ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION

Paper ID #45309

Designing an AI-Enhanced Module for Robotics Education in Mechanical Engineering Technology

Dr. Wenhai Li, Farmingdale State College

Assistant Professor in Department of Mechanical Engineering Technology, Farmingdale State College, Farmingdale, NY 11735

Dr. Yue Hung, Farmingdale State College

Dr. Yue (Jeff) Hung holds degrees in engineering and technology disciplines (Ph.D. in Materials Science and Engineering, M.S in Mechanical Engineering, and B.S in Manufacturing Engineering Technology). He has over 20 yearsâ€™ experience in Computer-Aided

Mr. Reiss Guttman, Farmingdale State College

Research Assistant

Sen Zhang, State University of New York, Oneonta
Ning Yu, State University of New York, Brockport

Designing an AI-Enhanced Module for Robotics Education in Mechanical Engineering Technology

Wenhai Li, Jeff Hung, Reiss Guttman
Department of Mechanical Engineering Technology, Farmingdale State College, Farmingdale, NY 11735

Sen Zhang
Department of Mathematics, Computer Science and Statistics, State University of New York College at Oneonta, Oneonta, NY 13820

Yu Ning
Department of Computer Science, State University of New York at Brockport, Brockport, NY 14420

Abstract

The rapid integration of Artificial Intelligence (AI) across industries necessitates the inclusion of AI-related education within engineering programs. However, AI education remains limited for Mechanical Engineering Technology (MET) students, who often lack strong backgrounds in math and programming. To address this gap, we design a five-week AI-enhanced module integrated within an existing robotics course at SUNY Farmingdale. This module introduces foundational AI and Machine Learning (ML) concepts, with a focus on practical applications in robotics and automation. Instead of diving deep into complex mathematics or programming, the module emphasizes the use of accessible AI tools, such as computer vision (CV), natural language processing (NLP), and reinforcement learning (RL), to solve engineering problems. By embedding the module into the robotics curriculum, students gain hands-on experience applying AI to real-world challenges, preparing them for AI-driven roles in engineering. This paper details the design and implementation of the module, explores the challenges of adapting AI education for MET students, and discusses strategies to bridge gaps in their technical knowledge. We also outline an assessment plan for evaluating student learning outcomes and provide insights into future improvements to meet evolving industry demands.

Introduction

Artificial Intelligence (AI) is increasingly becoming a vital tool across numerous industries. In engineering, particularly in fields like robotics and automation, AI is revolutionizing problem-solving approaches. As the demand for AI-driven solutions continues to grow, it's essential for Mechanical Engineering Technology (MET) students to gain a solid understanding of AI and Machine Learning (ML). With AI already playing crucial roles in areas such as automation and predictive maintenance, integrating AI education into MET programs is key to preparing students for the evolving challenges of their careers. Integrating AI/ML into the MET curriculum presents several unique challenges. These challenges arise from both the structure of the current MET program and the specific learning needs of MET students.

Firstly, the existing MET catalog is already packed with essential courses, making it difficult to introduce new AI/ML courses. Teaching the full spectrum of AI/ML, from theory to coding, typically requires

multiple courses. However, with limited space in the program, adding one or more dedicated AI/ML courses is a significant challenge. This would require a complete curriculum overhaul, which may not be feasible given the current structure.

Secondly, most MET students do not have a strong foundation in mathematics and programming, both of which are essential for understanding the theory behind AI/ML and for coding AI applications. AI/ML relies heavily on concepts from statistics and linear algebra, and applying these concepts requires proficiency in Python, a key programming language in the AI field. This knowledge gap makes it difficult to design AI/ML courses that MET students can easily engage with and succeed in.

Finally, the AI/ML skills that MET students need are different from those required by Computer Science students. While Computer Science students focus on developing advanced models and diving deep into AI/ML theory, MET students need practical skills that enable them to use existing AI/ML tools to solve real-world engineering problems. For most engineering applications, simple AI models are often sufficient, and the ability to effectively apply these models is more critical than developing new ones. As a result, traditional AI/ML courses designed for Computer Science students may not be appropriate for MET students, who should be trained as AI practitioners rather than AI model developers.

To address the challenges of teaching AI/ML to MET students, we've designed a five-week AI/ML module that integrates directly into an existing robotics class. This approach allows us to bypass the need for new courses in an already packed curriculum while providing practical exposure to AI within a relevant field.

By embedding the AI/ML module into the robotics class, we leverage a subject that's already familiar to MET students. Robotics and automation are naturally suited to learning AI/ML since many of the concepts and tools overlap with AI/ML applications. This integration allows us to enhance students' understanding of robotic systems while introducing them to AI/ML in a manageable way, without overwhelming them with entirely new material.

The module is centered around applied, hands-on experience. Rather than focusing solely on lectures, students will engage in a series of lab projects that directly apply AI/ML to real-world problems. These projects will help students gain practical experience with AI tools and techniques, such as using computer vision (CV) for robotic control or applying reinforcement learning (RL) to make the robot more intelligent. This hands-on approach helps students grasp AI/ML concepts in a more relatable way, giving them the skills needed for their future careers in engineering.

At the same time, the module provides students with a foundational understanding of AI/ML concepts. The lectures are designed to introduce the most fundamental theories and commonly used models, such as linear regression, classification, and neural networks, within the context of robotics and automation. By focusing on practical applications rather than abstract theory, we reduce the barriers MET students might face in grasping AI/ML and make it more relevant to their future work as engineers.

