Addressing Real-World Challenges Through AI and Robotics Solutions

1)Introduction

The aims of this assignment are to assess my knowledge about challenges to design and use solution based AI and robotics in real problems I have demonstrated a clear understanding of concepts about interactive autonomous systems and robots and their Applications in real-world scenarios.

I've decided to build a automatic sorting system within a manufacturing/ logistics environment, with the use of UR5e robot programmed using machine vision using camera detection to identify colours and objects

2) Literature review and motivation

The focus of my literature review is to analyse the difficulties associated with designing an AI robotic solution capable of using computer vision to identify and sort objects in a manufacturing environment. I am motivated by the challenges inherent in developing such applications to address real-world problems and the potential impact of creating systems that tackle challenges across various applications. Projects like mine could have a substantial significance in practical applications such as recycling centres, logistics/warehouses, and fulfilment centres.

A relevant study I came across is Li et al.'s (2020) investigation into an "Automatic Sorting System Using Machine Vision" within the food industry. This research explores various machine vision sensors suitable for sorting systems and diverse image processing algorithms for the identification and classification of objects.

Another notable paper, "Robot Autonomous Sorting System for Intelligent Logistics," outlines the design and execution of a robot system for autonomously sorting packages in a warehouse. The system employs binocular stereo vision for package identification

and integrates map building and path planning algorithms for navigation. Both simulated and real-world experiments were conducted to assess its effectiveness and accuracy.

In addition to academic sources, I incorporated insights from YouTube videos, acknowledging the inherent limitations of using them as primary sources. However one video demonstrated an individual implementing a robotic arm paired with an Arduino UNO to sort coloured objects based on their RGB values using an RGB colour sensor.

After delving into numerous studies and YouTube videos, I gained valuable insights into the complexities of developing my AI robotic automated sorting system. This exploration reinforced my determination to tackle real-world challenges, recognising the transformative potential of robotics and AI in diverse fields such as recycling centres, logistics/warehouses, and fulfilment centres.

Section 3: Implementation Details

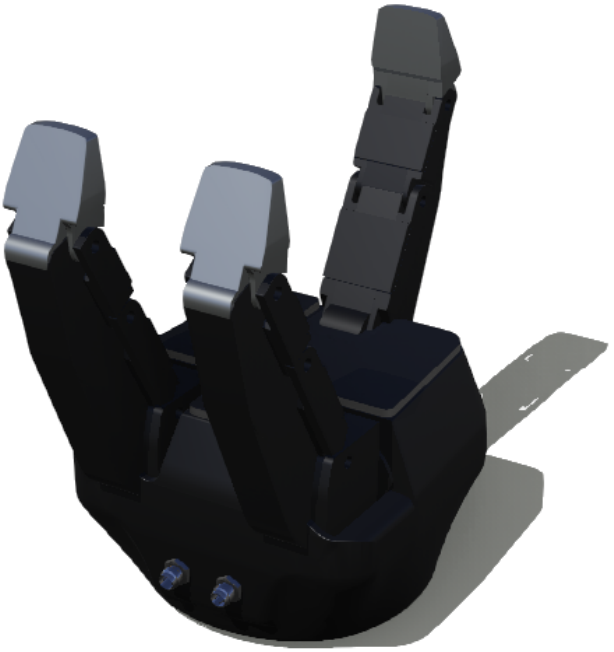
3.1 Robot Selection

UR5e Robot

I chose to use the UR5e robot from Universal Robots, the UR5e is a flexible robot and is able to be applied in various tasks this includes manufacturing tasks such as pick and place, assembly, machine tending and other automated tasks so this choice of robot aligned well with my goal for this assignment.

