOF) and torque sensors at each joint, the delicate and fast Arm offers customizable stiffness/compliance and sophisticated torque control capabilities.

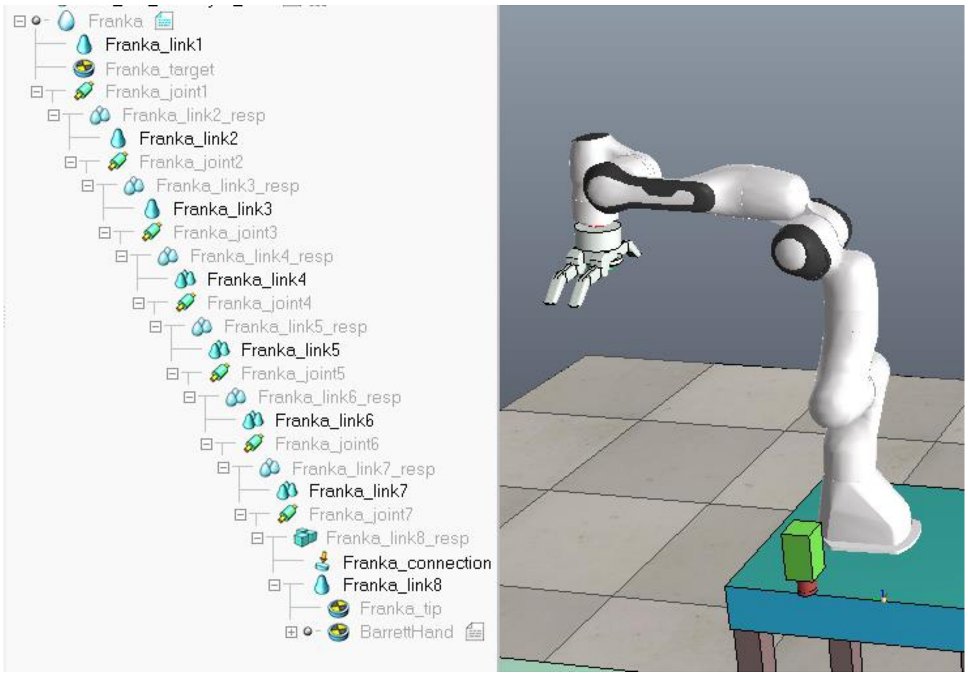


Figure 3. 3 FRANKA Cobot

Gripper: Multi-fingered programmable grasper with the dexterity to secure target items of various sizes, shapes, and orientations, the BH8 Barrett Hand is part of the Barrett Hand series. Despite its small size and light weight (1.18kg), it is completely self-contained.

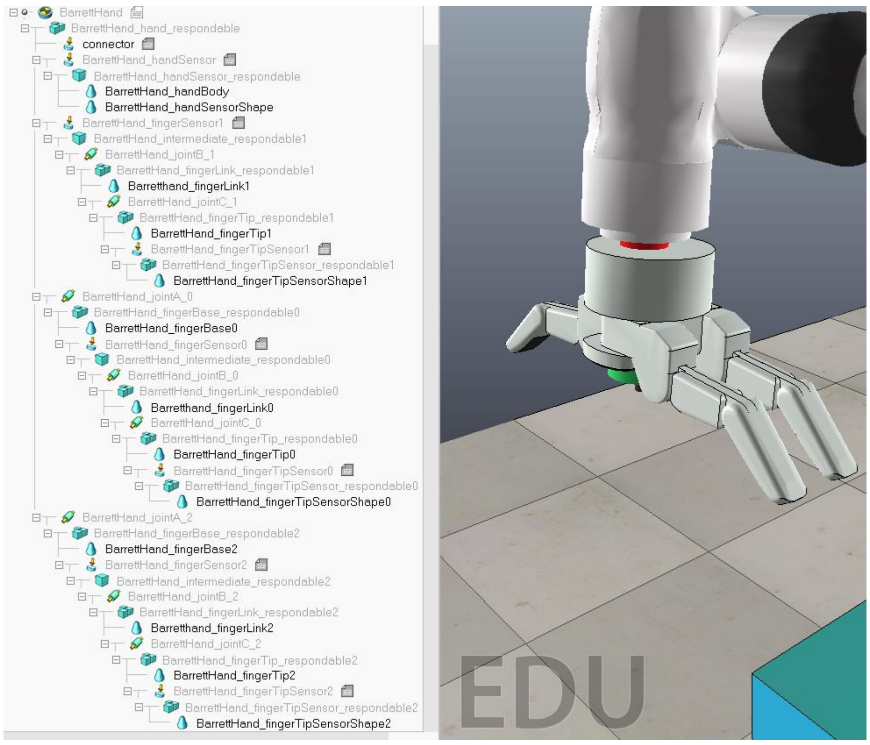

BarrettHand
BarrettHand_hand_respondable
connector
BarrettHand_handSensor
BarrettHand_handSensor_respondable
BarrettHand_handBody
BarrettHand_handSensorShape
BarrettHand_fingerSensor1
BarrettHand_intermediate_respondable1
BarrettHand_jointB_1
BarrettHand_fingerLink_respondable1
Barretthand_fingerLink1
BarrettHand_jointC_1
BarrettHand_fingerTip_respondable1
BarrettHand_fingerTip1
BarrettHand_fingerTipSensor1
BarrettHand_fingerTipSensor_respondable1
BarrettHand_fingerTipSensorShape1
BarrettHand_jointA_0
BarrettHand_fingerBase_respondable0
BarrettHand_fingerBase0
BarrettHand_fingerSensor0
BarrettHand_intermediate_respondable0
BarrettHand_jointB_0
BarrettHand_fingerLink_respondable0
Barretthand_fingerLink0
BarrettHand_jointC_0
BarrettHand_fingerTip_respondable0
BarrettHand_fingerTip0
BarrettHand_fingerTipSensor0
BarrettHand_fingerTipSensor_respondable0
BarrettHand_fingerTipSensorShape0
BarrettHand_jointA_2
BarrettHand_fingerBase_respondable2
BarrettHand_fingerBase2
BarrettHand_fingerSensor2
BarrettHand_intermediate_respondable2
BarrettHand_jointB_2
BarrettHand_fingerLink_respondable2
Barretthand_fingerLink2
BarrettHand_jointC_2
BarrettHand_fingerTip_respondable2
BarrettHand_fingerTip2
BarrettHand_fingerTipSensor2
BarrettHand_fingerTipSensor_respondable2
BarrettHand_fingerTipSensorShape2
EDU

Figure 3. 4 Barrett Hand Gripper

Proximity Sensor Simulation: CoppeliaSim simulates proximity sensors in a method that is both powerful and efficient. There are several different types of proximity sensors that may be modelled by the user. It is possible to identify observable things with the use of proximity sensors. The simulations shown in the following images were created using proximity sensors.

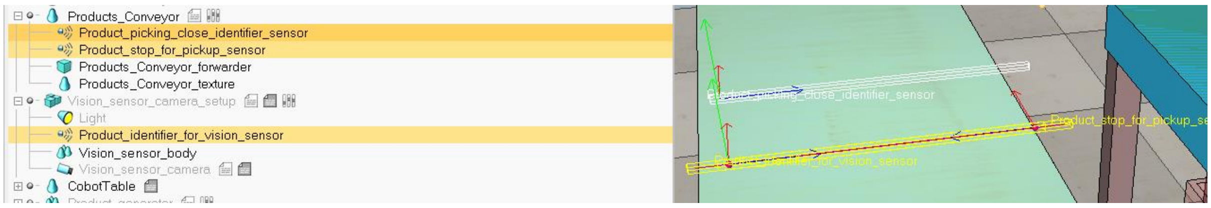

Products_Conveyor
Product_picking_close_identifier_sensor
Product_stop_for_pickup_sensor
Products_Conveyor_forwarder
Products_Conveyor_texture
Vision_sensor_camera_setup
Light
Product_identifier_for_vision_sensor
Vision_sensor_body
Vision_sensor_camera
CobotTable
Product_picking_close_identifier_sensor
Product_stop_for_pickup_se
Product_identifier_for_vision_sensor

Figure 3. 5 Proximity sensor

Vision Sensor Simulation: It is quite similar to how a camera works; vision sensors render the things in their range of vision and activate detection if certain thresholds are exceeded or missed. When colour, light, or structure are factors in the detection process, vision sensors, which can identify renderable things, should be utilised instead of proximity sensors. Vision sensors, on the other hand, may be somewhat slower than proximity sensors depending on the graphic card the programme is running on or the intricacy of the scene objects. The following is an example of how vision sensors are put to use:

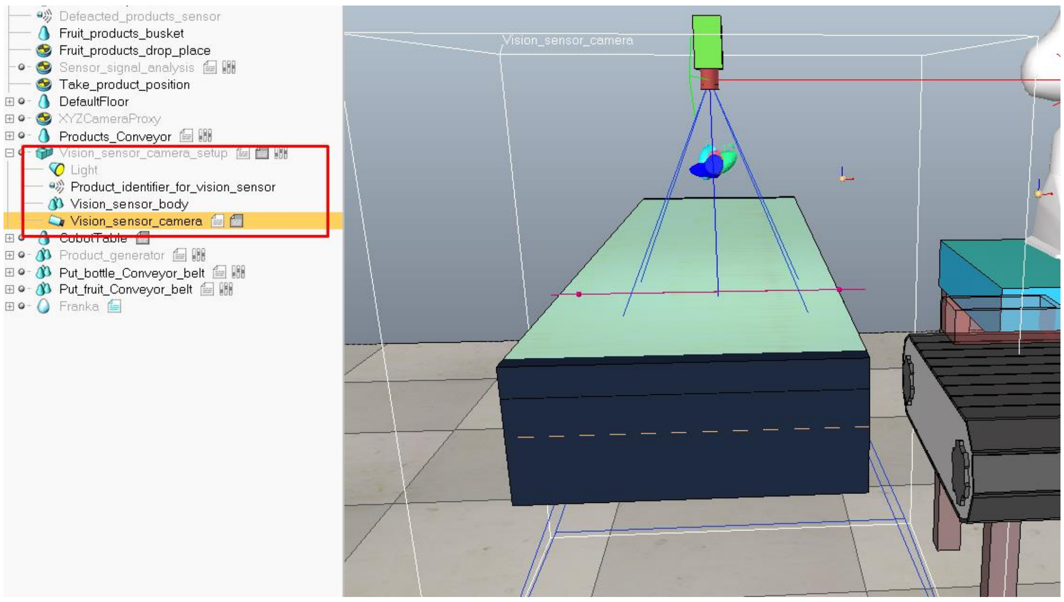
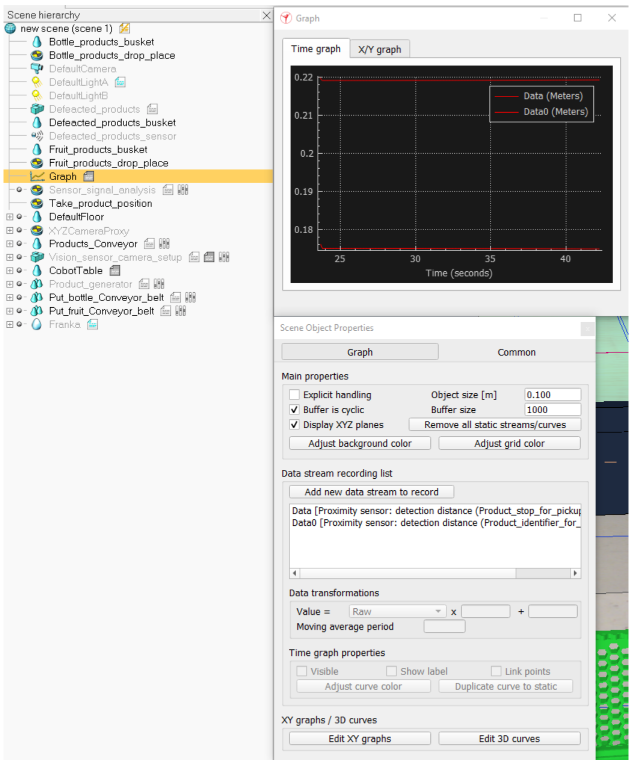


Figure 3. 6 Vision Sensor

Graph: In a simulation, graphs are objects that may be used to store, display, or export data They're very capable and adaptable. The user has a wide range of data kinds to choose from, each of which may be applied to different sorts of objects. To display data, use time graphs, X/Y graphs, and 3D curves. Data is stored in the form of a stream.