In the following sections, we will first provide an overview of the existing content in the Robotics class and how the new AI-enhanced modules have been integrated. We will then detail the structure of the five-week module. The lecture topics for each week will be explained, with a focus on how the material is tailored to be easily understandable for MET students. Similarly, we will describe the design of the weekly lab projects, which provide hands-on experience to reinforce theoretical concepts. Lastly, we will outline our assessment plan, designed to evaluate the effectiveness of the module and measure student learning outcomes.

Contents of the Module

A) Current Robotics Curriculum

The current robotics course at the MET department at Farmingdale State College is a senior-level course spanning 15 weeks, with both lecture and lab sessions. The lectures introduce students to the fundamental concepts of industrial robotic arms, covering their hardware components and control theories, including forward and inverse kinematics, dynamics, and more. The course equips students with the technical skills necessary to control robotic arms, preparing them for careers as robotics engineers. Lab sessions are structured around hands-on projects that allow students to operate robotic arms for tasks such as basic pick-and-place operations, measuring repeatability, integrating sensors, peripheral controls, path planning, and advanced programming techniques. These lab projects provide students with practical experience in automation design, ensuring they acquire the skills needed for industry applications.

The current course structure can be condensed into 10 weeks by concentrating on essential robotics control theories, such as forward and inverse kinematics, while omitting more advanced topics like robot dynamics, force and torque control, and path planning, which can be reserved for graduate-level instruction. In terms of lab work, the department has upgraded its equipment, replacing the Intelitek SCORBOT-ER 4u educational robots [1] with industrial-grade Universal Robots' UR3 models [2]. Unlike the SCORBOT, which required manually controlling the robot's pose using buttons on its software interface, the UR3 robot features a "free-drive" mode that allows students to adjust the robot's pose by hand, significantly speeding up the process. This reduces the time-consuming task of positioning the robot for position recording, enabling students to complete lab tasks more efficiently. As a result, the original 15-week lab curriculum can be completed in 10 weeks, creating room for the proposed 5-week AI-enhanced module.

B) Lectures of the Module

The AI-enhanced module is designed to span five weeks, with each week focusing on fundamental AI/ML topics that are directly applicable to robotics and automation. The lectures are structured to provide students with an introduction to key concepts in AI/ML, while keeping the content accessible to MET students, who often lack extensive backgrounds in mathematics or programming. The topics covered each week include:

- Week 1: Introduction to Machine Learning (ML) – This lecture introduces the core concepts of ML, focusing on applications in robotics and automation. The key topics include data-driven decision-making, pattern recognition, and how ML systems learn from data, make predictions, and improve over time. Key ML terms such as training data, features, models, and algorithms are introduced, alongside an explanation of model evaluation metrics like accuracy and precision, which are essential in assessing model performance.

- Week 2: Classification & Regression – This lecture focuses on supervised learning, explaining how classification and regression models make predictions based on input data. Simple models like the perceptron (for classification) and linear regression (for regression tasks) are introduced. The session includes an introduction to loss functions and advanced model evaluation metrics like confusion matrices, recall, and F1-score. Students learn how these models are trained using labeled data and understand how overfitting/underfitting can affect classification/regression accuracy.

- Week 3: Deep Learning and Neural Networks – This lecture introduces deep learning concepts, focusing on neural networks. Students learn the basics of how neural networks operate, with a focus

on understanding the input, hidden, and output layers. While advanced neural network architectures are briefly mentioned, the emphasis is on simple networks and how they recognize patterns. Students are introduced to the role of activation functions, which add non-linearity to models, making them more powerful for complex tasks like image recognition. This lecture simplifies backpropagation, focusing on its role in adjusting weights within neural networks.

- Week 4: Computer Vision (CV) & Natural Language Processing (NLP) – In this lecture, students explore two critical areas in AI: CV and NLP. For CV, the focus is on how convolutional neural networks (CNNs) are used for feature extraction and object detection. In robotics, these concepts are applied to tasks such as visual tracking or automation based on visual inputs. The NLP portion covers how machines interpret and generate human language, with a focus on voice-controlled robotics. Students are introduced to the basics of tokenization and language models, along with Recurrent Neural Networks (RNNs), which help machines process sequential data, such as sentences, and respond to natural language commands. RNNs are particularly emphasized for tasks involving sequential dependencies in speech recognition and language translation, making them a key tool in voice-controlled robotics.

- Week 5: Reinforcement Learning (RL) – The final lecture covers RL, focusing on how agents learn to maximize rewards through interaction with their environment. The lecture explains key RL concepts such as agents, states, actions, rewards, and policies. Students learn how reinforcement learning can be applied to robotics, where robots learn to optimize performance through trial and error. Concepts such as Q-learning and exploration vs. exploitation are briefly introduced, with practical examples tied to robotic systems.

Since MET students generally have less experience with programming and advanced mathematical concepts, the lectures are designed to simplify complex theories and focus on practical applications. Rather than delving into the theoretical depth of AI/ML models, we provide high-level overviews and focus on how these models apply to real-world robotics and automation tasks. For instance, when discussing neural networks, we highlight their ability to recognize patterns rather than the detailed mathematics of backpropagation.

The lectures incorporate visual learning techniques, such as case studies and practical examples in robotics, to help students connect AI concepts directly to their field of study. For example, when discussing classification and regression, we demonstrate how these models can predict object sizes based on sensor data. Additionally, programming requirements are kept to a minimum, using simplified code snippets or pre-built models that students can run with minimal modifications. Tools like Google Teachable Machine [3] and open-source computer vision libraries (OpenCV [4]) are used to give students hands-on experience without overwhelming them with coding.

We also encourage students to utilize large language model tools like ChatGPT [5] to help them understand code, draft code, and troubleshoot errors. By focusing on practical tools rather than building models from scratch, we enable MET students to quickly become familiar with basic AI/ML concepts in a relatively short time, equipping them to apply these tools effectively in real-world engineering problems.