Figure UR5e

Figure Robotiq3fGripper



I imported the UR5e robot from the Webots environment from "WEBOTS_HOME /projects/robots/universal_robots/protos/UR5e.proto." The UR5e initially comes without a hand in Webots. However, in the UR5e Scene Tree Editor, there's a tool slot that allows the addition of components. In this slot, this is where I incorporated the Robotiq3fGripper and also added a camera and a distance sensor to the hand gripper. I specified and configured the robot and its components in my code to align with the goals of my project

UR5e Robot code

```
from controller import Robot, DistanceSensor, Motor, PositionSensor

# Define the time step for the simulation
TIME_STEP = 32

# Different states of the robot
class State:
    WAITING = 0
    GRASPING = 1
    ROTATING = 2
    RELEASING = 3
    ROTATING_BACK = 4

# Initialise the robot and set initial parameters
robot = Robot()
counter = 0
state = State.WAITING
speed = 2.0

# Define motors for hand and arm movements
hand_motors = [robot.getDevice("finger_1_joint_1"),
               robot.getDevice("finger_2_joint_1"),
               robot.getDevice("finger_middle_joint_1")]

ur_motors = [robot.getDevice("shoulder_lift_joint"),
             robot.getDevice("elbow_joint"),
             robot.getDevice("wrist_1_joint"),
             robot.getDevice("wrist_2_joint")]

# Set initial velocities for arm motors
for motor in ur_motors:
    motor.setVelocity(speed)

# Initialise distance sensor and enable it
distance_sensor = robot.getDevice("distance sensor")
distance_sensor.enable(TIME_STEP)

# Initialise position sensor for wrist joint and enable it
position_sensor = robot.getDevice("wrist_1_joint_sensor")
position_sensor.enable(TIME_STEP)
```

In this code segment , I import necessary modules from the Webots library. These modules facilitate the control of the robot, as well as manage distance sensors and motors. I establish different states for the robot within a class, signifying distinct stages in its movement during the sorting process. The hand and arm movements are governed by motors obtained through 'robot.getDevice', with each joint individually defined to correspond to specific parts of the robot. Each joint has its own dedicated motor, providing precise control over each individual joint allowing for a high degree of flexibility and customisation in the robot's behaviour. Furthermore, the code initialises

and enables distance sensors for the robot, and a parallel treatment is applied to the position sensor.

```
# Add camera initialisation
camera = robot.getDevice("camera")
camera.enable(2 * TIME_STEP)
camera.recognitionEnable(2 * TIME_STEP)
```

This segment of code initialises and enables the camera on the robot while also enabling camera recognition so the camera is able to process and recognise objects the (time_step) is used to determine how often camera data is updated.

```
# Define target positions for different colored objects
#[x,y,z,w] coordinates in a 3d environment
# coordiantes on where to drop specific objects
target_positions_red = [-1.88, -4.0, -2.38, -1.51]
target_positions_green = [-1.88, 2.14, -2.38, -1.51]
target_positions_blue = [-1.88, -1.14, -2.38, -1.51]
target_positions_foreign = [0, -1.14, -2.38, -1.51]
```

This code segment establishes target positions for various coloured objects that the robot is sorting in my Webots 3D environment. The coordinates (x, y, z, w) specify precise locations where the robot should place each object during the sorting process.

```
# Main control loop
while robot.step(TIME_STEP) != -1:
    # Counter logic to manage time delays between actions
    foreign_color = True #Any colour that is no green red or blue will be defined as
    #foreign colour

    # Camera recognition code
    if counter <= 0:
        if state == State.WAITING:
            # Transition to GRASPING state when an object is within a certain distance
            if distance_sensor.getValue() < 500:
                state = State.GRASPING
                counter = 8
                print("Grasping object")
                for motor in hand_motors:
                    motor.setPosition(0.85)

        elif state == State.GRASPING:
            # Camera colour recognition code
            colors = camera.getRecognitionObjects()# Retrieves the list of objects
            #reconised by the camera
            target_positions = None  # Initialise target_positions variable
            #outside the loop
```

This segment of the code establishes the main control loop, ensuring continuous robot movement until the simulation concludes (foreign_color = True). The variable “foreign_color” serves to identify any colour beyond green, red, or blue as a foreign object.

The code then checks if the robot is in the waiting state, a phase where the robot remains stationary on the conveyor belt, anticipating an object. The transition to the

grasping state occurs when an object within a distance less than 500 is detected. In this state, the hand motors are activated, adjusting to a position of 0.85 to initiate the

grasping action. Additionally, a console message, "Grasping object," is printed to the Webots console to display the robot's ongoing activity.