Main properties

- [ ] Explicit handling — Object size [m]: 0.100
- [✓] Buffer is cyclic — Buffer size: 1000
- [✓] Display XYZ planes — Remove all static streams/curves

Data stream recording list

Data [Proximity sensor: detection distance (Product_stop_for_pickup
Data0 [Proximity sensor: detection distance (Product_identifier_for_

Data transformations

Value = Raw x +
Moving average period

Time graph properties

- [ ] Visible
- [ ] Show label
- [ ] Link points

Figure 3. 7 Graph data

Dummy: A dummy object is the simplest object available: just a point with some orientation to it; it may be used as a frame of reference. However, when utilised with other objects or computation modules, dummies may be very important; multifunctional or assistance objects can be dummies.

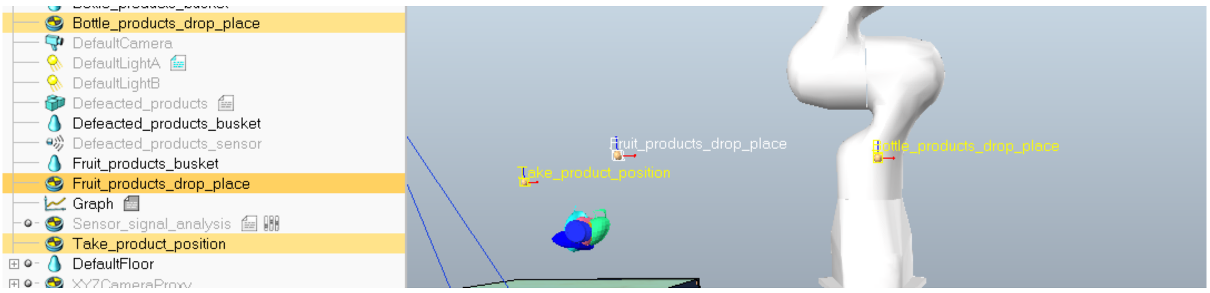


Figure 3. 8 Dummy object

Primitive Shape: Shapes are triangular-faced, stiff mesh objects. The files may be used in many ways, such as importing, exporting, and editing. It is capable of representing any kind of mesh. It comes in a variety of shades and has a diverse range of aesthetic characteristics.

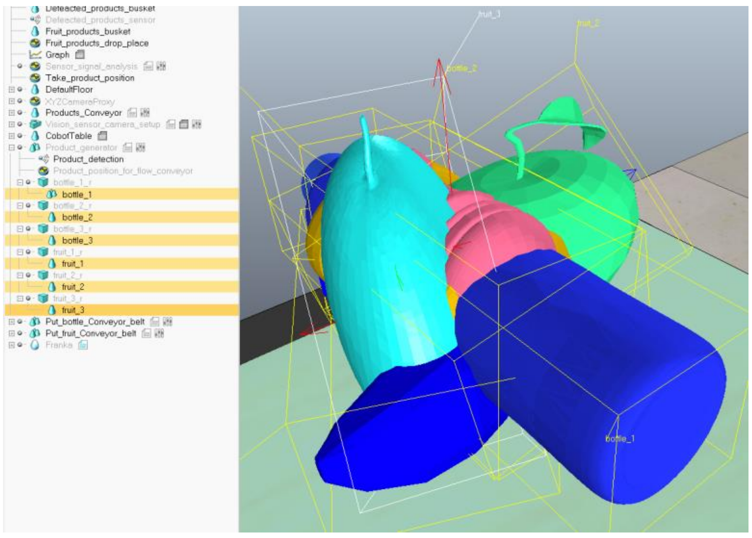


Figure 3. 9 Primitive Shape

Joint: To have a joint is to have at least one inherent degree of freedom in an item (DoF). Mechanisms and things that move rely on joints.

Camera: Viewable items, such as cameras, allow to see through them to see what they are seeing. The number of cameras can use in scenario is up to depend; each one may provide a unique perspective on the situation.

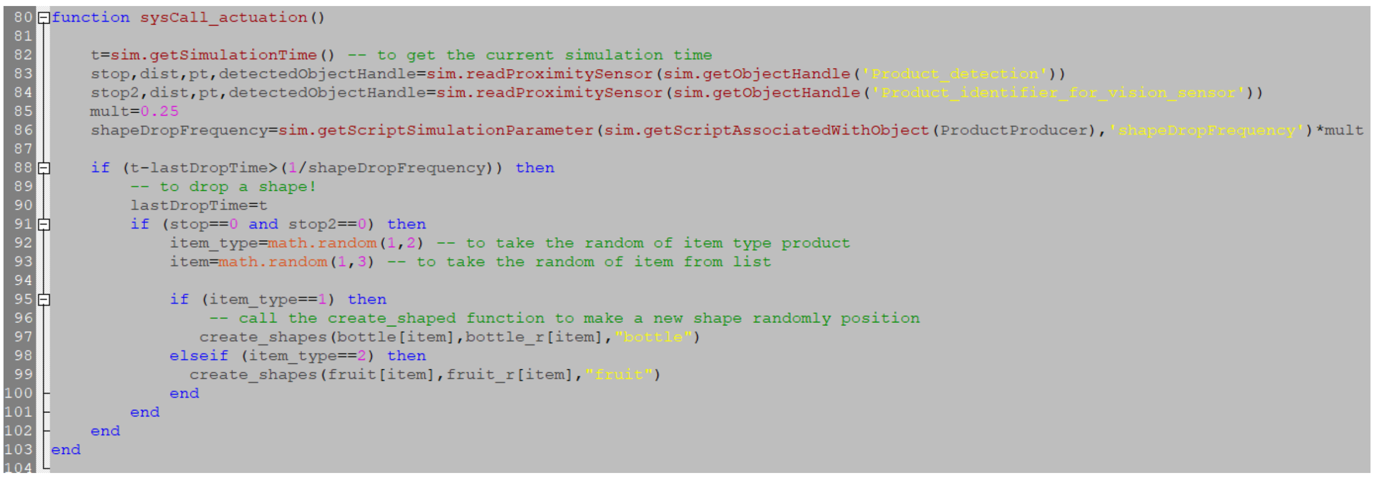
Remote API: Cross-platform and supporting service calls (i.e. blocking calls) as well as bidirectional data streaming, it lets CoppeliaSim connect to an external application (i.e. one operating in a separate process or on a different computer).

Inverse kinematics: The inverse kinematics (IK) calculation module in CoppeliaSim is very strong and adaptable. There are a wide range of mechanisms that can be handled using IK or FK modes, depending on the user's preference (FK mode). The IK issue may be seen as a quest for the joint values that correlate to a particular body element location and/or orientation (generally the end effector). To put it another way, it is a change in coordinate system from the task space into the joint space. To determine the value of all the joints in a serial manipulator, for example, you'd have to know the end effector's location (and/or orientation). An simpler issue than IK is known as an FK problem, which involves determining the location of the end effector given the values of the joints.

3.1.2 Implementation

To implement this project, first we setup a conveyor from the model browser options, then we made a proximity sensor and fixed both positions to stop the product at a specific place. Moreover, import the product mesh.obj file into the simulator, then fix the position on the main conveyor, then add a proximity sensor in the middle of the main conveyor to detect the product. Next, take a FRANKA Cobot from the model browser and take a table to fix the position of the Cobot beside the conveyor. In addition. Add the gripper with FRANKA Cobot as a child. Similarly, take two small conveyors and fix their positions with a Cobot. Besides, import the basket mesh.obj file and fix the position of the small conveyor. In front of the main conveyor, put another basket to get the defected products. To make three dummies to send the command to the Cobot for picking and placing the targeted position. Finally, adding a vision sensor in front of the main conveyor where there is also a proximity sensor to detect the product and send the command to the vision sensor to take the image.

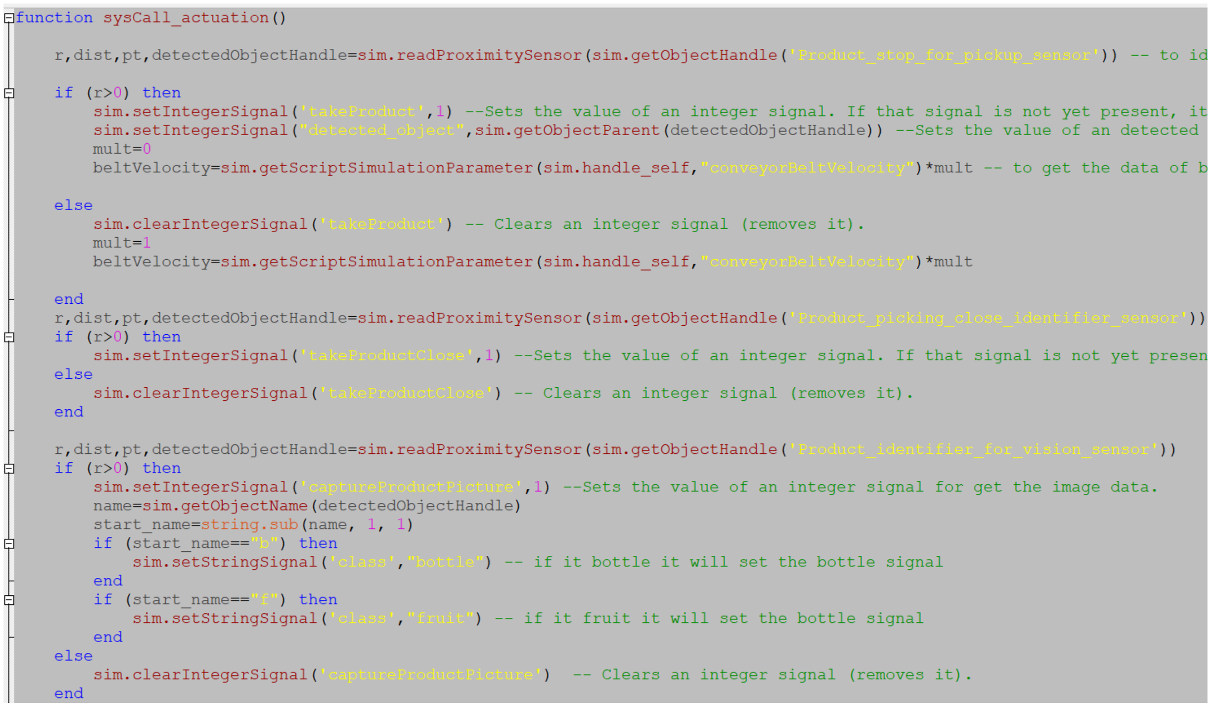
When running the simulator, it will generate a random shape and position of the product which will come on the conveyor. Where have you used a condition with math? The random function is used to generate the product at random and to create the shape as a declaration basis.



```lua
function sysCall_actuation()

    t=sim.getSimulationTime() -- to get the current simulation time
    stop,dist,pt,detectedObjectHandle=sim.readProximitySensor(sim.getObjectHandle('Product_detection'))
    stop2,dist,pt,detectedObjectHandle=sim.readProximitySensor(sim.getObjectHandle('Product_identifier_for_vision_sensor'))
    mult=0.25
    shapeDropFrequency=sim.getScriptSimulationParameter(sim.getScriptAssociatedWithObject(ProductProducer),'shapeDropFrequency')*mult

    if (t-lastDropTime>(1/shapeDropFrequency)) then
        -- to drop a shape!
        lastDropTime=t
        if (stop==0 and stop2==0) then
            item_type=math.random(1,2) -- to take the random of item type product
            item=math.random(1,3) -- to take the random of item from list

            if (item_type==1) then
                -- call the create_shaped function to make a new shape randomly position
               create_shapes(bottle[item],bottle_r[item],"bottle")
            elseif (item_type==2) then
              create_shapes(fruit[item],fruit_r[item],"fruit")
            end
        end
    end
end
```

Figure 3. 10 Product generator Code

After coming to a specific position, the conveyor proximity sensor will detect the product and stop the product from going over. Another proximity sensor has been used as a vision sensor.

