AI Risk Mitigation for Industrial Robotics

Ahras Ali

200389400

ENSE 480

Project Report

April 12th, 2023

Table of Contents

Introduction

Automated industrial robotics are a marvel of engineering. They allow us to accomplish tasks at higher quality, accuracy, and consistency. Automated industrial robots are used for a variety of applications such as assembly lines, welding, painting, material handling, packaging and many more. These robots can lift upwards up to 500 kilograms allowing them to be efficient and versatile in heavy duty manufacturing lines. Often these robots are in collaborative work environments with humans making them an essential part of current industrial settings. With the increasing adoption of automation in industrial settings, it's imperative to ensure that these systems operate safe and efficiently.

Problem

There are many risk factors to automated industrial robotics. A major concern is their lack of self-awareness of their environments. Many people have been injured and some have lost their lives. Wendy Holbrook was killed in a Michigan car plant due to a malfunction in the robots embedded architecture. Furthermore, Robert Williams was slammed into a wall by a robotic arm leading to his death. These incidents could have easily been avoided if a scene understanding application was implemented within the robot's embedded system.

Solution

I developed a scaled model of an automated industrial robot with the addition of artificial intelligence. The goal of my project is to provide a proof of concept of a working robot that has features of risk mitigation using artificial intelligence. In essence, this project is based around human safety in industrial collaborative work environments.

To achieve this goal, I implemented a miniaturized robotic arm that can move and detect if any humans are in near proximity. I implemented 5 techniques; scene understanding, risk identification, risk evaluation, risk mitigation and robot navigation to enforce risk mitigation and elimination. (***Source:*** *A. Hata, R. Inam, K. Raizer, S. Wang and E. Cao, "**AI-based Safety Analysis for Collaborative Mobile Robots**")*

Knowledge and data representation

To develop an integrated solution, I'm using a wide variety of software applications to perform object detection as well as control hardware. These include the NVIDIA jetpack, Nvidia's tensor RT, python, SSD-Mobile-net-v2, coco data set and the Keil uvision development platform.

Nvidia's Jetpack is a software development operating system designed for fast deployment of artificial intelligence and machine learning on the jetson nano. The reason I chose jetpack as the operating system is because it comes with a variety of GPU accelerated computing tools such as CUDA, CUDDNN tensor RT and many more. I specifically implemented tensor RT which can take trained neural networks and optimize them to be more efficient on the Nvidia GPU's by applying optimization techniques like auto turning, layer fusion, precision calibration, and dynamic tensor memory management.

I used Python as the platform for writing the object detection algorithm. Using Python allowed me to take full advantage of SSD-Mobile-net-v2 model and objects detection through a live image feed. The reason I chose SSD-Mobile-net-v2 is because it is designed for light computing devices and real-time object detection. To train the object detection algorithm, I used the Coco data set which consists of 330,000 images and over 90 different classes. Having a large data set allows for an accurate detection of humans in real time as seen in Figure 1: Human detection.

Figure 1: Human Detection

Lastly, I decided to use Keil tools by arm as the programming platform for the STM 32F103RB. The reason I chose Keil uvision is because it has a debugging feature which is primitive when working back from an error. The debugging feature allows me to step through each line of code and see the outcome on the robot to determine the exact point of error.

Approach and technique

The approach I used to achieve a self-aware robot was to develop a real time algorithm that can be deployed while the robot is in motion. Real time object detection is a computer vision technique that involved detecting objects through a live image feed. These images are then fed through SSD-Mobile-net-v2's deep learning model that was trained on the Coco data. This process is commonly known as a Convolutional Neural network (CNN).

The Technique I implemented achieve my project goals were to use jetpack as the operating system for the jetson nano which how's all the artificial intelligence algorithms and computer vision processing. I used Python to write the object detection algorithm. The algorithm uses SSD-Mobile-net-v2 which is the trained model for the object detection algorithm. I trained the SSD-Mobile-net-v2 model on the Coco data set allowing it to detect objects within 90 plus different categories. Lastly, I implemented tensor RT which is deep learning inference library developed by NVIDIA to help optimize and accelerate the detection algorithm for the jetson nano. Once the process is complete, the algorithm in Python will determine if there are any humans in the frame and indicate that to the STM32F103RB microcontroller. In Figure 2: Real time Human detection, the Robotic arm is able to detect a human using the onboard came concurrently while trying to place a box on the platform.