Following this, the code verifies if the robot is in the grasping state. During this phase, it utilises (“colors = camera.getRecognitionObjects() “) to retrieve a list of objects recognised by the camera. The initialisation of the target_positions variable tells the robot the specific target position the object must go to based on its recognised colour.

```
# Identify the colour of the detected object
for color in colors:
    color_rgb = color.getColors()#This gets the RGB values of the
    #current object.
    red, green, blue = color_rgb[0], color_rgb[1], color_rgb[2]
    # Set target positions based on detected colour
    # This section uses RGB values to determine the colour of the objects
    #If the conditions are not met it assumes its a foreign object
    # based on the detected colour, it sets the target poition variable to
    #the corresponding  list of positions
    if red > 0.8 and green < 0.2 and blue < 0.2:  # Red
        print("Detected Red object")
        target_positions = target_positions_red
    elif red < 0.2 and green < 0.2 and blue > 0.8:  # Blue
        print("Detected Blue object")
        target_positions = target_positions_blue
    elif red < 0.2 and green > 0.8 and blue < 0.2:  # Green
        print("Detected Green object")
        target_positions = target_positions_green
    elif foreign_color:  # Set foreign_color to True for any other color
        print("Detected foreign object")
        target_positions = target_positions_foreign
```

This code initiates by capturing the colours detected by the camera through the line “color_rgb = color.getColors()”. This involves calling the “getColors()” method on the colour objects, which returns a list of RGB values corresponding to the recognised colours.

Following this, the code proceeds to analyse these RGB values using specific thresholds, to determine the colour of the detected objects. When these conditions are met, the code generates console messages, identifying the coloured object, such as "Detected Blue object." Depending on the identified colour, the robot will use the predefined corresponding target position, e.g. characterised by coordinates “target_positions_blue”.

For each colour, predetermined target positions guide the robot to the designated location. If conditions are not met by recognised colours(green, red, blue) the system assumes the object is a foreign object meaning there has been contamination and the object does not belong on that specific conveyor belt. In such cases, the robot utilises “target_positions_foreign” to dispose of the foreign object, preventing potential

contamination on conveyor belts tailored for recognised colours.

This systematic approach ensures precise identification and routing of each colour to its designated location, while foreign objects are systematically addressed to uphold the sorting process's integrity.

```
            # Move the robot arm based on the detected colour
            if target_positions is not None:
                for i in range(4):
                    ur_motors[i].setPosition(target_positions[i])
                print("Rotating arm")
                state = State.ROTATING

        elif state == State.ROTATING:
            # Rotate the arm back after reaching a certain position
            if position_sensor.getValue() < -2.3:
                counter = 8
                print("Releasing object")
                state = State.RELEASING
                for motor in hand_motors:
                    motor.setPosition(motor.getMinPosition())

        elif state == State.RELEASING:
            # Release the object by resetting arm positions
            for motor in ur_motors:
                motor.setPosition(0.0)
            print("Rotating arm back")
            state = State.ROTATING_BACK

        elif state == State.ROTATING_BACK:
            # Transition back to WAITING state after arm rotation
            if position_sensor.getValue() > -0.1:
                state = State.WAITING
                print("Waiting for object")

    #Counter for managing time delays
    counter -= 1

# Cleanup after the main loop
robot.cleanup()
```

This segment of code is part of the control loop that orchestrates the robot's responses based on colour detection. The loop persists until the simulation is manually stopped. Afterwards, a cleanup process is initiated to ensure the simulated environment reverts to its initial state.

During the "Rotating" state, the robot determines the target position to place the object

using the value obtained from position_sensor.getValue(). Upon reaching a specific value, it shifts to the "Releasing" state, signifying the release of the object. A console comment, "Releasing object," is then printed. After the object is placed in its specific target position the hand motors reset to their initial positions using “motor.setPosition(0.0)”. The robot proceeds to rotate back to its original position on the conveyor belt, transitioning back to the "Waiting" state, where it anticipates the arrival of a new object on the conveyor belt.