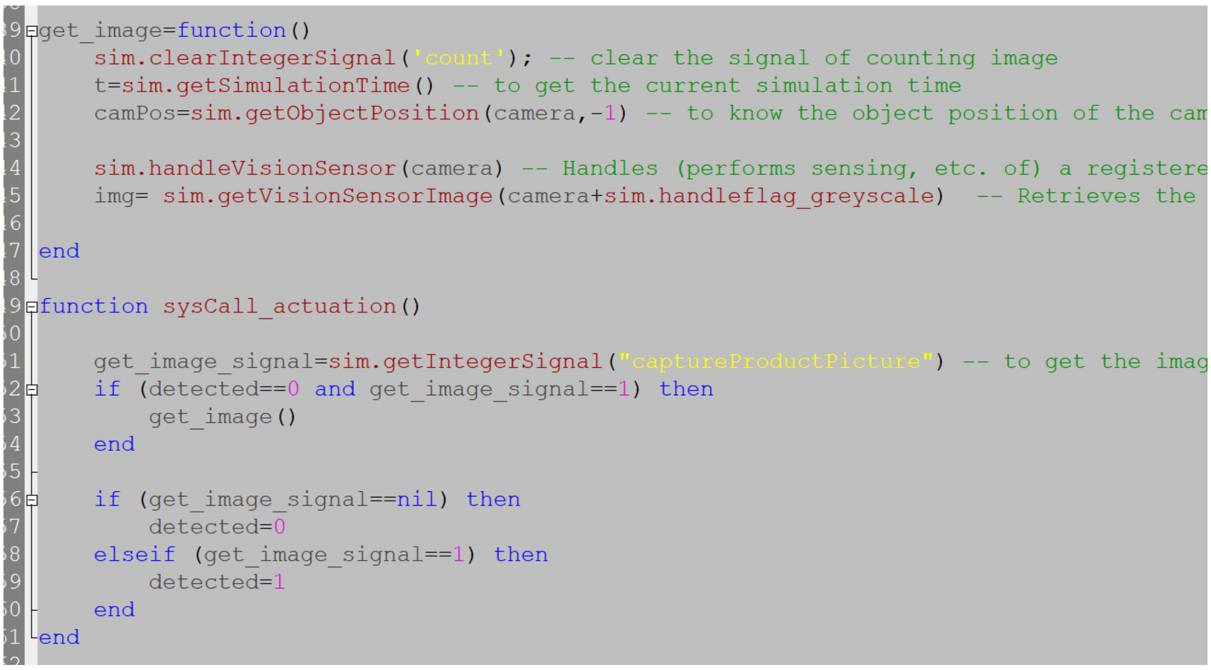


```lua
function sysCall_actuation()

    r,dist,pt,detectedObjectHandle=sim.readProximitySensor(sim.getObjectHandle('Product_stop_for_pickup_sensor')) -- to id

    if (r>0) then
        sim.setIntegerSignal('takeProduct',1) --Sets the value of an integer signal. If that signal is not yet present, it
        sim.setIntegerSignal("detected_object",sim.getObjectParent(detectedObjectHandle)) --Sets the value of an detected
        mult=0
        beltVelocity=sim.getScriptSimulationParameter(sim.handle_self,"conveyorBeltVelocity")*mult -- to get the data of b

    else
        sim.clearIntegerSignal('takeProduct') -- Clears an integer signal (removes it).
        mult=1
        beltVelocity=sim.getScriptSimulationParameter(sim.handle_self,"conveyorBeltVelocity")*mult

    end
    r,dist,pt,detectedObjectHandle=sim.readProximitySensor(sim.getObjectHandle('Product_picking_close_identifier_sensor'))
    if (r>0) then
        sim.setIntegerSignal('takeProductClose',1) --Sets the value of an integer signal. If that signal is not yet presen
    else
        sim.clearIntegerSignal('takeProductClose') -- Clears an integer signal (removes it).
    end

    r,dist,pt,detectedObjectHandle=sim.readProximitySensor(sim.getObjectHandle('Product_identifier_for_vision_sensor'))
    if (r>0) then
        sim.setIntegerSignal('captureProductPicture',1) --Sets the value of an integer signal for get the image data.
        name=sim.getObjectName(detectedObjectHandle)
        start_name=string.sub(name, 1, 1)
        if (start_name=="b") then
            sim.setStringSignal('class',"bottle") -- if it bottle it will set the bottle signal
        end
        if (start_name=="f") then
            sim.setStringSignal('class',"fruit") -- if it fruit it will set the bottle signal
        end
    else
        sim.clearIntegerSignal('captureProductPicture')  -- Clears an integer signal (removes it).
    end
```

Figure 3. 11 Product Conveyor Code

When it detects the product, it uses the setIntegerSignal function data for the Python API to send the command to the vision sensor via the setIntegerSignal function to take the picture.

When the vision sensor gets the signal by the getIntergerSignal function, then it will call the function of get_image() and take the picture by the getVisionSensorImage function .



```
get_image=function()
    sim.clearIntegerSignal('count'); -- clear the signal of counting image
    t=sim.getSimulationTime() -- to get the current simulation time
    camPos=sim.getObjectPosition(camera,-1) -- to know the object position of the cam

    sim.handleVisionSensor(camera) -- Handles (performs sensing, etc. of) a registere
    img= sim.getVisionSensorImage(camera+sim.handleflag_greyscale)  -- Retrieves the

end

function sysCall_actuation()

    get_image_signal=sim.getIntegerSignal("captureProductPicture") -- to get the imag
    if (detected==0 and get_image_signal==1) then
        get_image()
    end

    if (get_image_signal==nil) then
        detected=0
    elseif (get_image_signal==1) then
        detected=1
    end
end
```

Figure 3. 12 Vision Sensor Code

Similarly, when FRANKA Cobot gets the signal by getIntegerSignal, it will activate and follow the command of dummy object target position, then pick the product and place it at the class type and number matching object position.

3.2 Machine Learning and Deep Learning:

Image recognition algorithms are often used in robotics to identify the presence of items and people, forecast the location of an object, check product quality and estimate storage needs. These and other common applications of machine learning in robotics are discussed below. Machine learning is widely utilised for decision-making in the robotics sector. Using AI and information from the environment, a robotic system may, for example, package an item, reject a manufactured component, stop the process owing to safety conditions and adjust their operation in the presence of other robots and human collaborators. We will utilise convolutional neural networks to enhance robot component categorization and packing accuracy in this assignment.

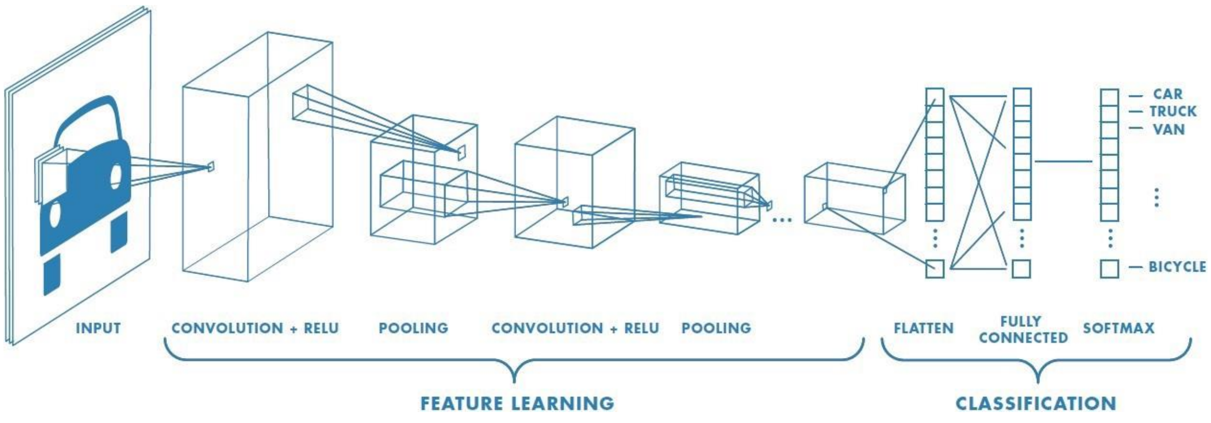


Figure 3. 13 Convolutional Neural Network with classification Source: towardsdatascience

[41] As part of the decision-making process, machine learning (ML) technologies look at the potential of allowing digital computing units to learn and improve over time. In the past, Arthur Samuel invented the term ML in 1959. Moreover, Deep Learning (DL) is a subfield of Machine Learning (ML) that makes use of the learning model's many hidden layers. In addition, [42] Deep Learning is capable of supporting both supervised and unsupervised learning techniques. As previously said, deep learning is mostly used to problems requiring big models that need a significant amount of computing power to solve.

3.2.1 Architecture and used features

Convolutional Neural Network: [43] a Convolutional Neural Network (ConvNet/CNN), an image as input into a machine learning algorithm that uses weights and biases to determine the significance of different features and objects in the picture. When compared to other classification methods, the amount of pre-processing needed by a ConvNet is much smaller. While filters are hand-engineered in basic techniques, ConvNets can learn these filters/characteristics with enough training. Constraint-based learning in convolutional neural networks (CNNs) allows for the automated learning of many filters unique to a training dataset within the restrictions of a particular predictive modelling task like image categorization. Thus, very particular characteristics appear on the input pictures that can be seen almost everywhere.

TensorFlow: Data flow graphs are used to construct models in this open source artificial intelligence framework. It gives programmers the ability to build complex neural networks with a high number of layers. Classes, perception, understanding, discovery and creation are among of the most common uses of TensorFlow. [44] This framework gives programmers the ability to build dataflow graphs, which define how data flows via a network or set of processing nodes. TensorFlow is an open source framework. As a result, each network node represents a mathematical process, and each link or edge between nodes represents a multidimensional data set (or tensor).

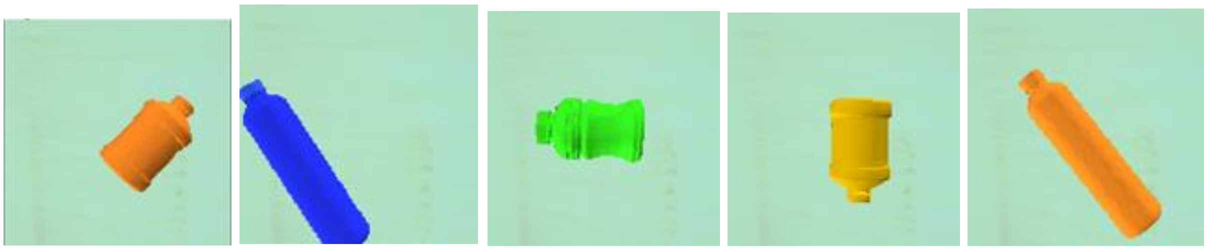
keras : An accessible and productive interface to addressing machine learning issues, focusing on contemporary deep learning, makes up the Keras high-level API for TensorFlow 2. In order to create and launch machine learning solutions with high iteration velocity, it offers necessary abstractions and building pieces. Keras uses a basic Python framework to build deep learning models since it's easy to understand and faster to write. It's scalable without sacrificing usability. They may be mixed in many ways since the learning models created by Keras are separate components. Recurrent and convolution networks are both supported by Keras. A lot of people use and support neural networks since they're built in Python and have a big user base.

Numpy: NumPy is a critical part of Python's scientific computing infrastructure. In this Python library, a multidimensional array object is provided, as well as various derived objects (such as masked arrays and matrices) and a variety of routines for performing fast operations on arrays (such as discrete Fourier transforms), including mathematical, logical and shape manipulation, sorting and selecting, and I/O.

3.2.2 Implementation

At first, need to take the images from the vision sensor for prediction of classification data. The network you've been given has been trained on a product distribution picture dataset. According to the picture below, the dataset includes photos of several items from two different product classes, all in different colours and orientations. Table 3.1 shows how the dataset is split into three sections: training, validation, and testing. The dataset for product classification is located in the data folder, and the pictures for the various training, validation, and testing sets are arranged in the folder.