C) Lab Projects of the Module

The lab component of the AI-enhanced robotics module is designed to provide students with hands-on experience in applying AI/ML concepts to real-world robotics applications. The lab projects center around the use of industrial-grade robotic equipment, including Universal Robots' UR3 robotic arms, with Robotiq's two-finger gripper [6] as the end effector. These robotic arms are equipped with two vision sensors: the Pixy2 camera [7] and the Logitech C920 camera [8], both installed at the robot arm's wrist. The Pixy2 camera offers built-in color detection, making it an efficient and cost-effective tool for students to practice basic object detection based on color. The Logitech C920 camera, paired with the OpenCV package, enables more advanced object detection tasks. Additionally, its built-in microphone can serve as an audio input device for voice-controlled projects. Figure 1 shows the hardware setup, including robotic arms and sensors.

Each week's lab project consists of two components: a demo/teaching section and an assignment. In the demo/teaching section, students are introduced to the essential skills, tools, and code required to complete the corresponding project. Once students are familiar with these tools and techniques, they are given an assignment that challenges them to complete a specific task. These assignments are designed to reinforce the week's lecture content, giving students practical experience in using AI/ML to solve real-world engineering problems. The details of each week's lab project are given below:

- Week 1: Control Robotic Arms with Python

  In this week's lab, students are introduced to Python-based control of the UR3 robotic arm. Since some students may lack programming experience, the lab starts with a demonstration of the basic hardware setup and how to establish a remote connection between the UR3 robot and a computer via Ethernet. Students learn to enable the remote-control feature on the UR3 and set up their development environment.

  The lab focuses on using the *ur_rtde* Python package [9], developed by SDU Robotics, which allows for programmatic control of the UR3 arm. The demo will explain key concepts such as controlling joint variables, defining the robot's pose (position and orientation), and using motion commands (*movej* for joint-based motion and *movel* for linear motion). Additionally, students will learn how to utilize kinematic functions like *getForwardKinematics* and *getInverseKinematics* to calculate positions and joint values for the robotic arm.

  Students are provided with sample Python code that demonstrates how to control the UR3's movements and operate the attached Robotiq gripper. The code will serve as a foundation for Lab Assignment #1, where students must modify the code provided to perform a simple pick-and-place operation. This assignment helps familiarize students with both Python programming and robot control, which will be critical for future AI/ML-integrated labs. A sample of the handout for Week 1 Lab Assignment is shown in Figure 2.

- Week 2: Automation using Computer Vision (CV)

  This lab session introduces students to the use of CV for a basic automated sorting operation. The Pixy2 camera, known for its built-in color detection capabilities, is selected to simplify programming while still allowing students to understand the fundamental principles of CV.

  The lab begins with a step-by-step demonstration of the Pixy2 setup. Students will learn how to teach the Pixy2 to detect specific colors using the manufacturer's PixyMon software [10]. Once configured, the Pixy2 can detect colored blocks and output their pixel coordinates in the image frame. Students will access these coordinates using the Pixy2 Python API [11].

The core of this lab focuses on the Affine transformation theory [12], which is used to map 2D pixel coordinates from the camera's image to real-world coordinates in the robot's workspace. A simple 2D calibration process will be demonstrated, guiding students on how to find the mapping matrix between the image and the robot's base coordinates. This transformation allows the UR3 robotic arm to identify and manipulate objects based on their location in the real world.

Sample Python code will be provided to students, showing how to obtain pixel coordinates from Pixy2, apply the transformation matrix, and instruct the UR3 robot to automatically move to the calculated position for a pick-and-place task. In the live demonstration, students will see how the robot uses the Pixy2 camera to autonomously detect the position of a color block and execute the pick-and-place operation.

For Lab Assignment #2, students will be required to modify the provided code to detect multiple-colored blocks and sort them into separate baskets. This task challenges students to extend the code by enabling the robot to differentiate between blocks of various colors, compute their positions, and perform the necessary sorting. Through this assignment, students gain hands-on experience with CV and coordinate transformations in robotic automation. A sample of the handout for Week 2 Lab Assignment is shown in Figure 3.

- Week 3: Automation using Self-Trained Model

In this week's lab, students will explore more advanced CV techniques, incorporating ML for object classification. While the previous lab focused on color-based sorting with Pixy2, this project leverages the Logitech C920 webcam and the OpenCV package to perform more complex classification tasks based on features like shape, orientation, and alphanumeric characters.

Students will be introduced to Google Teachable Machine, a no-code platform that allows users to train models for image classification by simply uploading labeled images. This approach eliminates the need for detailed coding and model tuning, making it highly accessible for MET students.

The lab begins with an introduction to OpenCV, covering the fundamentals of using the Logitech C920 webcam to capture images and detect objects. The sample code demonstrates how to use OpenCV to control the webcam and extract relevant features from captured images. Unlike Pixy2, which is limited to color detection, OpenCV offers the capability to detect object shape, size, and other attributes, allowing students to perform more sophisticated image analysis.

Next, the students will train their own ML models using Google Teachable Machine. For this project, they will create a model to classify blocks with different letters (e.g., A, B, C) written on top. Once trained, students will integrate the model into their Python code to classify objects in real-time, using OpenCV to capture images and the trained model to identify the letters.

The demo will show how the C920 webcam captures images, which are then processed by the trained model to classify the detected object. The entire process, from image capture to classification, will be demonstrated live. Students will receive the demo code for both image capture and classification.