Supervisor robot

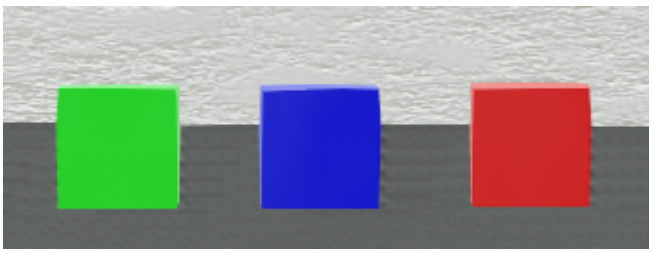
```
from controller import Supervisor
import random

# Create the supervisor
supervisor = Supervisor()

# List of available shapes in Webots
available_shapes = ["Cube", "Cube2", "Cube3", "Cube4", "Cube5", "Cube6","Cube7", "Cube8", "Cube9", "Cube10", "Cube11", "Cube12"]

# Define specific positions for each cube
cube_positions = {
    "Cube": [6.83, -0.82, 0.96],
    "Cube1": [6.83, -0.82, 0.96],
    "Cube2": [6.83, -0.82, 0.96],
    "Cube3": [6.83, -0.82, 0.96],
    "Cube4": [6.83, -0.82, 0.96],
    "Cube5": [6.83, -0.82, 0.96],
    "Cube6": [6.83, -0.82, 0.96],
    "Cube7": [6.83, -0.82, 0.96],
    "Cube8": [6.83, -0.82, 0.96],
    "Cube9": [6.83, -0.82, 0.96],
    "Cube10": [6.83, -0.82, 0.96],
    "Cube11": [6.83, -0.82, 0.96],
    "Cube12": [6.83, -0.82, 0.96],
}

# Loop and spawn a new random cube every 2 seconds
while True:

    # Wait for 2 seconds
    supervisor.step(time_step=8000)

    # Spawn a new random cube
    random_shape = random.choice(available_shapes)
    obj = supervisor.getFromDef(random_shape)

    if obj is not None:

        # Set position for the cube
        if random_shape in cube_positions:
            obj.getField("translation").setSFVec3f(cube_positions[random_shape])
            print(f"Object {random_shape} spawned successfully at {cube_positions[random_shape]}.")

        else:
            print(f"No specific position defined for {random_shape}. Using random position.")
            obj.getField("translation").setSFVec3f([0, 0, 0])  # Default to (0, 0, 0)

while supervisor.step(32) != -1:
    # Your simulation logic goes here
    pass
```

I developed a code for my Supervisor robot to simulate a manufacturing environment using Webots. The code generates custom shapes and places them on a conveyor belt. The program maintains a list of available shapes that were customly made in my Webots environment, it would randomly select cubes from its list of available to spawn on the conveyor belt. The UR5e robot is programmed to identify 'green,' 'red,' and 'blue' cubes.

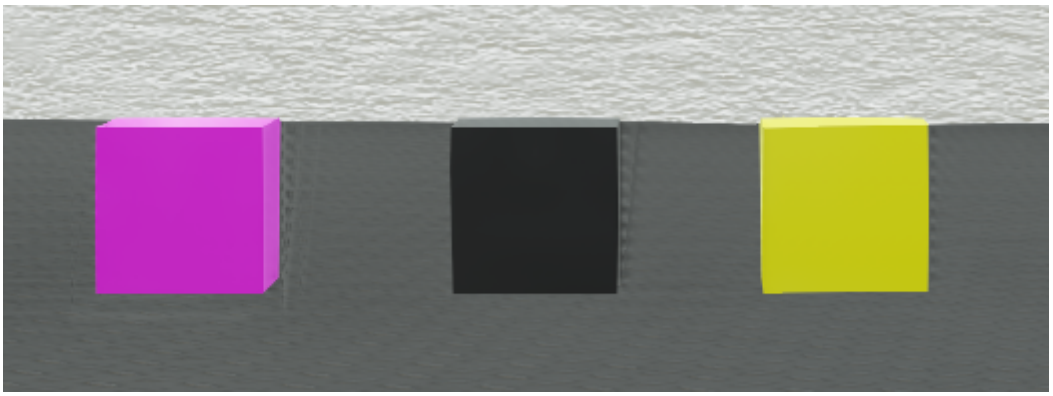
In order to assess the robot's performance in a real-world scenario, I introduced cubes of colours that the robot isn't programmed to recognise. I coded the robot to classify any object outside the designated colours as foreign, simulating contamination on the conveyor belt. This setup enables me to observe how the robot reacts when confronted with unexpected cubes during its operation on the conveyor belt.