Bottle:



Fruit:

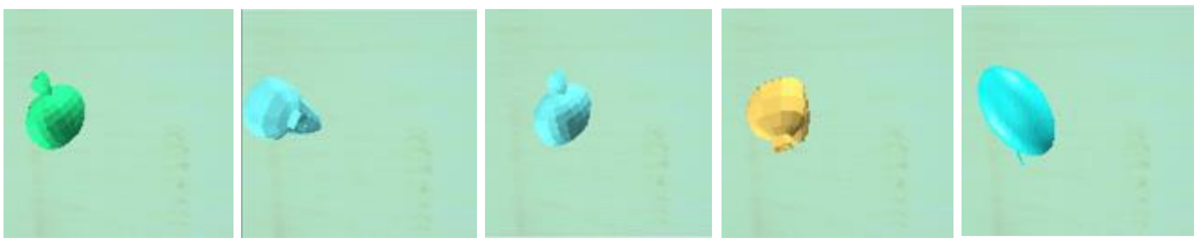


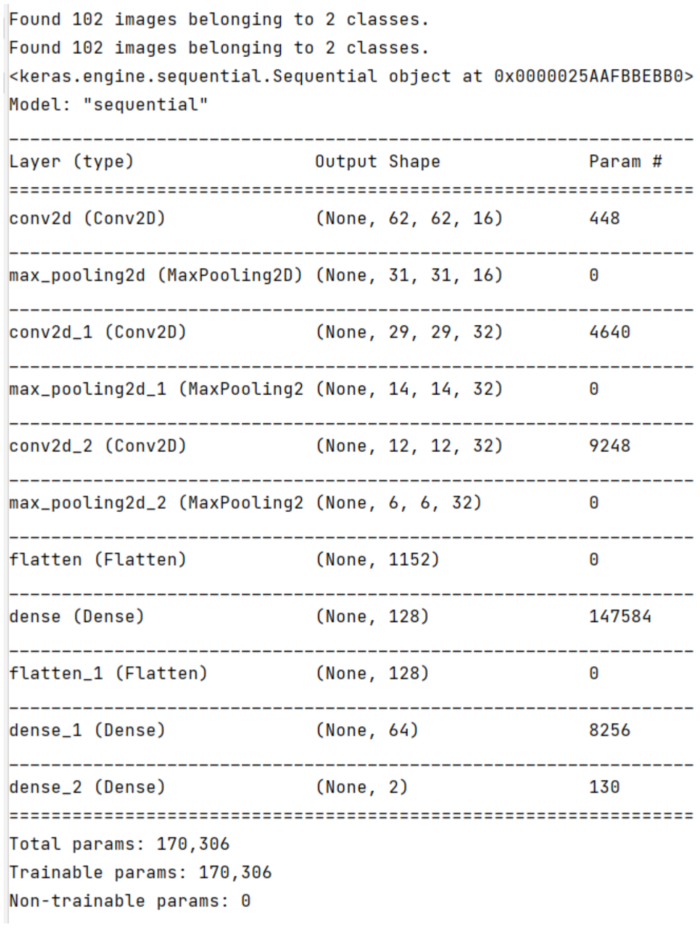
Figure 3. 14 Grocery Products distribution dataset

	Bottle	**Fruit**
Testing	50	50
Training	50	50
Validation	50	50

Table 3. 1 Dataset size of the 2 classes

Check out testing_training_validation_image_classification.py to see that all Use the "data" folder to train a convolutional neural network using this python script. The "image_classification.h5" file in the project-file folder is a pretrained neural network. There is an issue where, when we execute the script testing_training_validation_image_classification.py from the console, it uses the model stored as "image_classification.h5" to evaluate the pretrained model on the validation and testing sets.

As illustrated in Figure 3.15, the code prints the network's fundamental architecture as well as the model's correctness throughout training, validation, and testing. Error! Unfortunately, the cited source could not be located.



```
Found 102 images belonging to 2 classes.
Found 102 images belonging to 2 classes.
<keras.engine.sequential.Sequential object at 0x0000025AAFBBEBB0>
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 62, 62, 16)        448       
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 31, 31, 16)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 29, 29, 32)        4640      
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 32)        9248      
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 32)          0         
_________________________________________________________________
flatten (Flatten)            (None, 1152)              0         
_________________________________________________________________
dense (Dense)                (None, 128)               147584    
_________________________________________________________________
flatten_1 (Flatten)          (None, 128)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256      
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 130       
=================================================================
Total params: 170,306
Trainable params: 170,306
Non-trainable params: 0
```

Figure 3. 15 Summary of the Architecture of Convolutional Neural Network

The architecture of the network provided in this picture shows how to evaluate the accuracy of the dataset.

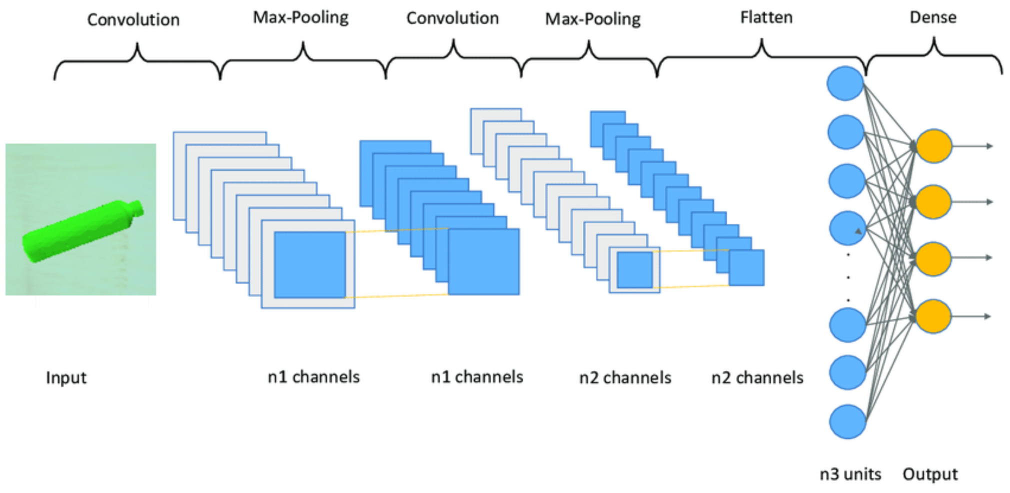
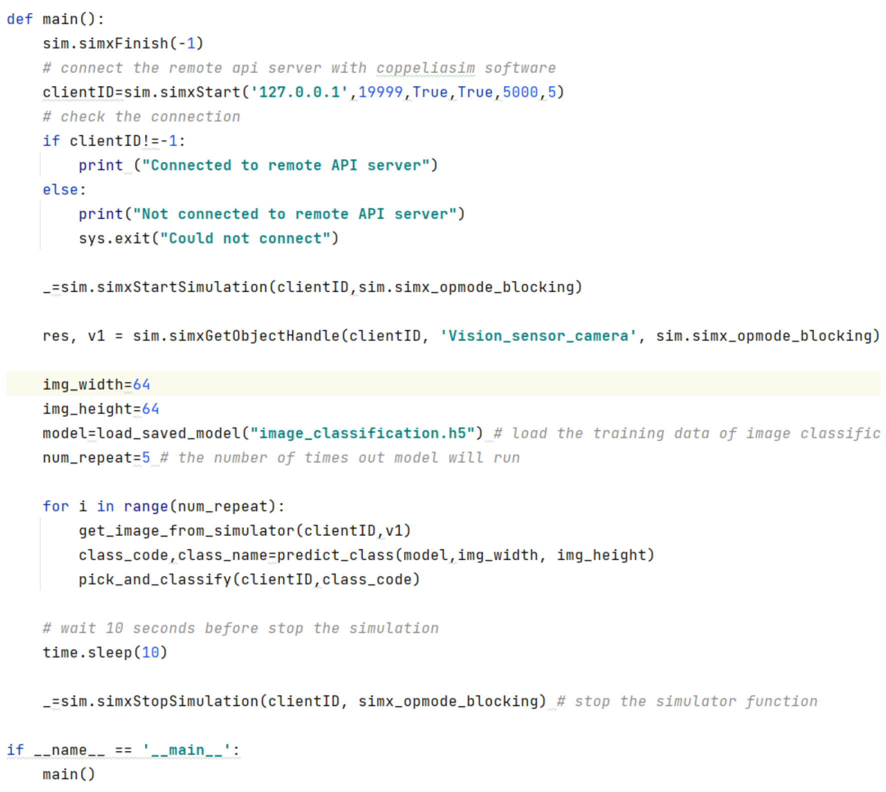


Figure 3. 16 Architecture of Convolutional Neural network

After running the training network file, they have setup a model which will follow their layer of steps for data classification and it will be saved in the "image_classification.h5" file. In addition, it will also give information on data accuracy and the prediction of one product image.