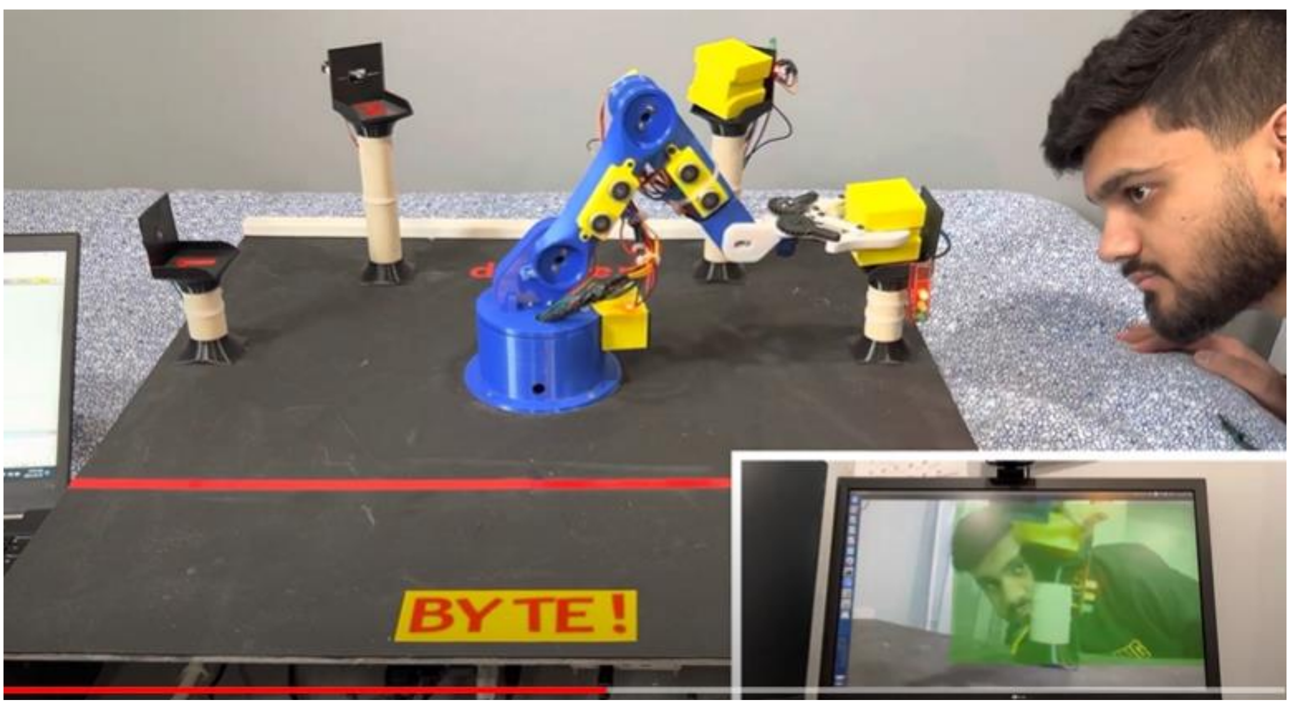


Figure 2: Real time Human detection

Furthermore, I used Hardware components such as ultrasonic sensors on each of the robot's arms facing both directions to detect if there is any human or objects within close proximity. The combination of the two systems working to together allows for a complete robotic awareness operating system.

Hardware

The hardware components used on my project consisted of 3D printing all the housing for the robotic arm, a jetson nano for the AI algorithm computations, the STM32F103RB microcontroller for all the hardware communications, servo motors for the movement of the robotic arms, ultrasonic sensors for close proximity range detection, buzzer and an RGB LED.

The 3D printed robotic arms were printed using a filament type called PLA. The design and the files were provided by a youtuber named "howtomechatronics" (Figure 3). These files consisted of base the arms the gears and the grippers. I had to design the box, the platforms and redesign the gears to allow a smoother movement of the robotic arm.