3.2 Environment Setup

Recognised objects

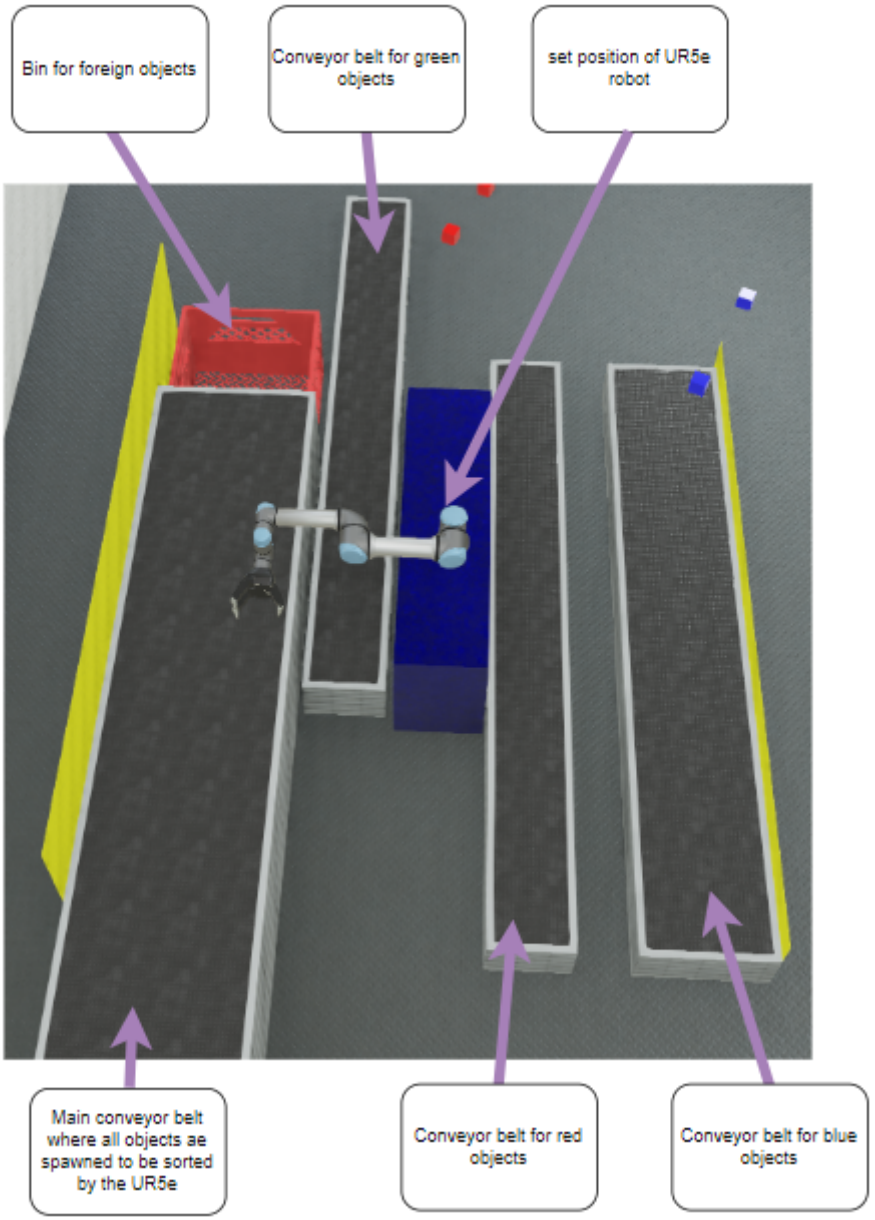


Foreign objects



I made custom cubes using the Webots environment by adding solids and making customs shapes using shape. The custom cubes are characterised by their height, width, and length parameters. To simulate real-world conditions, the cubes are enabled with physics, by assigning them with a specific mass of ‘-1’. Additionally, each cube is equipped with a bounding box for camera recognition, and Recognition colours are assigned for accurate identification. The only difference between the recognised objects and foreign objects are the colours.