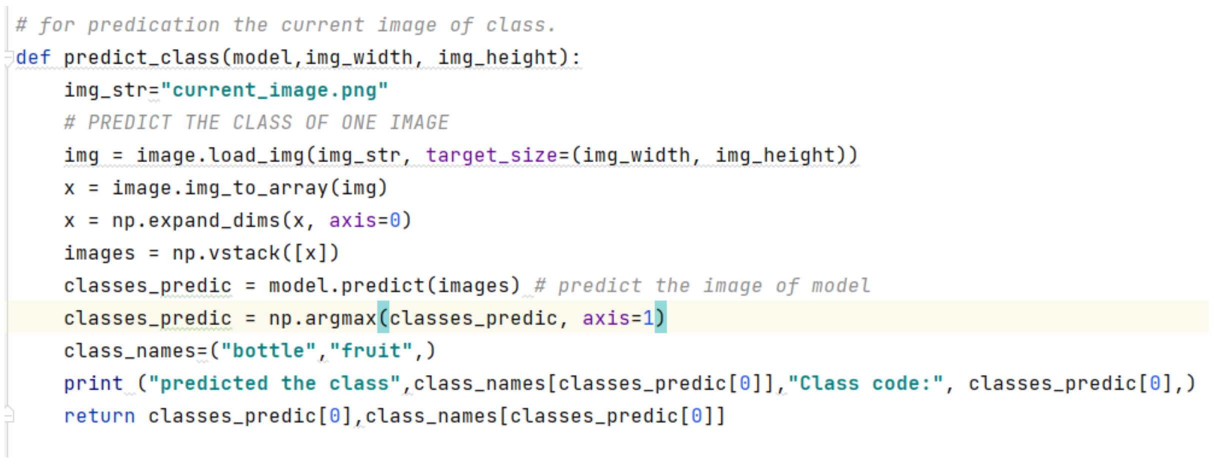
Furthermore, first of all, the run_the_simulator.py file will connect with the coppeliaSim simulator api and load the "image_classification.h5" file for classifying images. The following code has been shared with this figure 3.17.



```python
def main():
    sim.simxFinish(-1)
    # connect the remote api server with coppeliasim software
    clientID=sim.simxStart('127.0.0.1',19999,True,True,5000,5)
    # check the connection
    if clientID!=-1:
        print ("Connected to remote API server")
    else:
        print("Not connected to remote API server")
        sys.exit("Could not connect")

    _=sim.simxStartSimulation(clientID,sim.simx_opmode_blocking)

    res, v1 = sim.simxGetObjectHandle(clientID, 'Vision_sensor_camera', sim.simx_opmode_blocking)

    img_width=64
    img_height=64
    model=load_saved_model("image_classification.h5") # load the training data of image classific
    num_repeat=5 # the number of times out model will run

    for i in range(num_repeat):
        get_image_from_simulator(clientID,v1)
        class_code,class_name=predict_class(model,img_width, img_height)
        pick_and_classify(clientID,class_code)

    # wait 10 seconds before stop the simulation
    time.sleep(10)

    _=sim.simxStopSimulation(clientID, simx_opmode_blocking) # stop the simulator function

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

Figure 3. 17 run simulator code

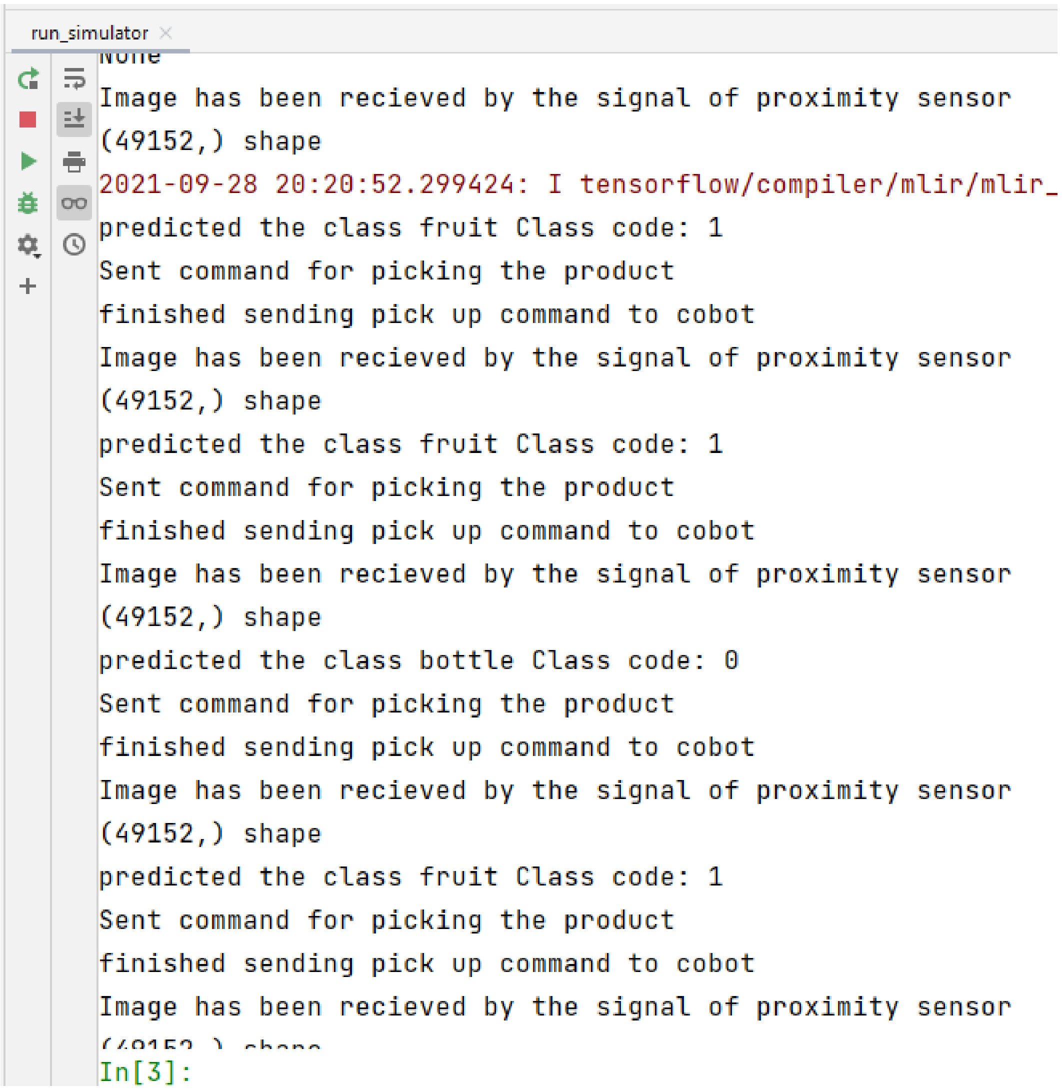
Next, call the get_image_from_simulator() function, which will take the image with a vision sensor and save it in the "current_image.png" file. After taking the image, there is also a function called the predict_class () function which will help predict the image class type and number.



```python
# for predication the current image of class.
def predict_class(model,img_width, img_height):
    img_str="current_image.png"
    # PREDICT THE CLASS OF ONE IMAGE
    img = image.load_img(img_str, target_size=(img_width, img_height))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    images = np.vstack([x])
    classes_predic = model.predict(images) # predict the image of model
    classes_predic = np.argmax(classes_predic, axis=1)
    class_names=("bottle","fruit",)
    print ("predicted the class",class_names[classes_predic[0]],"Class code:", classes_predic[0],)
    return classes_predic[0],class_names[classes_predic[0]]
```

Figure 3. 18 prediction of class type and number code

We see in figure 3.18 that there has been called the "current_image.png" file for load to give the output of predicted class types and numbers. Finally, in the console of the code editor, we can see the output of the prediction class type and number that have been shared in figure 3.19. There we can see the class code and class type of data, which will help us compare with real data.



```
run_simulator
Image has been recieved by the signal of proximity sensor
(49152,) shape
2021-09-28 20:20:52.299424: I tensorflow/compiler/mlir/mlir_
predicted the class fruit Class code: 1
Sent command for picking the product
finished sending pick up command to cobot
Image has been recieved by the signal of proximity sensor
(49152,) shape
predicted the class fruit Class code: 1
Sent command for picking the product
finished sending pick up command to cobot
Image has been recieved by the signal of proximity sensor
(49152,) shape
predicted the class bottle Class code: 0
Sent command for picking the product
finished sending pick up command to cobot
Image has been recieved by the signal of proximity sensor
(49152,) shape
predicted the class fruit Class code: 1
Sent command for picking the product
finished sending pick up command to cobot
Image has been recieved by the signal of proximity sensor
In[3]:
```

Figure 3. 19 CNN prediction output result

Chapter 4: results

[29] The integration of artificial intelligence (AI), machine learning (ML), and industrial robotics technologies into a single manufacturing production system has many advantages.[21] Formalized paraphrase Today's manufacturers profit greatly from increased uptime, decreased programming time, and increased production. There we have experimented to see how the next generation of industrial robots will be advanced by Artificial Intelligence technology to make a project which is a grocery product distribution system. As a result, we have found some specific data to check real predictions and deep learning predictions. How do they predict and identify the product correctly with a convolutional neural network?

4.1 CoppeliaSim Simulator Result

To begin, we played the simulator with a run programme where we observed to get the data from all the sensors and class types in real time. It was run for 5 products again and again to get the output data shared on Table 4.1, where we saw that the class of products the system identified correctly and the takeProducclose sensor, takeproduct sensor, and captureProductPicture sensor all worked perfectly with 1 value of output. In general, the detected_object sensor identifies the different products and shows the value of the output data. It is clear, Take_for_classification made two mistakes and wasn't able to select the product and get the value of output 0.

Quantity	**Class**	**Take Product close**	**Take product**	**Detected _object**	**Capture Product Picture**	**Take _for_ classification**
1	Bottle	1	1	309	1	1
2	Fruit	1	1	311	1	0

3	Fruit	1	1	313	1	1
4	Bottle	1	1	315	1	1
5	Bottle	1	1	317	1	0

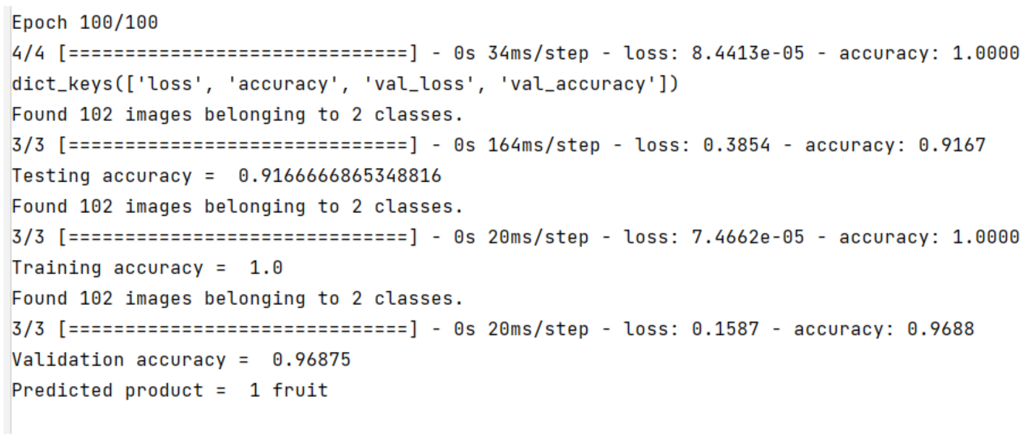
Table 4. 1 CoppeliaSim simulator sensors signal output and type of class

4.2 Machine Learning and Deep Learning Result

[45] When working with high dimensional raw data, deep neural networks allow for strong feature learning that would be impossible to accomplish with hand-crafted feature engineering. This implementation's goal is to build a DL model that can identify the product category and provide that information to the programme editor's console.

4.2.1 Deep learning training accuracy output

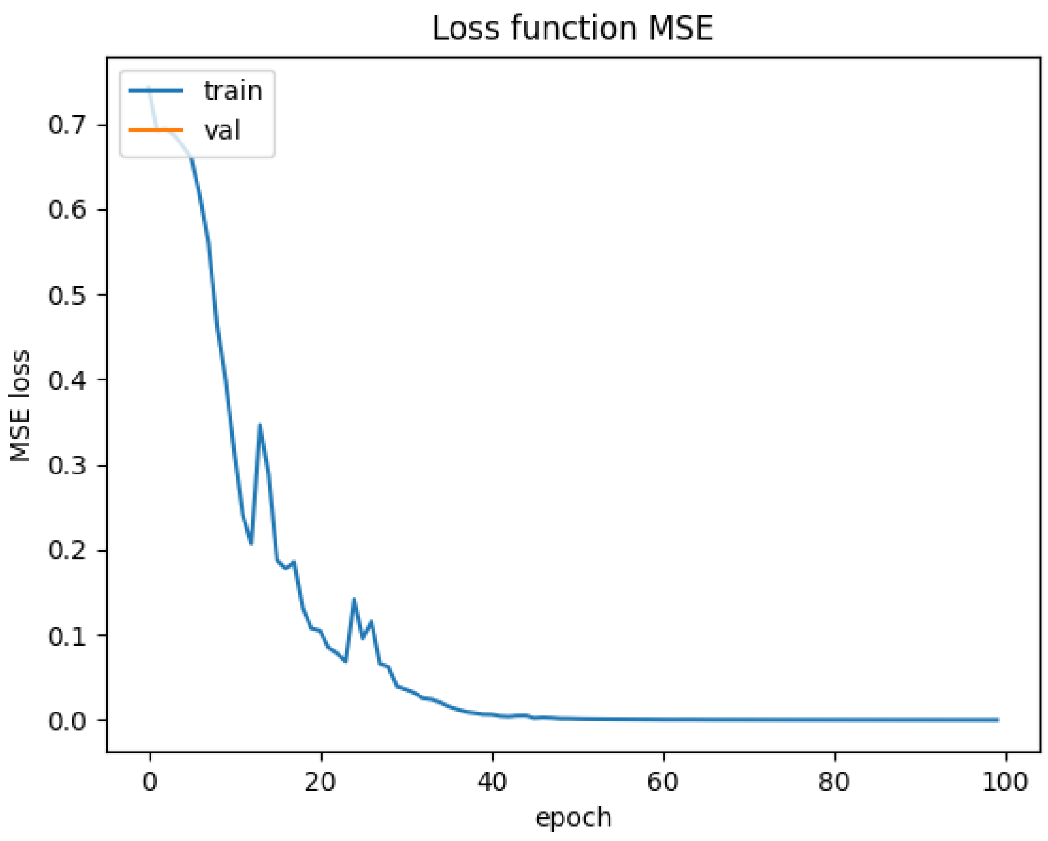
To train the data, we have run the training file where we have got some accuracy in testing, training, and validation data output results. It can be clearly seen that after the epoch there are 102 images belonging to 2 classes. As it was observed, the testing accuracy was 0.9166666865348816, which is a good accuracy result. It is interesting to note that the training accuracy is 1.0, which is a perfect output result. Similarly, the output result of validation accuracy is 0.96875. Overall, accuracy data depends on the images. How many images have been included for classification? If there are more images imported, then the result will come more accurately. Finally, one prediction was set to predict the product from the current image, where the predicted product is 1 fruit.