Figure 3: Computer Model of the Robotic Arm

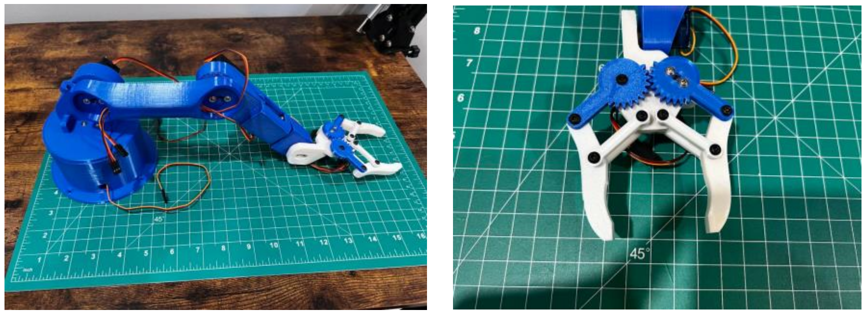


Figure 4: 3D Printed Robotic arm

STL Files provided by:

https://howtomechatronics.com/tutorials/arduino/diy-arduino-robot-arm-with-smartphone-control/#unlock

https://thangs.com/designer/m/3d-model/38899

For the real-time object detection computational device, I used a Nvidia jetson nano which is a single board computer used for embedded AI applications (Figure 5). The jetson nano consists of a quad core and arm cortex A51 CPU and a 128-bit Nvidia Maxwell GPU. The jetson nano also has 4 gigabytes of DDR4 ram and a variety of general-purpose Input and output pins. The reason I chose a jetson nano over the popular Raspberry Pi 4 is because of its Nvidia Maxwell GPU. As I am running a real time object detection using computer vision, it will require a graphics processing unit which the Raspberry Pi does not have.

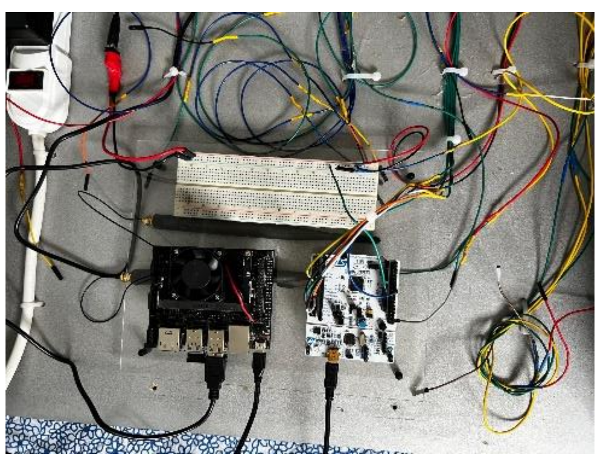


Figure 5: Jetson Nano (right), STM32F103RB (left)

I used an STM32F103RB as the main microcontroller for all the hardware components for the project. The STM32F103RB is a Development Board with a 32-bit arm cortex M3 processor and 128 kilobytes of flash memory. It allows for a wide range of applications and consist of many peripherals such as GPIO, analog to digital converters and pulse width modulation which was used to control their servos.

Furthermore, are used HC SR04 ultrasonic sensors to detect close proximity objects near the robotic arm. The age the ultrasonic sensors work by sending a high frequency sound wave which bounces off objects and returns back to the centre the sensor that measures the time it takes for the sound wave to travel to the object and back and then calculates the distance of the object. Lastly, I have used the buzzer to alert the user (Figure6) within the close proximity that they are in a direct Line of movement with the robotic arm.