For Lab Assignment #3, students will be tasked with extending the demo to create a system capable of sorting wooden blocks based on the letter detected (e.g., A, B, C). The system will detect the letter, classify the object, and instruct the UR3 robotic arm to place each block into the corresponding basket based on the classification. This lab gives students practical experience with

ML-guided automation design, allowing them to train models and use them to solve real-world classification tasks. A sample of the handout for Week 3 Lab Assignment is shown in Figure 4.

- Week 4: Develop Voice-Controlled Robot Arm through Natural Language Processing (NLP)

  This week's lab introduces students to voice control for the UR3 robotic arm, enhanced by NLP to interpret more complex voice commands. We utilize the Logitech C920 webcam's built-in microphone for audio input and Python's *SpeechRecognition* package [13] to convert spoken commands into text. This package allows students to leverage various popular Speech-to-Text (STT) services, such as Google, Microsoft, or IBM, to handle the conversion.

  The demo begins by showing how to use the *SpeechRecognition* package to record voice input and convert it into text using the Google STT service. Students are provided with sample code to handle basic voice commands. In this demo, simple voice commands, such as "left," "right," "up," and "down," are used to control the UR3 arm, with the robot responding by moving a predefined distance in the specified direction.

  Once the basic voice control functionality is demonstrated, the limitations of such a system are discussed. Namely, it relies on predefined keywords and cannot handle more complex commands, such as specifying both movement direction and distance ("Move left by 5 cm"). We then introduce NLP as a solution to overcome this limitation. By sending the transcribed text to OpenAI's ChatGPT, the system can interpret more detailed and natural language commands. The demo will show how ChatGPT processes the command "Move left by 5 cm" to extract both the direction and the distance parameters and then relays this information back to the robot for execution. The students are provided with sample code to connect ChatGPT through OpenAI's API [14] and shown how to design prompts that extract direction and distance from natural language commands.

  For Lab Assignment #4, students are tasked with building a voice-controlled system for the UR3 robotic arm that can understand and execute both simple and complex commands. This involves combining speech recognition and NLP to create a more intelligent voice interface for controlling the robot. By the end of this project, students will have gained practical experience in voice recognition and NLP, seeing how these AI technologies can be applied to automate robotic systems. A sample of the handout for Week 4 Lab Assignment is shown in Figure 5.

- Week 5: Reinforcement Learning (RL) for Robotic Arms

  In this final lab, students are introduced to RL through a Python simulation that teaches a robotic arm how to reach a target. The RL algorithm used is the Deep Deterministic Policy Gradient (DDPG) [15], which is suited for continuous control tasks. The goal is to show students how RL can be applied to teach robots through trial-and-error interactions with their environment.

  The lab starts with a detailed explanation of the Python simulation, where a two-joint robotic arm learns to reach randomly placed target points. Initially, the arm moves randomly, but each time it moves closer to the target, it receives a reward proportional to the proximity. Over time, as the arm completes multiple learning episodes (epochs), it optimizes its movement strategies and becomes more efficient in reaching the target. A live visualization of this learning process allows students to observe how the robot progressively improves through RL.

  The code used in this lab is adapted from a version shared by MorvanZhou on GitHub [16], which has been modified to use TensorFlow [17] and TF-Agents' [18] implementation of DDPG. While

the underlying algorithms are complex, the focus in this lab is on helping students understand how RL works conceptually without delving too deeply into the advanced math and code.

For Lab Assignment #5, rather than asking students to modify the complex RL code, they will focus on observing and documenting the learning process. Students will run the RL simulation multiple times, tracking metrics such as reward functions, the time taken to reach the target, and the number of successful attempts. They will be required to plot these results and document how the robot's performance improves over time. This assignment allows students to gain a practical understanding of how RL can improve robotic behavior without overwhelming them with the complexities of RL algorithm development. A sample of the handout for Week 5 Lab Assignment is shown in Figure 6.

These lab projects are challenging, and it is highly recommended that students work in groups to collaborate and share knowledge. Given the complexity of the tasks, group work will help students tackle problems more effectively and foster teamwork. It is also expected that students will encounter various coding issues throughout the projects. Strong classroom support from the instructor and teaching assistants is crucial to help students navigate these difficulties and ensure successful project completion. Additionally, students are encouraged to leverage LLM tools like ChatGPT to assist in understanding Python code and troubleshooting any errors that arise, further enhancing their problem-solving capabilities.

Assessment Plan

The assessment plan for the AI-enhanced robotics module evaluates students' understanding of AI/ML concepts and their application through hands-on lab work. The assessment components are listed below:

- Lab Assignments (60%) – Lab assignments are the most significant part of the assessment plan, focusing on real-world applications of AI/ML concepts in robotics. Each lab assignment requires students to modify, implement, or observe code and systems related to AI/ML, such as CV, NLP, or RL. By completing these assignments, students will demonstrate their ability to apply AI/ML concepts to real-world engineering challenges in robotics.

- Weekly Quizzes (20%) – Weekly quizzes assess the students' grasp of the theoretical content covered during lectures. These quizzes ensure that students are internalizing AI/ML concepts, such as supervised learning, neural networks, and reinforcement learning, before applying them in lab projects. The quizzes will test students on key AI/ML concepts and their ability to apply them to engineering problems in robotics.

- Final Report (20%) – The final report requires students to reflect on their overall learning experience in the module, focusing on the AI/ML concepts learned and how they were applied in lab projects. Instead of repeating lab details, students will summarize key concepts like CV, NLP, and RL, and discuss their relevance to real-world engineering challenges. They will reflect on the technical and conceptual challenges faced and how they overcame them, while also suggesting improvements to the AI/ML solutions they developed. Finally, students will describe their personal learning outcomes and propose future applications of AI in engineering.