```
Epoch 100/100
4/4 [==============================] - 0s 34ms/step - loss: 8.4413e-05 - accuracy: 1.0000
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
Found 102 images belonging to 2 classes.
3/3 [==============================] - 0s 164ms/step - loss: 0.3854 - accuracy: 0.9167
Testing accuracy =  0.9166666865348816
Found 102 images belonging to 2 classes.
3/3 [==============================] - 0s 20ms/step - loss: 7.4662e-05 - accuracy: 1.0000
Training accuracy =  1.0
Found 102 images belonging to 2 classes.
3/3 [==============================] - 0s 20ms/step - loss: 0.1587 - accuracy: 0.9688
Validation accuracy =  0.96875
Predicted product =  1 fruit
```

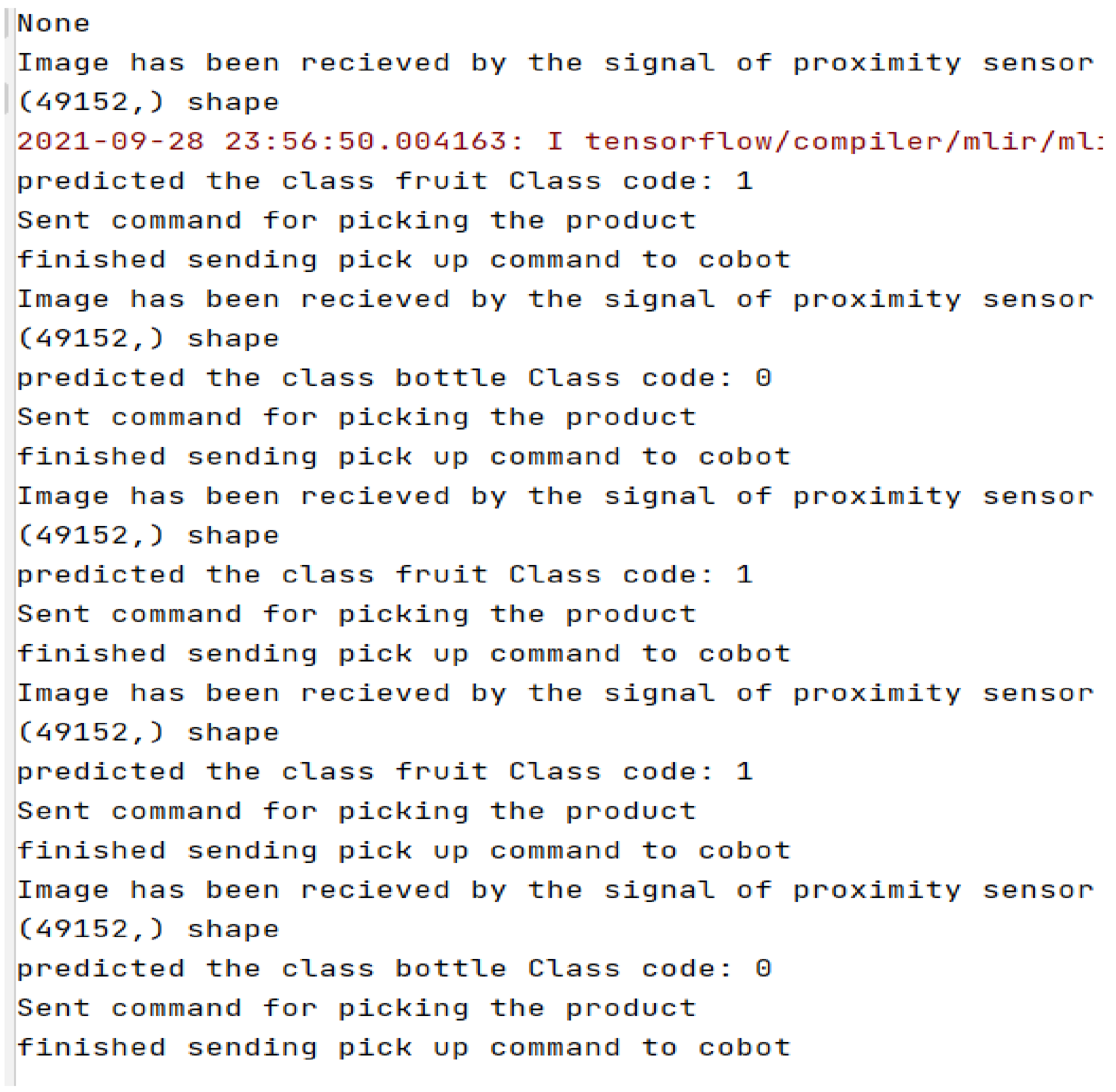
4.2.2 Training epoch and MSE graph

After training the data, they saved one graph output result image, "training_output.png". The graph depicts the MSE loss and epoch result for training and validation data.As it was observed, from the beginning there were so many losses in epoch data accuracy, but at the end of the epoch, the data achieved stabilised accuracy. We saw that the validation accuracy is unchanged in the network procedure.



4.2.3 Deep learning prediction output

The final prediction result is found after running the simulator with the ai system. It could be plainly seen that the vision sensor took the image and predicted that image with the classification of the dataset five times. Firstly, it shows the result of a class type fruit, and the class code is 1. Secondly, it predicted the product of bottle type and class code is 0. It is clear from the data that the 3rd and 4th times it predicted the product's fruit type and class code are 1. Finally, in the 5th position of prediction, it was predicted that the bottle type and class code would be 0.