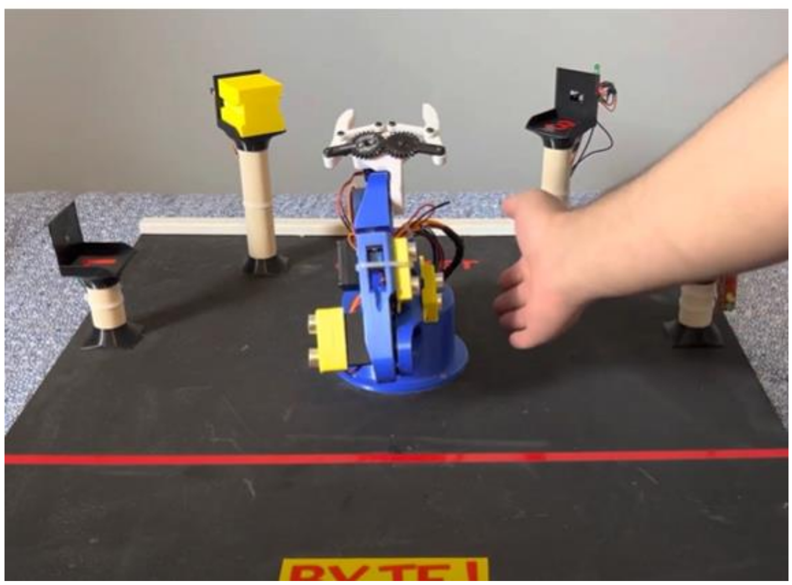


Figure 6: Close Range object detection

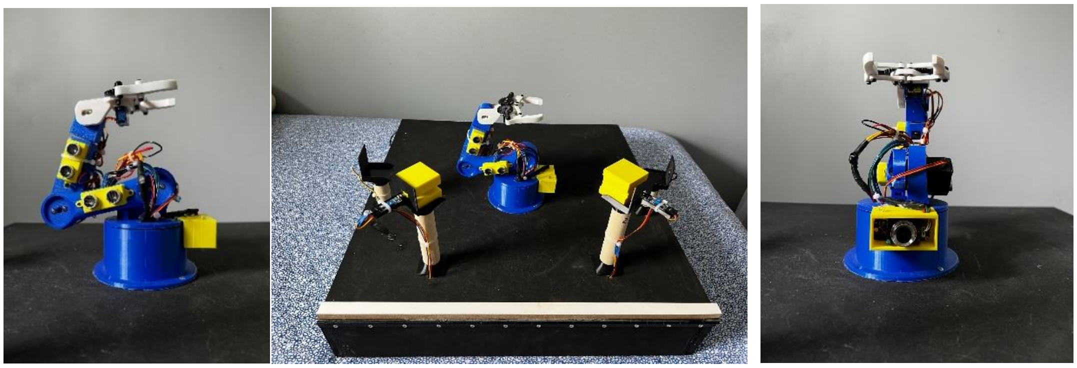


Figure 7: Final Robotic design

Structural diagram

The structure diagram below consists of the two microcontrollers used in the project. The green is the jetson nano which is used for object detection and the blue is the STM32F103RB used for controlling the hardware within the project. (Please refer to the How-to Modules for a complete break down of the process.)

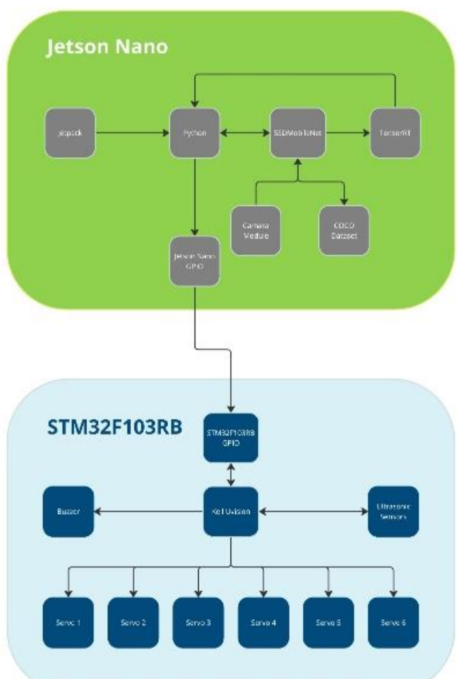
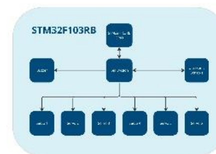
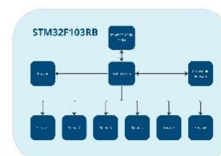
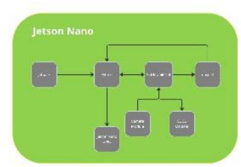
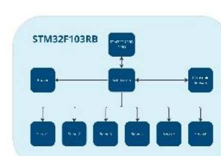
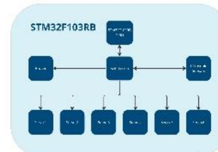
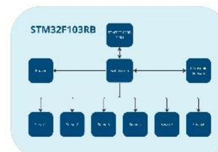
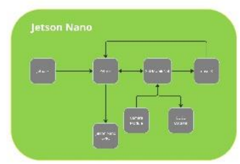
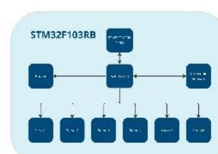
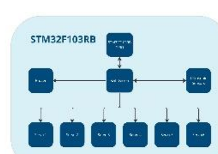