The assessment plan above aligns with several of ABET's Student Outcomes (SOs) as follows:

- SO1: An ability to apply knowledge, techniques, skills, and modern tools of mathematics, science, engineering, and technology to solve broadly-defined engineering problems appropriate to the

discipline. This is demonstrated through lab assignments, where students apply AI/ML principles and modern AI tools, such as CV and RL, to solve real-world robotics challenges.

- SO3: An ability to apply written, oral, and graphical communication in broadly-defined technical and non-technical environments, and an ability to identify and use appropriate technical literature. The final project report supports this outcome by requiring students to clearly communicate their methodologies, results, and conclusions in a professional format, demonstrating their technical communication skills.

- SO5: An ability to function effectively as a member or leader on technical teams. Group work is encouraged in lab assignments, fostering teamwork as students collaborate to solve coding challenges and execute complex robotics projects.

The learning outcomes evaluated through the assessment plan not only measure the students' understanding and application of AI/ML concepts but also serve as critical feedback for assessing the overall effectiveness of the AI-enhanced robotics module. By analyzing student performance across lab assignments, quizzes, and the final report, we can identify areas for improvement in both the course content and delivery methods. The feedback will help refine the module for future improvements, ensuring it stays aligned with student needs and industry demands. The assessment plan is currently being implemented, and the learning outcomes will be evaluated post-implementation. The results of this evaluation will be presented at a future ASEE conference to share insights and potential improvements.

Conclusion

Robotics and automation provide a natural and ideal platform for MET students to learn and apply AI/ ML concepts. The integration of AI into robotics aligns with the practical, hands-on learning style suited to MET students, making it an excellent way to meet both the students' desire to learn AI and the growing industrial demand for AI-driven solutions.

The AI-enhanced module is specifically designed to cater to MET students, who often lack strong backgrounds in mathematics and programming. The module addresses this by introducing only the essential, foundational ML knowledge while emphasizing practical applications over theory. In addition, the lab projects focus on applying existing AI tools, such as CV, RL and NLP, to solve real-world automation problems. By shifting the focus from complex model development and training to problem-solving using AI tools, students are better equipped to succeed and understand how AI can be applied in engineering contexts.

This approach has the potential to be easily scaled up for other courses or disciplines, making it adaptable beyond robotics. The module could also serve as the foundation for developing a full course in AI/ML applications for MET students or for students in other engineering fields. Ongoing assessment of learning outcomes, combined with student feedback, will be critical for refining the module and ensuring it continues to meet both educational and industry needs, making it adaptable for future advancements in the field.

Teaching Material Sharing

To support educators interested in implementing similar AI-enhanced modules for robotics, we are happy to share the teaching materials developed for this course. These include lecture slides, lab project handouts, and sample code. To request access to the materials, please send an email to the corresponding author with the subject line "AI-Enhanced Module for Robotics."

Acknowledgment

This work was supported by the SUNY Innovative Instruction Technology Grant (IITG). We thank SUNY for their continued support in advancing educational innovations, which made the development of this AI-enhanced robotics module possible.

References

1 SCORBOT ER-4u Educational Robot (https://intelitek.com/scorbot-er-4u-educational-robot/)
2 Universal Robots UR3e (https://intelitek.com/scorbot-er-4u-educational-robot/)
3 Google Teachable Machine (https://teachablemachine.withgoogle.com/)
4 OpenCV (https://opencv.org/)
5 ChatGPT (https://chatgpt.com/)
6 Robotiq Hand-E Gripper (https://robotiq.com/products/adaptive-grippers#Hand-E)
7 Pixy2 (https://pixycam.com/pixy2/)
8 Logitech C920 (https://www.logitech.com/en-eu/shop/p/c920-pro-hd-webcam.960-001055)
9 *ur_rtde* - Universal Robots RTDE Interface (https://sdurobotics.gitlab.io/ur_rtde/)
10 PixyMon (https://pixycam.com/downloads-pixy2/)
11 Pixy2 Python API (https://docs.pixycam.com/wiki/doku.php?id=wiki:v2:building_libpixyusb_as_a_python_module_on_linux)
12 Affine Transformation (https://en.wikipedia.org/wiki/Affine_transformation)
13 SpeechRecognition Package (https://github.com/Uberi/speech_recognition)
14 OpenAI API (https://platform.openai.com/)
15 Deep Deterministic Policy Gradient Algorithm (https://spinningup.openai.com/en/latest/algorithms/ddpg.html)
16 MorvanZhou/train-robot-arm-from-scratch (https://github.com/MorvanZhou/train-robot-arm-from-scratch)
17 TensorFlow (https://www.tensorflow.org/)
18 TF-Agents (https://github.com/tensorflow/agents)