```
None
Image has been recieved by the signal of proximity sensor
(49152,) shape
2021-09-28 23:56:50.004163: I tensorflow/compiler/mlir/ml:
predicted the class fruit Class code: 1
Sent command for picking the product
finished sending pick up command to cobot
Image has been recieved by the signal of proximity sensor
(49152,) shape
predicted the class bottle Class code: 0
Sent command for picking the product
finished sending pick up command to cobot
Image has been recieved by the signal of proximity sensor
(49152,) shape
predicted the class fruit Class code: 1
Sent command for picking the product
finished sending pick up command to cobot
Image has been recieved by the signal of proximity sensor
(49152,) shape
predicted the class fruit Class code: 1
Sent command for picking the product
finished sending pick up command to cobot
Image has been recieved by the signal of proximity sensor
(49152,) shape
predicted the class bottle Class code: 0
Sent command for picking the product
finished sending pick up command to cobot
```

Figure 4. 1 Deep learning output prediction

Quantity	**Class Type**	**Class Code**
1	Fruit	1
2	Bottle	0
3	Fruit	1
4	Fruit	1
5	Bottle	0

Table 4. 2 Deep learning output result of classes

To conclude, the coppeliaSim simulator detected all the products correctly and the take_for_classification sensor didn’t detect them twice. In addition, the training accuracy result was very good without loss of layers. Finally, the deep learning prediction predicted around 80% of 5 products' detection.

Chapter 5: Discussion and analysis of findings

We have demonstrated the design and implementation of a Cobot with artificial intelligence. To the best of our knowledge, this is the first implementation of real-time detection of products and artificial intelligence prediction of product comparison output results. Moreover, these features enable the proposed CNN-based method to achieve the correct classification of product types in the network. There we have found DL works perfectly with the vision sensor data. The data suggests that artificial intelligence can predict the classification with the highest label accuracy, nearing similar real output results. As we have seen in 5 data points, they both predict the same type 4 times, leaving one which wasn't perfectly predicted. The analysis identifies the accuracy of training data perfectly. In quantitative evaluations, there needs to be more datasets in the training and validation folders which will give a more accurate prediction of class type. For future work, we will try to find the defecated products with this vision sensor and deep learning, so when defecated products come on the conveyor belt, the network of CNN will detect the defecated product and pass it to the defecated product basket.

Chapter 6: Conclusion and recommendations

[33] Early users of these new robotic systems are benefiting from the convergence of robotics and artificial intelligence technologies. AI and machine learning are often used by collaborative robots, and they may be designed to do many jobs at once while also learning to operate securely with people.[11] Formalized artificial intelligence (AI), machine learning (ML), and industrial robotics (IR) technology has many advantages in manufacturing output. This study aimed to investigate how industrial robots and AI work correspondingly in the same workplace and did the project work. Further findings show that these predictions of output results are both near to similar in percentage. However, the grocery product distribution system makes this project one of the greatest improvement potentials found in the case study. It uses information from Cobots and deep learning. The results presented in this part of the dissertation are considered as contributions toward the goal of having next generation industrial robotics advanced by artificial intelligence technology. It will help to work collaboratively and accurately with output.

References

[1] S. AUDO, “School of Innovation, Design, and Engineering A study on Cobot investment in the manufacturing industry Master thesis work 30 credits, Advanced level Product and process development Production and Logistics SANDRA AUDO Report code: Tutor (university): Mik,” Accessed: Aug. 26, 2021. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1345097/FULLTEXT01.pdf.

[2] T. H. E. Challenge, “NEXT-GENERATION INDUSTRIAL ROBOTIC CAPABILITIES ADVANCED BY ARTIFICIAL INTELLIGENCE,” [Online]. Available: https://www.controldesign.com/assets/wp_downloads/pdf/2020/11/Wind-River-AI-Robotics-Use-Case.pdf.

[3] McKinsey, “Industrial robotics - Insights into the sector’s grwoth dynamics,” *Mckinsey Co.*, no. July, pp. 1–33, 2019, [Online]. Available: https://www.mckinsey.com/industries/advanced-electronics/our-insights/growth-dynamics-in-industrial-robotics.

[4] J. Wallén, “The history of the industrial robot,” 2008, Accessed: Aug. 16, 2021. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:316930/FULLTEXT01.pdf.

[5] Alex Misiti, “A History of Industrial Robots,” *Wevolver*, Aug. 14, 2020. https://www.wevolver.com/article/a-history-of-industrial-robots (accessed Aug. 16, 2021).

[6] Mariane Davids, “A Brief History of Robots in Manufacturing,” *blog robotiq*, Jul. 17, 2017. https://blog.robotiq.com/a-brief-history-of-robots-in-manufacturing (accessed Aug. 16, 2021).

[7] A. Hitchcox, “Robots - the Next Generation.,” *Power Transm. Des.*, vol. 23, no. 10,

1981.

[8] T. H. E. Challenge and T. H. E. Approach, “COMPUTER VISION AND ROBOTICS USE CASE COMPUTER VISION AND ROBOTICS EXPAND Computer Vision Changes the Scope of Industrial Robotics,” 1954.

[9] K. Xia and Z. Weng, “Workpieces sorting system based on industrial robot of machine vision,” *2016 3rd Int. Conf. Syst. Informatics, ICSAI 2016*, pp. 422–426, Jan. 2017, doi: 10.1109/ICSAI.2016.7810992.

[10] K. Romanainen, Jari ; Perez, Maialen ; Roman, Laura ; Izsak, “Advanced technologies for industry,” no. July, p. 27 p., 2021, [Online]. Available: https://op.europa.eu/en/publication-detail/-/publication/387ef7b6-b9e0-11eb-8aca-01aa75ed71a1/language-en/format-PDF.

[11] M. L. Daniel Küpper, “Advanced Robotics in the Factory of the Future,” *bcg*, Mar. 27, 2019. https://www.bcg.com/publications/2019/advanced-robotics-factory-future (accessed Aug. 18, 2021).

[12] R. Devin Partida, “Robotics Changed Industry 4.0; Now, How Will Industry 4.0 Change Robotics? | RoboticsTomorrow,” *roboticstomorrow*, Feb. 08, 2021. https://www.roboticstomorrow.com/story/2021/07/robotics-changed-industry-40-now-how-will-industry-40-change-robotics/17226/ (accessed Aug. 18, 2021).

[13] C. of S. M. Nigel Smith, “Tips for choosing a 3D vision system - The future of vision systems in manufacturing | RoboticsTomorrow,” *roboticstomorrow*, Jul. 30, 2021. https://www.roboticstomorrow.com/story/2021/07/tips-for-choosing-a-3d-vision-system-the-future-of-vision-systems-in-manufacturing-/17184/ (accessed Aug. 18, 2021).

[14] M. Edwards, “Robots in industry: An overview,” *Appl. Ergon.*, vol. 15, no. 1, pp. 45–53, Mar. 1984, doi: 10.1016/S0003-6870(84)90121-2.

[15] F. D. Araya, P. J. Sullivan, and P. A. Heinricher, “Project Number : IQP-2845 Role of Industrial Robotics Automation Systems An Interactive Qualifying Project Submitted to the Faculty of the In partial fulfillment of the requirements for the degree of Bachelor of Science by Robotics Engineering Approved :,” 2019.

[16] M. Slamani, A. Nubiola, and I. Bonev, “Assessment of the positioning performance of an industrial robot,” *Ind. Rob.*, vol. 39, no. 1, pp. 57–68, 2012, doi: 10.1108/01439911211192501.

[17] R. Spiegel, “Do You Want a Fast, Hard Traditional Robot or a Soft, Slow Cobot?,” *designnews*, 2018. https://www.designnews.com/automation-motion-control/do-you-want-fast-hard-traditional-robot-or-soft-slow-cobot (accessed Aug. 19, 2021).

[18] T. Dietz, A. Pott, and A. Verl, “Practice for planning and realization of advanced industrial robot systems,” *2013 44th Int. Symp. Robot. ISR 2013*, 2013, doi: 10.1109/ISR.2013.6695744.

[19] Wevolver Staff, “Industry 4.0 Deep Dive. Part 4: Evolving Industrial Robots,” *wevolver*, May 26, 2020. https://www.wevolver.com/article/industrie.40.deep.dive.part.4.evolving.industrial.robots (accessed Aug. 19, 2021).

[20] J. Edwards, “Building a Smart Factory with AI and Robotics,” 2018.

[21] M. Faccio, M. Bottin, and G. Rosati, “Collaborative and traditional robotic assembly: a comparison model,” *Int. J. Adv. Manuf. Technol. 2019 1025*, vol. 102, no. 5, pp. 1355–1372, Jan. 2019, doi: 10.1007/S00170-018-03247-Z.

[22] Steven Keeping, “Designing Collaborative Robots: Maximizing Productivity and Safety,” *wevolver*, Jul. 27, 2020. https://www.wevolver.com/article/designing-collaborative-robots-maximizing-productivity-and-safety (accessed Aug. 19, 2021).

[23] P. By, “Collaborative Robotics 2021 - New Systems , Applications and Opportunities -

,” 2021.

[24] M. Shirine El Zaataria, “Cobot programming for collaborative industrial tasks: An overview - ScienceDirect,” *ScienceDirect*, Jun. 10, 2019. https://www-sciencedirect-com.salford.idm.oclc.org/science/article/pii/S092188901830602X (accessed Aug. 19, 2021).

[25] S. Kianoush, S. Savazzi, M. Beschi, S. Sigg, and V. Rampa, “A Multisensory Edge-Cloud Platform for Opportunistic Radio Sensing in Cobot Environments,” *IEEE Internet Things J.*, vol. 8, no. 2, pp. 1154–1168, Jan. 2021, doi: 10.1109/JIOT.2020.3011809.

[26] A. Cherubini and D. Navarro-Alarcon, “Sensor-Based Control for Collaborative Robots: Fundamentals, Challenges, and Opportunities,” *Front. Neurorobot.*, vol. 0, p. 113, Jan. 2021, doi: 10.3389/FNBOT.2020.576846.

[27] Cory Roehl, “Know Your Machine: Industrial Robots vs.Collaborative Robots,” *universalrobots*, Oct. 12, 2017. https://www.universal-robots.com/blog/know-your-machine-industrial-robots-vs-cobots/ (accessed Aug. 20, 2021).

[28] Carlos Gonzalez, “7 Common Applications for Cobots | Machine Design,” *machinedesign*, Jan. 19, 2018. https://www.machinedesign.com/mechanical-motion-systems/article/21836350/7-common-applications-for-cobots (accessed Aug. 20, 2021).

[29] “Automate assembly lines with collaborative robots | cobots | Numatic Engineering.” https://www.collaborativerobotarm.com/robot-assembly-line (accessed Aug. 24, 2021).

[30] Carlos Gonzalez, “7 Common Applications for Cobots | Machine Design,” *Machinedisign*, Jan. 19, 2018. https://www.machinedesign.com/mechanical-motion-systems/article/21836350/7-common-applications-for-cobots (accessed Aug. 24,

2021).

[31] "What are the Most Popular Applications for Cobots?," *Automate*, Jan. 14, 2020. https://www.automate.org/blogs/what-are-the-most-popular-applications-for-cobots (accessed Aug. 26, 2021).

[32] M. Beaupre, "Collaborative Robot Technology and Applications Collaborative Robots."

[33] H. Ding, M. Schipper, and B. Matthias, "Collaborative behavior design of industrial robots for multiple human-robot collaboration," *2013 44th Int. Symp. Robot. ISR 2013*, 2013, doi: 10.1109/ISR.2013.6695707.

[34] Z. M. Bi, M. Luo, Z. Miao, B. Zhang, W. J. Zhang, and L. Wang, "Safety assurance mechanisms of collaborative robotic systems in manufacturing," *Robot. Comput. Integr. Manuf.*, vol. 67, p. 102022, Feb. 2021, doi: 10.1016/J.RCIM.2020.102022.

[35] C. Byner, B. Matthias, and H. Ding, "Dynamic speed and separation monitoring for collaborative robot applications – Concepts and performance," *Robot. Comput. Integr. Manuf.*, vol. 58, pp. 239–252, Aug. 2019, doi: 10.1016/J.RCIM.2018.11.002.

[36] John Blyler, "The Complexities of Mimicking Humans is Just the Beginning," *wevolver*, Aug. 20, 2020. https://www.wevolver.com/article/the-complexities-of-mimicking-humans-is-just-the-beginning (accessed Aug. 20, 2021).

[37] Barry Manz, "Cobots Are Collaborators. AI Will Make Them Partners," *wevolver*, Sep. 17, 2020. https://www.wevolver.com/article/cobots-are-collaborators-ai-will-make-them-partners (accessed Aug. 26, 2021).

[38] Joe Campbell, "Collaborative Robots Versus Traditional Industrial Robots," Mar. 13, 2020. https://www.universal-robots.com/blog/how-to-choose-between-collaborative-and-traditional-industrial-robots/ (accessed Aug. 26, 2021).

[39] Dr Kumar Gaurav, "CoppeliaSim: The Open Source Tool for Robotic Simulations,"

opensourceforu. https://www.opensourceforu.com/2021/03/coppeliasim-the-open-source-tool-for-robotic-simulations/ (accessed Sep. 27, 2021).

[40] Marc Freese, “CoppeliaSim, Successor of V-REP, was Released!” https://www.linkedin.com/pulse/coppeliasim-successor-v-rep-released-marc-freese (accessed Sep. 27, 2021).

[41] A. L. Samuel, “‘Some Studies in Machine Learning Using the Game of Checkers,’” 1959.

[42] and X. Z. Q. Li, J. Zhao, “‘An Unsupervised Learning Algorithm for Intelligent Image Analysis,.’” 9th International Conference on Control, Automation, Robotics and Vision, pp. 1–5, 2006.

[43] Sumit Saha, “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science,” *towardsdatascience*. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (accessed Sep. 27, 2021).

[44] Serdar Yegulalp, “What is TensorFlow? The machine learning library explained | InfoWorld,” *infoworld*. https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html (accessed Sep. 27, 2021).

[45] Y. J. Heo, D. Kim, W. Lee, H. Kim, J. Park, and W. K. Chung, “Collision detection for industrial collaborative robots: A deep learning approach,” *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 740–746, Apr. 2019, doi: 10.1109/LRA.2019.2893400.

Appendices

Conveyor belt program code:

```
 -- This script is for handles the conveyor belt motion

function sysCall_init()

    forwarder=sim.getObjectHandle('Products_Conveyor_forwarder')

    textureShape=sim.getObjectHandle('Products_Conveyor_texture')

end




function sysCall_actuation()


            r,dist,pt,detectedObjectHandle=sim.readProximitySensor(sim.getObjectHandle('Product_stop_for_pickup_sensor')) -- to identify the product then stop


    if (r>0) then

        sim.setIntegerSignal('takeProduct',1) --Sets the value of an integer signal. If that signal is not yet present, it is added.

        sim.setIntegerSignal("detected_object",sim.getObjectParent(detectedObjectHandle)) --Sets the value of an detected object integer signal.

        mult=0


beltVelocity=sim.getScriptSimulationParameter(sim.handle_self,"conveyorBeltVelocity")*mult -- to get the data of beltvelocity
```

```
   else
      sim.clearIntegerSignal('takeProduct') -- Clears an integer signal (removes it).
      mult=1

beltVelocity=sim.getScriptSimulationParameter(sim.handle_self,"conveyorBeltVelocity")*m
ult


   end
         r,dist,pt,detectedObjectHandle=sim.readProximitySensor(sim.getObjectHandle('Produ
ct_picking_close_identifier_sensor'))
   if (r>0) then
      sim.setIntegerSignal('takeProductClose',1) --Sets the value of an integer signal. If that
signal is not yet present, it is added.
   else
      sim.clearIntegerSignal('takeProductClose') -- Clears an integer signal (removes it).
   end

         r,dist,pt,detectedObjectHandle=sim.readProximitySensor(sim.getObjectHandle('Produ
ct_identifier_for_vision_sensor'))
   if (r>0) then
      sim.setIntegerSignal('captureProductPicture',1) --Sets the value of an integer signal for
get the image data.
      name=sim.getObjectName(detectedObjectHandle)
```

```
        start_name=string.sub(name, 1, 1)

        if (start_name=="b") then

            sim.setStringSignal('class',"bottle") -- if it bottle it will set the bottle signal

        end

        if (start_name=="f") then

            sim.setStringSignal('class',"fruit") -- if it fruit it will set the bottle signal

        end

    else

        sim.clearIntegerSignal('captureProductPicture')  -- Clears an integer signal (removes it).

    end


--------------------


    t=sim.getSimulationTime()

    -- We move the texture attached to the conveyor belt to give the impression of movement:

    sim.setObjectFloatParameter(textureShape,3007,t*beltVelocity)



    -- Here we "fake" the transportation pads with a single static rectangle that we dynamically
reset

    -- at each simulation pass (while not forgetting to set its initial velocity vector) :

    relativeLinearVelocity={beltVelocity,0,0}

    -- Reset the dynamic rectangle from the simulation (it will be removed and added again)

    sim.resetDynamicObject(forwarder)

    -- Compute the absolute velocity vector:
```

```
    m=sim.getObjectMatrix(forwarder,-1)

    m[4]=0 -- Make sure the translation component is discarded

    m[8]=0 -- Make sure the translation component is discarded

    m[12]=0 -- Make sure the translation component is discarded

    absoluteLinearVelocity=sim.multiplyVector(m,relativeLinearVelocity)

    -- Now set the initial velocity of the dynamic rectangle:

    sim.setObjectFloatParameter(forwarder,3000,absoluteLinearVelocity[1])

    sim.setObjectFloatParameter(forwarder,3001,absoluteLinearVelocity[2])

    sim.setObjectFloatParameter(forwarder,3002,absoluteLinearVelocity[3])

end
```

Product Generator program code:

```
 -- This script is for handles the product to generate for ongoing the conveyor belt

colorCorrectionFunction=function(_aShapeHandle_)

    local version=sim.getInt32Parameter(sim.intparam_program_version)

    local revision=sim.getInt32Parameter(sim.intparam_program_revision)

    if (version<30104)and(revision<3) then

        return _aShapeHandle_

    end

    return '@backCompatibility1:'.._aShapeHandle_

end

--------------------------------------------------------------------------




-- This script is in charge of generatin shapes of color the product.
```

```
function sysCall_init()
    math.randomseed( os.time() )
    ProductProducer=sim.getObjectAssociatedWithScript(sim.handle_self)
    -- To get the all fruit products object data

fruit={sim.getObjectHandle('fruit_1'),sim.getObjectHandle('fruit_2'),sim.getObjectHandle('fr
uit_3')}

fruit_r={sim.getObjectHandle('fruit_1_r'),sim.getObjectHandle('fruit_2_r'),sim.getObjectHan
dle('fruit_3_r')}