Figure 8: Structural Diagram

Manual/ Modules

Regular operations	
The STM32F103RB will ask the ultrasonic sensor and the jetson nano if there are any humans in the area.	
The Ultrasonic sensors sends a sonic ware and checks is there is any object within 10 cm. If there is no object, the ultrasonic sensor responses with no. If there is an object within the search area, the ultrasonic sensor will respond with yes.	
The Jetson nano will use the camara and run the object detection algorithm to check if there is a human in the search area. If the algorithm picks up a human, the Jetson will respond with a yes by transmitting a high output through GPIO (general purpose Input output) pin 40. Otherwise, the GPIO Pin will respond with a low output.	
The STM32F103RB will read both the response is from the ultrasonic sensor and the jetson nano GPIO pins. Once the STM32F103RB has confirmed that both devices have responded with a no, the STM32F103RB will trigger the servo motors to move to the desired location.	

Human detection	
The STM32F103RB will ask the ultrasonic sensor and the jetson nano if there are any humans in the area.	
The Ultrasonic sensors sends a sonic ware and checks is there is any object with in 10 cm. If there is an object within the search area, the ultrasonic sensor will respond with yes.	
The Jetson nano will use the camara and run the object detection algorithm to check if there is a human in the search area. If the algorithm picks up a human, the Jetson will respond with a yes by transmitting a high output through GPIO (general purpose Input output) pin 40.	
The STM32F103RB will read both the response is from the ultrasonic sensor and the jetson nano GPIO pin. Since both the devices have responded with a yes, the STM32F103RB will sound the alarm and stop all operations of the robotic arm. The STM will now wait a specific period of time to ensure them human has moved out of the area of operations before continuing.	
Once the alarm has completed, the STM32F103RB will move the robot arm to the desired location.	

Sample code

```python
#################
#
# Author Ahras Ali
# Date April 12, 2023
# ENSE 480 Project
#
# Referenced https://www.youtube.com/watch?v=bcM5AQSAzUY&t=1380s
#
#################

#importing Nvidia's Deep learning inferencing models
import jetson.inference

#importing Nvidia's multimedia module for capturing and processing images
import jetson.utils

#importing Python pacakge to intract with the General purpose Input/output pins
import Jetson.GPIO as GPIO

# set up GPIO pin numbering mode
GPIO.setmode(GPIO.BOARD)

# Setting up GPIO pin 40 and 38 as an outputs
GPIO.setup(40, GPIO.OUT)
GPIO.setup(38, GPIO.OUT)

#######
#
# The net variable initiates the SSS-Mobilenet-v2 model and sets the minimum confedence level to 70%
# The Camara variable is used to capture the live video and the resolution
# The Display variable is used to display the camara feed after the object detection has run
#
#######
net = jetson.inference.detectNet("ssd-mobilenet-v2", threshold=0.7)
camera = jetson.utils.gstCamera(1920, 1080, "/dev/video0")
display = jetson.utils.glDisplay()

#######
#
# This section of code is used to run the Object detection algorithm continuously until the esc key is hit on the keyboard
# If a person is detected at anytime during the frame, GPIO pin 4o is set high and alerts the STM32F103RB that a person is detected
#
#######
while display.IsOpen():
    img, width, height = camera.CaptureRGBA()
    detections = net.Detect(img, width, height)

    person_detected = False
    for detection in detections:
        if detection.ClassID == 1:  # 1 is the ID for person class
            person_detected = True
            break

    if person_detected:
        GPIO.output(40, GPIO.HIGH)
        GPIO.output(38, GPIO.LOW)

    else:
        GPIO.output(40, GPIO.LOW)
        GPIO.output(38, GPIO.HIGH)


    display.RenderOnce(img, width, height)


GPIO.output(40, GPIO.LOW)
GPIO.output(38, GPIO.LOW)
# clean up GPIO resources
GPIO.cleanup()
```