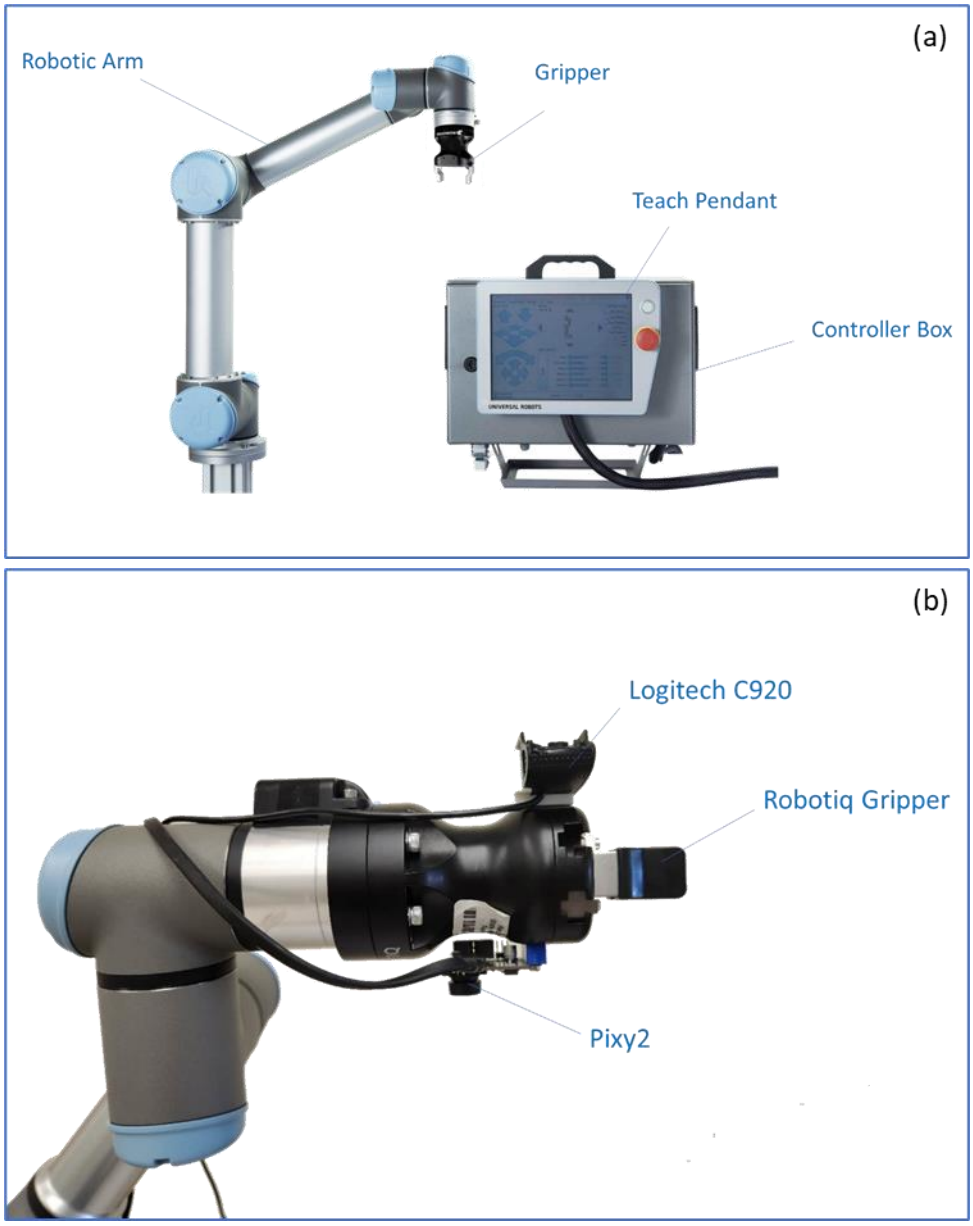
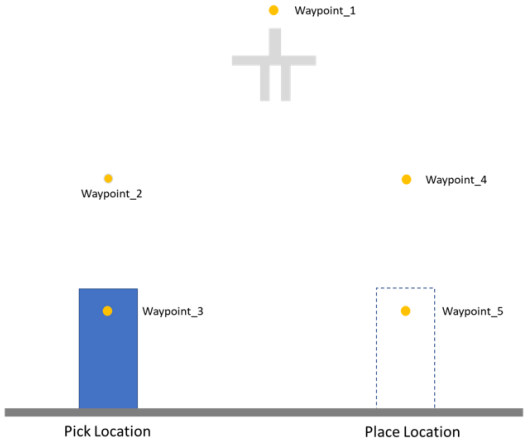


Fig. 1. Lab hardware setup: (a) UR3 robot; (b) Sensors installed.

Lab Assignment 01 - Controlling a Robotic Arm with Python

Problem Description

In this lab assignment, students will learn how to control a Universal Robots robotic arm using Python. The objective is to introduce basic concepts of robotic control through the *ur_rtde* Python package. Students will remotely control the UR3 robot, understanding how to manipulate its joint positions and execute basic movements. This lab is essential for developing the foundational programming skills needed for future AI/ML-enhanced labs.



Project Task

Follow these steps to complete the assignment:

- Establish a remote connection between the UR3 robot and the computer using an Ethernet cable. Enable the remote control function on the UR3 robot through its software interface.
- Import the *ur_rtde* package in Python and run the provided sample code to familiarize yourself with controlling the robot's movements and controlling the gripper.
- Modify the sample code to perform a basic pick-and-place operation. You will need to control the robot to pick up a block from one location and place and release it at another predefined position.
- Ensure the operation is smooth and successful, avoiding any collisions or errors.

Report Checklist

Compose a lab report including the following:

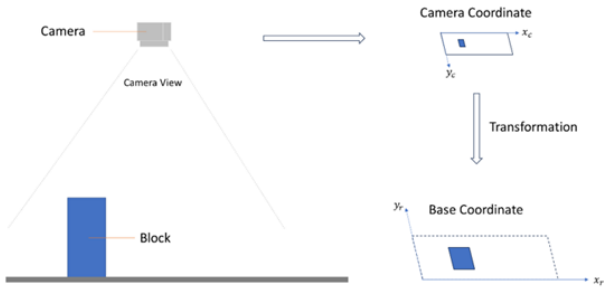
1. Project Overview (10 points): Provide a brief summary of the project and its objectives.
2. Robot Setup (10 points): Describe the steps taken to connect the UR3 robot to the computer and enable remote control. Include screenshots if necessary.
3. Code Explanation (20 points): Provide an explanation of the modified Python code for the pick-and-place operation. Highlight the sections where you implemented the joint variable control and the pick-and-place logic.
4. Observations and Results (30 points): Discuss your observations while running the robot, including whether the operation was successful and any challenges faced. Include pictures of the live robot operations to confirm that the tasks were completed successfully.
5. Appendix: Code (20 points): Include the full Python code used to complete the project, with comments explaining key sections.