Test environment



This is my testing environment, set up to simulate a real manufacturing setting. The labelled conveyor belts are imported from the Webots environment at "WEBOTS_HOME/projects/objects/factory/conveyor_belt/protos/ConveyorBelt.proto," and have been customised in size and speed using the Webots Scene Tree Editor to suit the specifics of my custom environment. Similarly, the UR5e robot and the crate, serving as a bin for foreign objects, are also imported from the Webots environment at "WEBOTS_HOME/projects/robots/universal_robots/protos/UR5e.proto" and "WEBOTS_HOME/projects/objects/factory/containers/protos/PlasticCrate.proto,"

respectively. The yellow walls, similar to the custom cubes mentioned earlier, are custom made and act as a preventative barrier to prevent cubes from falling off as they move along the conveyor belt after being sorted.

4) Advantages and disadvantages

4.1 Advantages

Choosing the UR5e robot for my project offered inherent flexibility, making it adaptable to a range of manufacturing tasks, including the specific requirements of my project. The ability to meticulously control each individual motor joint provided me with precise control over the robot's movements, granting a high degree of flexibility and customisable behaviour. This feature was paramount in my project, facilitating the accurate and reliable sorting of objects that simulated a real manufacturing task environment.

Additionally, the integration of sensors, cameras, and distance detection enhanced the robot's interaction and environmental awareness, a critical aspect for my project. Improved environmental awareness enabled the robot to interact more effectively with its surroundings. With enhanced capabilities, the robot could track objects on the conveyor belt and recognise different-coloured items using RGB values. This capability proved crucial in ensuring the robot processed and sorted colours accurately, contributing to the overall success of the project.

The inclusion of distance sensors added an additional layer of precision to my robot. These sensors were instrumental in determining when the gripper should pick up an object, ensuring accurate and precise grasping when the object was in close proximity. This functionality significantly reduced the likelihood of the gripper missing objects during the picking process.

Furthermore, the incorporation of the Robotiq3fGripper further broadened the robot's grasping capabilities, enabling precise handling of objects. The ability to manipulate each finger joint provided a high level of accuracy in picking up items within my simulated environment. The Robotiq3fGripper is also versatile as it can seamlessly adapt to different projects with minimal code adjustments, making it a valuable asset that can effectively adapt to specific environmental requirements.

4.2 Disadvantages

An inherent drawback in my project is the overreliance on colour recognition by the robot's cameras. This dependency becomes a concern in real-life scenarios where objects might share similar colours or experience variations in lighting conditions, leading to potential misclassifications. The risk of sorting items into incorrect areas arises, presenting a notable limitation that isn't evident in the simulation but poses challenges for practical applications in the real world.

Furthermore, the restriction to a single gripper configuration proves to be a significant disadvantage. While it suffices for handling individual objects sequentially, it falls short in manufacturing environments demanding the simultaneous management of multiple items. This limitation hampers the robot's efficiency in scenarios that require concurrent handling, highlighting the need for a more advanced gripper system to enhance versatility throughout.

Another challenge lies in the system's reliance on predetermined coordinates for sorting both coloured and foreign objects which introduces a need for manual adjustments to these positions when used in a different environment. This dependence hinders the robot's inherent adaptability to diverse manufacturing scenarios, making it less initially versatile and responsive to dynamic changes in its environment.

Section 5: Use of the robot in the real world

5.1 Real - world Applications

The robotic system I've developed has immense potential across various practical applications. While I've demonstrated its effectiveness in manufacturing through simulations, its versatility extends to sectors such as logistics, warehouses, and recycling centres. In logistics, the robot excels at sorting and categorising products, optimising supply chain efficiency. In recycling centres, it automates material sorting, enhancing the recycling process and promoting sustainability. Additionally, the system proves valuable in E-commerce fulfilment centres, streamlining product sorting and organisation for faster order processing.

5.2 New Implications

The implementation of automation in tasks offers a substantial boost to operational efficiency, diminishing the need for manual labour and accelerating processes. The robot's capacity to operate continuously, except for scheduled maintenance or external

factors like power outages, ensures a 24/7 working capability. In the long run, integrating robots into warehouse operations stands as a cost-saving measure for businesses, eliminating labour expenses and generating quicker returns on investment compared to manual labour.