Figure 9: Python Algorithm

```
//#########
//
// In this Section of code, everytime the move_base function is called, it checks the three Ultrasonic
// sensors and then the jetson nano camara output.
//
// Once the checks are complete, the Servo location is sent to TIMER 4 on Channal 1.
// That output is then proceeded by Timer 4 and executed
//
//#########

void move_base(uint16_t test)
{
  Check_ultral();
  Check_ultra2();
  Check_ultra2();
  Check_Cam();
  TIM4->CCR1 = converter(test);
  TIM4 -> EGR |= TIM_EGR_UG;
  sleep_ms(time);


}
```

Figure 10: Robot Verification checks

Future development

Future developments consist of integrating close proximity sensors and touch sensors to enhance the robotic awareness of these industrial arms. A PhD study in China found that touch sensors can be implemented on the side of the robotic arms. These sensors can be activated when the robot has collided with an object or a human.

Secondly, further iterations of the object detection algorithm can be implemented on the robotic arm and the environment of the robotic arm itself. This would conclude adding cameras to corners of robotic arm workstations for a field of view of the whole environment. This integration will allow for the algorithm to detect humans that are in safe zones, close proximity and danger zones.

Lastly, the embedded system operating code can be upgraded with a real-time operating system to handle faster response times for the ultrasonic and the jetson nano verifications steps. This suggestion was provided by Dave Duguid, from the electronic systems engineering department at the University of Regina.

Conclusion

In conclusion, automated industrial robotic arms provide a wide range of benefits to society ranging from mass production to efficiency and quality to economic benefit. These robots are essential to the growing economy of Canada and the world. As more robots are immersed into a warehouse and collaborative workspaces, AI robotic awareness is essential to protecting human life. As a robot developed a sense of their environment, they will be able to better understand and predict outcomes when faced with hazardous case scenario. Integrating artificial intelligence and human detection is a steppingstone into developing a human safe robot for a collaborative work environment.

Disclaimer

I would like to make a note that the base for this project will also be used for another class, ENEL 351: Microcontroller Systems. To ensure that there is no overlap between the projects, the core component projects are different. For ENSE 480, I have used ultrasonic sensors which are digital as well as a Nvidia jetson nano for real time object detection. ENEL 351 does not allow Jetson Nano as the main microcontroller and has different operational goals. If there are any concerns of overlap, please feel free to contact me or my professor, Robert Martens.

References

https://www.youtube.com/watch?v=bcM5AQSAzUY&t=1381s

https://developer.nvidia.com/blog/realtime-object-detection-in-10-lines-of-python-on-jetson-nano/

https://www.youtube.com/watch?v=tFNJGim3FXw

https://x2robotics.ca/tower-pro-180-degree-metal-gear-servo-mg996r

https://developer.nvidia.com/tensorrt#:~:text=TensorRT%2C%20built%20on%20the%20NVIDIA,and%20others%20on%20NVIDIA%20GPUs.

https://cocodataset.org/#home

https://www.therichest.com/technologies/15-shocking-deaths-caused-by-robots/

https://www.thingiverse.com/thing:1838120

A. Hata, R. Inam, K. Raizer, S. Wang and E. Cao, "AI-based Safety Analysis for Collaborative Mobile Robots," 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, Spain, 2019, pp. 1722-1729, doi: 10.1109/ETFA.2019.8869263.