Fig. 2. Lab assignment #1.

Lab Assignment 02 - Automation using Computer Vision

Problem Description

In this lab assignment, students will explore how to use computer vision (CV) to automate robotic tasks. Using the Pixy2 camera with built-in color detection capabilities, students will guide a Universal Robots UR3 robotic arm to perform an automated sorting task. By detecting blocks of different colors, the robot will be programmed to pick and place these blocks in designated locations. The task involves using a calibration process to transform the camera's pixel coordinates into the robot's base coordinates, enabling accurate robot control.



Project Task

Follow these steps to complete the assignment:

- Connect the Pixy2 camera to the computer and configure it using the PixyMon software to recognize distinct color signatures.
- Import the Pixy2 Python API and run the provided sample code to familiarize yourself with capturing pixel coordinates of the detected color blocks.
- To convert the camera's pixel coordinates into the robot's base coordinates, a calibration process must be completed. Place three blocks in an L-shape on the table and record the corresponding camera coordinates and robot coordinates for each block. Use the sample code provided to calculate the transformation matrix.
- Place a color block at a random position on the table, ensuring the same camera setup as during calibration. Use the computed transformation to calculate the block's position in the robot's base coordinates. Compare the computed value with the actual block position to verify if the transformation is correct. If the position is inaccurate, recheck the calibration process.
- Modify the provided Python code to control the UR3 robotic arm based on the detected color blocks' positions. The robot should detect each block's color, pick it up, and place it in a corresponding predefined location based on color. Ensure the sorting process runs smoothly and the robot correctly sorts each block.

Report Checklist

Compose a lab report including the following:

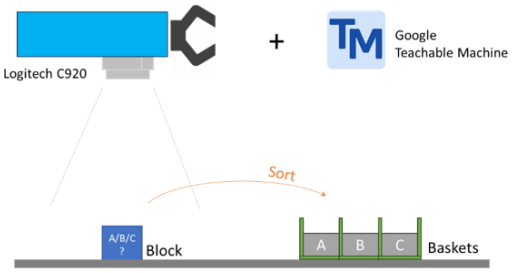
1. Project Overview (10 points): Provide a brief summary of the project and its objectives.
2. Pixy2 Camera Setup (10 points): Describe how you configured the Pixy2 camera and trained it to detect different color blocks. Include screenshots or images showing the color detection process.
3. Coordinate Transformation Process (20 points): Explain the calibration process and how the transformation matrix was derived. Include the camera and robot coordinates of the three blocks. Provide evidence (e.g., screenshots or photos) showing how you verified the transformation by comparing the computed robot base coordinates with the actual block position.
4. Code Explanation (20 points): Provide an explanation of the modified Python code that transforms pixel coordinates into robot workspace coordinates and controls the robot to perform the sorting task based on color detection.
5. Observations and Results (30 points): Discuss your observations during the automated sorting operation. Did the robot correctly sort the colored blocks? What challenges did you encounter? Include pictures of the robot performing the sorting task to confirm the operation was successful.
6. Appendix: Code (10 points): Include the full Python code used to complete the project, with comments explaining key sections.

Fig. 3. Lab assignment #2.

Lab Assignment 03 - Automation Using a Self-Trained Machine Learning Model

Problem Description

In this lab assignment, students will explore how to use machine learning (ML) for automating robotic tasks involving object classification. While previous labs focused on color-based sorting, this lab introduces a more advanced approach using the Logitech C920 webcam and OpenCV for capturing images and Google Teachable Machine for training an image classification model. The goal is to create a system where the robot can sort blocks based on the letter (e.g., A, B, C) on top of each block using the trained ML model.



Project Task

Follow these steps to complete the assignment:

- Connect the Logitech C920 webcam to the computer and use OpenCV (sample code provided) to capture real-time images of blocks on the table.
- Use Google Teachable Machine to train a simple image classification model. Train the model to recognize different letters (A, B, C). After training, download the model files and integrate them into the Python code.
- Use the provided Python code to test the trained model. Place a block with a letter (A, B, or C) on top and run the code to check if the model can successfully classify the block based on the letter. Make sure the classification is correct before proceeding to the next step.
- Modify the provided Python code to use the trained model for real-time classification. When the webcam detects a block, it should use the model to classify the letter on top of the block and output the classification result (A, B, or C).
- After classifying the block, program the UR3 robot to move the block to a corresponding basket based on its classification (A, B, or C). Ensure that the block is placed in the correct basket according to its label.
- Test the full system to ensure smooth operation, and verify that the robot is sorting blocks accurately based on the ML model's classification.