    -- To get the all bottle products object data

bottle={sim.getObjectHandle('bottle_1'),sim.getObjectHandle('bottle_2'),sim.getObjectHandl
e('bottle_3')}

bottle_r={sim.getObjectHandle('bottle_1_r'),sim.getObjectHandle('bottle_2_r'),sim.getObject
Handle('bottle_3_r')}

    -- for coloring the products
    colors_bottle={{2,10,1},{74,172,233},{37,20,5},{9,5,1}}
    colors_fruit={{122,252,255},{136,117,57},{118,251,255}}
```

```
   ProductPosObj=sim.getObjectHandle('Product_position_for_flow_conveyor') -- to get the
product flow object position
   lastDropTime=0
end



create_shapes=function(piece,piece_r,type)
        objBase=piece
        objBase_r=piece_r

        copy=sim.copyPasteObjects({objBase},0) -- to take the copy of object data
        copy_r=sim.copyPasteObjects({objBase_r},0)

        copy_r=copy_r[1]
        copy=copy[1]
        -- Give the object a random position and orientation around the flow conveyor drop
position:
        m=sim.getObjectMatrix(copy_r,-1)  -- Retrieves the transformation matrix of an object.
        p=sim.getObjectPosition(ProductPosObj,-1) -- get the product flowing position

m2=sim.buildMatrix({0,0,0},{0,(math.random())*2*math.pi,(math.random())*2*math.pi})  -
- Builds a transformation matrix based on a position vector and Euler angles.
        m2=sim.buildMatrix({0,0,0},{0,0,(-0.5+math.random())*2*math.pi}) --Builds a
transformation matrix based on a position vector and Euler angles.
```

```
    m=sim.multiplyMatrices(m2,m)  -- Multiplies two transformation matrices.
    m[4]=p[1]+(math.random())*0.15
    m[8]=p[2]
    m[12]=p[3]
    sim.setObjectMatrix(copy_r,-1,m) -- Sets the transformation matrix of an object.
Dynamically simulated objects will implicitely be reset before the command is applied
    sim.setObjectMatrix(copy,-1,m)
    -- Make the copied object dynamic (i.e. non-static):
    sim.setObjectParent(copy,copy_r,true)  -- Sets an object's parent object.
    sim.setObjectInt32Parameter(copy_r,3003,0)  --Sets an int32 parameter of a scene object
or calculation object.

    -- Now change the color:
    if type=="fruit" then
      p_n=2
      val=math.random(1,p_n+1)
      if val~=p_n+1 then
        sim.setShapeColor(colorCorrectionFunction(copy),nil,0,colors_fruit[val]) -- Sets
the color (or transforms it) of one or several shapes.
      end
    end
    if type=="bottle" then
      p_g=4
```

```
            child=sim.getObjectChild(copy,0) -- the T shape's visible component is child of this
object
            if (child==-1) then
                child=copy
            end
            val=math.random(1,p_g+1)
            if val ~=p_g+1 then
                sim.setShapeColor(colorCorrectionFunction(child),nil,0,colors_bottle[val])   -- Sets
the color (or transforms it) of one or several shapes.
            end
        end
    end



function sysCall_actuation()


    t=sim.getSimulationTime() -- to get the current simulation time
            stop,dist,pt,detectedObjectHandle=sim.readProximitySensor(sim.getObjectHandle('Pr
oduct_detection'))


stop2,dist,pt,detectedObjectHandle=sim.readProximitySensor(sim.getObjectHandle('Product
_identifier_for_vision_sensor'))
    mult=0.25
```

```
shapeDropFrequency=sim.getScriptSimulationParameter(sim.getScriptAssociatedWithObject
(ProductProducer),'shapeDropFrequency')*mult

   if (t-lastDropTime>(1/shapeDropFrequency)) then
      -- to drop a shape!
      lastDropTime=t
      if (stop==0 and stop2==0) then
         item_type=math.random(1,2) -- to take the random of item type product
         item=math.random(1,3) -- to take the random of item from list

         if (item_type==1) then
            -- call the create_shaped function to make a new shape randomly position
           create_shapes(bottle[item],bottle_r[item],"bottle")
         elseif (item_type==2) then
           create_shapes(fruit[item],fruit_r[item],"fruit")
         end
      end
   end
end
```

Vision Sensor Program code:

```
 -- This script is for handles the Vision sensor camera setup to take the image of product
function sysCall_init()
   camera=sim.getObjectHandle('Vision_sensor_camera') -- to get the vision camera data
```

```
    kn=0;
    baseObject=sim.getObjectAssociatedWithScript(sim.handle_self) -- Retrives the handle of
the object the script is attached to.
    conveyor=sim.getObjectHandle('Products_Conveyor')  -- conveyor variable
    res=sim.getVisionSensorResolution(camera) -- to get camera resulation data
    lastImageAnalysisTime=0
    lastLowestBlobDetectionPositionY=100
    lastLowestBlobDetectionTime=0
    object=0;
    detected=1

    getPixel=function(img,x,y, maxY) -- to get the pixel of image
    --grey scale image from 0 to 255
        return (math.floor(img[(x-1)*maxY + y]*255))
    end

    setPixel=function(img,x,y,maxY,value) -- to set the pixel of image
        img[(x-1)*maxY + y] = value/255
    end

    copy = function(orig)
        local orig_type = type(orig)
        local copy
        if orig_type == 'table' then
```

```
            copy = {}
            for orig_key, orig_value in pairs(orig) do
                copy[orig_key] = orig_value
            end
        else -- number, string, boolean, etc
            copy = orig
        end
        return copy
    end
end



get_image=function()
    sim.clearIntegerSignal('count'); -- clear the signal of counting image
    t=sim.getSimulationTime() -- to get the current simulation time
    camPos=sim.getObjectPosition(camera,-1) -- to know the object position of the camera

    sim.handleVisionSensor(camera) -- Handles (performs sensing, etc. of) a registered vision
sensor object.
    img= sim.getVisionSensorImage(camera+sim.handleflag_greyscale)  -- Retrieves the rgb-
image (or a portion of it) of a vision sensor.

end
```

```
function sysCall_actuation()


    get_image_signal=sim.getIntegerSignal("captureProductPicture") -- to get the image
picture
    if (detected==0 and get_image_signal==1) then
        get_image()
    end


    if (get_image_signal==nil) then
        detected=0
    elseif (get_image_signal==1) then
        detected=1
    end
end
```

Franka Cobot Program code:

```
 -- This script is for cobot which will pick the product and place it specific place
getObject=function(fingerAngle,relVel,detected_object)
    data=sim.packInt32Table({0})..sim.packFloatTable({fingerAngle})
    sim.tubeWrite(gripperCommunicationTube,data)
    sim.setExplicitHandling(ikTask1,1)
    sim.setExplicitHandling(ikTask2,1)


sim.moveToJointPositions(joints,upperPickupJointPos,jointVelocity*relVel,jointAcceleration
*relVel)
```

```
    m=sim.getObjectMatrix(tip,-1)

    sim.setObjectMatrix(target,-1,m)

    sim.setExplicitHandling(ikTask1,0)

    sim.setExplicitHandling(ikTask2,0)


sim.moveToPosition(target,Franka,pickupPos,pickupOrientation,movementVelocity*relVel,
movementAcceleration*relVel)

    data=sim.packInt32Table({1})..sim.packFloatTable({fingerAngle})

    sim.tubeWrite(gripperCommunicationTube,data)

    sim.wait(1.5)

    grasp(detected_object)


sim.moveToPosition(target,Franka,{pickupPos[1],pickupPos[2],pickupPos[3]+0.1},pickupOr
ientation,movementVelocity*relVel,movementAcceleration*relVel)


end

grasp =function(obj)

    sim.setObjectParent(obj,connector,true)

end


release=function(obj)

    sim.setObjectParent(obj,-1,false)

end
```

```
moveToIntermediateDropPos=function(relVel)
    sim.setExplicitHandling(ikTask1,1)
    sim.setExplicitHandling(ikTask2,1)

sim.moveToJointPositions(joints,intermediateDropPos,jointVelocity*relVel,jointAcceleratio
n*relVel)
    m=sim.getObjectMatrix(tip,-1)
    sim.setObjectMatrix(target,-1,m)
    sim.setExplicitHandling(ikTask1,0)
    sim.setExplicitHandling(ikTask2,0)
end

dropObject=function(dropPos,dropOrientation,fingerAngle,relVel,detected_object)

sim.moveToPosition(target,Franka,{dropPos[1],dropPos[2],dropPos[3]+0.2},dropOrientation
,movementVelocity*relVel,movementAcceleration*relVel)

sim.moveToPosition(target,Franka,dropPos,dropOrientation,movementVelocity*relVel,move
mentAcceleration*relVel)
    data=sim.packInt32Table({0})..sim.packFloatTable({fingerAngle})
    sim.tubeWrite(gripperCommunicationTube,data)
    sim.wait(0.5)
    release(detected_object)
    sim.wait(1)
sim.moveToPosition(target,Franka,{dropPos[1],dropPos[2],dropPos[3]+0.2},dropOrientation
,movementVelocity*relVel,movementAcceleration*relVel)
end

getBoxAndDropIt=function(dropPos,dropOrientation,useIntermediateDropPos,fingerAngle,r
elVel)
   detected_object=sim.getIntegerSignal("detected_object")
   getObject(fingerAngle,relVel,detected_object)
   if (useIntermediateDropPos) then
      sim.switchThread() -- Make sure sim.handleIkGroup was already called in this pass
      moveToIntermediateDropPos(relVel)
   end
   dropObject(dropPos,dropOrientation,fingerAngle,relVel,detected_object)
end

take_for_classification=function(inInts,inFloats,inStrings,inBuffer)
   class_num=inInts[1]
   sim.setIntegerSignal("take_for_classification",class_num)
         return {},{},{},''
end

function sysCall_threadmain()
   -- Initialization:
   sim.setThreadSwitchTiming(2) -- Default timing for automatic thread switching
   Franka=sim.getObjectHandle('Franka')
   target=sim.getObjectHandle('Franka_target')
   tip=sim.getObjectHandle('Franka_tip')
   pick_pos=sim.getObjectHandle('Take_product_position')
   connector=sim.getObjectHandle("connector")
   bottle_drop_place=sim.getObjectHandle("Bottle_products_drop_place")
   fruit_drop_place=sim.getObjectHandle("Fruit_products_drop_place")

   joints={-1,-1,-1,-1,-1,-1,-1}
   for i=1,7,1 do
      joints[i]=sim.getObjectHandle('Franka_joint'..i)
   end
   ikTask1=sim.getIkGroupHandle('Franka_undamped')
   ikTask2=sim.getIkGroupHandle('Franka_damped')

gripperCommunicationTube=sim.tubeOpen(0,'BarrettGripperState'..sim.getNameSuffix(nil),
1)
   pickupOrientation={0,0,math.pi/2}
   upperPickupJointPos={math.pi/2,-0.247,-0.2042,0,-1.1193,0,-5.2042}
   intermediateDropPos={0,-0.2421,-0.4452,0,-0.8835,-1.5708,0}
   movementVelocity=0.6
   movementAcceleration=2
   jointVelocity=math.pi*0.6
   jointAcceleration=10

   data=sim.packInt32Table({0})..sim.packFloatTable({0})

   sim.tubeWrite(gripperCommunicationTube,data)

   record=false


   for i=1,100 do

      class_num=sim.waitForSignal("take_for_classification")


r,dist,pt,detectedObjectHandle=sim.readProximitySensor(sim.getObjectHandle('Product_stop
_for_pickup_sensor'))

      if (r>0 and class_num~=nil) then


          if class_num==0 then

                  drop_pos=sim.getObjectPosition(bottle_drop_place,Franka)

          elseif class_num==1 then

                  drop_pos=sim.getObjectPosition(fruit_drop_place,Franka)

          end


          pickupPos_offset=sim.getObjectPosition(pick_pos,Franka)

          pickupPos={pickupPos_offset[1],pickupPos_offset[2]+dist,pickupPos_offset[3]}

          getBoxAndDropIt(drop_pos,{0,0,math.pi/2},false,0,1)


      end

      result=sim.clearIntegerSignal("take_for_classification")
   end

end
```