While the integration of robots may result in job displacements in specific areas, it presents an opportunity for the emergence of new roles. This shift focuses on responsibilities such as robot maintenance, programming, and supervision. Ongoing support from engineers and robotics experts becomes paramount to sustain the system's functionality throughout the year, ensuring its continuous and correct operation.

5.3 Ethical implications

AI automation has the potential to transform industries, but in doing so could potentially lead to displacement of human employees. With the integration of Artificial Intelligence in various industries, the skill requirements of jobs is changing. ``According to a report from the McKinsey Global Institute, up to 375 million workers may need to change their occupations or acquire new skills by 2030 due to automation and AI. Employers and governments may need to invest in re-skilling and training programs to help workers acquire new skills that are in demand in an AI-driven world.”

The impact on AI in the job market is becoming increasingly significant. With the growing adoption of AI, there are some jobs that are most likely to be replaced by machines. One of The industries that are most vulnerable to job loss due to AI is the manufacturing industry. Automation of manufacturing processes has been on the rise for several years and robots are increasingly being used to replace human workers. Lower skilled workers are more vulnerable to job displacement because of automation, where they work in jobs that involve routine and repetitive tasks, for example, manual labour and administrative jobs. “According to a report from the World Economic Forum, AI and robotics will lead to a net loss of over 5 million jobs in 15 major developed and emerging economies by 2020. In the United States, a study by the Brookings Institution estimated that 25% of jobs are at high risk of automation, while another 36% are at medium risks”.

7) Conclusions

In summary, my creation of an automated robotic sorting system has effectively tackled real-life challenges by leveraging AI and Robotics to address a practical problem. The success of my project is evident in seamlessly integrating the UR5e robot into a sorting system, achieving precise control over its movements, and enhancing task flexibility through the strategic use of sensors, cameras, and distance detection. The choice of the UR5e robot proved advantageous due to its inherent flexibility, making it easily adaptable to various manufacturing tasks.

However, a notable limitation in my project lies in the overreliance on colour recognition by the robot's cameras. While effective in simulated environments, real-life conditions, such as varying light levels causing glare, may lead to colour recognition errors. Another constraint is observed in the Robotiq3fGripper, which, while sufficient for handling rindividual objects sequentially, falls short in environments requiring the simultaneous management of multiple items.

Despite these challenges, my project holds significant potential for various real-world applications. For instance, it could revolutionise recycling centres by automating the sorting of recyclable materials, enhancing efficiency, and promoting sustainable practices. Nevertheless, the development of such systems raises ethical concerns, particularly in the context of increasing AI automation that could potentially displace human workers, especially in repetitive labour tasks. In the ever-evolving landscape of AI and robotics, it is crucial to embrace technological advancements responsibly and address ethical implications. Despite these challenges, the practical applications of my project in manufacturing, logistics, recycling centres, and e-commerce fulfilment offer opportunities for efficiency improvements and sustainable practices.

8) References

Literature review

Sheth, S. M., Kher, R. K., Shah, R., & Jani. (2010). Automatic Sorting System Using Machine vision. *ResearchGate*. https://doi.org/10.13140/2.1.1432.1448

Song, M., & Xin, S. (2021, September 1). *Robot autonomous sorting system for intelligent logistics*. IEEE Xplore. https://doi.org/10.1109/CEI52496.2021.9574514

Color Sorting Pick and Place Robotic Arm using Arduino UNO, TCS34725. (n.d.). Www.youtube.com. Retrieved December 14, 2023, from https://youtu.be/a_Ib23hJXIc?si=RzP8-DMyrNjM910q

Implementation details

Webots: robot simulator. (2019). Cyberbotics.com. https://cyberbotics.com/

Ethics

What jobs are more likely to replaced by Artificial intelligence
McKinsey Global Institute. (2017). Jobs lost, jobs gained: What the future of work will mean for jobs, skills, and wages.
https://www.mckinsey.com/featured-insights/future-of-work/jobs-lost-jobs-gained-what-the-future-of-work-will-mean-for-jobs-skills-and-wages

World Economic Forum. (2018). The future of jobs report.
https://www.weforum.org/reports/the-future-of-jobs-report-2018