Report Checklist

Compose a lab report including the following:

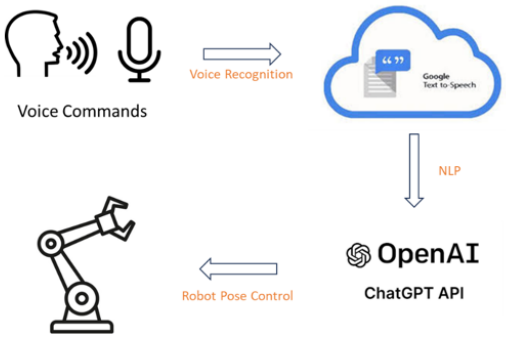
1. Project Overview (10 points): Provide a brief summary of the project and its objectives.
2. Webcam and OpenCV Setup (10 points): Describe the steps you took to configure the Logitech C920 webcam and capture images using OpenCV. Include screenshots or images of the captured images from the webcam feed.
3. Model Training (20 points): Explain how you used Google Teachable Machine to train the model. Include details such as the number of images used for training, the types of letters (A, B, C) detected, and the accuracy of the model during training.
4. Code Explanation (20 points): Provide an explanation of the modified Python code that integrates the trained ML model with the robotic system. Highlight the sections that use OpenCV to capture images and classify them using the model, as well as the logic controlling the UR3 robot's movement.
5. Observations and Results (30 points): Discuss the performance of the system. Did the robot correctly classify and sort the blocks based on the detected letters? Were there any challenges during classification or robot control? Include pictures showing the robot performing the sorting task to confirm the system's operation.
6. Appendix: Code (10 points): Include the full Python code used to complete the project, with comments explaining key sections.

Fig. 4. Lab assignment #3.

Lab Assignment 04 - Develop a Voice-Controlled Robot Arm using Natural Language Processing (NLP)

Problem Description

In this lab assignment, students will explore the integration of voice control with robotic systems using Natural Language Processing (NLP). By utilizing the Logitech C920 webcam's built-in microphone for audio input and Python's *SpeechRecognition* package, students will create a system that converts voice commands into robot movements. Additionally, using OpenAI's ChatGPT API, students will enhance the system to interpret more complex voice commands and execute tasks with the UR3 robotic arm.



Project Task

Follow these steps to complete the assignment:

- Connect the Logitech C920 webcam to the computer and configure its built-in microphone as the audio input device.
- Install and set up Python's SpeechRecognition package, and use the provided sample code to convert simple voice commands (e.g., "left," "right," "up," "down") into text. Test the system to ensure the voice commands are accurately transcribed.
- Use the transcribed text to control the robot's movements. Write a basic script that recognizes specific keywords from the voice input and moves the UR3 robot in the corresponding direction. For example, if the user says "left," the robot arm should move left.
- Design an appropriate prompt to send to OpenAI's ChatGPT API that can process more complex voice commands, such as "Move left by 5 cm." Use the provided sample code to connect to the ChatGPT API, and ensure the response from ChatGPT extracts both the direction and movement distance from the voice command. Ensure that the results are returned in a format that can be easily parsed and used by the robot control code.
- Modify the previous voice-controlled code (from step 3) so that the system can interpret both the direction and distance for robot movements using the processed response from ChatGPT. Test the code by giving commands like "Move right by 2 inches" or "Move forward by 10 cm," and ensure the robot executes both direction and distance accurately.
- Ensure smooth operation, and troubleshoot any issues with speech recognition, NLP, or robot control.

Report Checklist

Compose a lab report including the following:

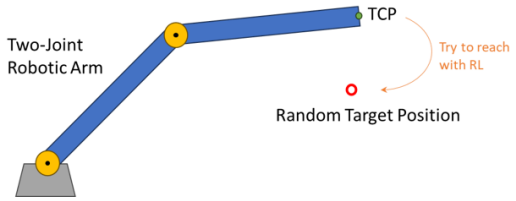
1. Project Overview (10 points): Provide a brief summary of the project and its objectives.
2. Voice Recognition Setup (10 points): Describe the steps you took to configure the microphone and SpeechRecognition package.
3. ChatGPT Prompt Design and API Connection (20 points): Explain how you designed the prompt to process both direction and distance from complex voice commands and how you used the provided sample code to connect to ChatGPT via the OpenAI API. Include the prompt you used and how ChatGPT processed the command into direction and distance values.
4. Code Explanation (20 points): Provide an explanation of the modified Python code that integrates speech recognition, NLP, and robot control. Highlight the sections that process voice commands, extract direction and distance from ChatGPT's response, and send movement instructions to the UR3 robot.
5. Observations and Results (30 points): Discuss the performance of the voice-controlled robot system. Were the commands executed correctly? Include pictures or video evidence of the robot responding to both simple and complex voice commands. If any challenges arose, describe how you resolved them.
6. Appendix: Code (10 points): Include the full Python code used to complete the project, with comments explaining key sections.

Fig. 5. Lab assignment #4.

Lab Assignment 05 - Reinforcement Learning (RL) for Robotic Arm Control

Problem Description

In this lab assignment, students will explore the concept of Reinforcement Learning (RL) and how it can be applied to optimize the movements of a robotic arm. The objective is to simulate the learning process of a 2D robotic arm in Python, using the Deep Deterministic Policy Gradient (DDPG) algorithm. Through this simulation, students will observe how the robotic arm learns to reach random target points through trial and error, improving its efficiency over time.



Project Task

Follow these steps to complete the assignment:

- Familiarize yourself with the provided Python code that simulates the reinforcement learning process for a 2D robotic arm. Review how the RL algorithm enables the robotic arm to optimize its movements to reach target points.
- Run the RL simulation to observe the initial random movements of the robotic arm as it attempts to reach a target point. Watch how, over time, the arm learns to perform more efficient and precise movements through the reinforcement learning process.
- Track key metrics, such as the reward function, time taken to reach the target, and the number of successful attempts during each training episode. Record your observations at different stages of the learning process (e.g., after 100, 200, and 300 episodes).
- Allow the simulation to complete multiple training episodes and collect data on the robotic arm's performance. Focus on understanding how the RL agent learns and improves through trial and error.
- Test the model's final performance after the training is complete. Document whether the robotic arm is consistently reaching the target and compare its performance to earlier stages of learning.

Report Checklist

Compose a lab report including the following:

1. Project Overview (10 points): Provide a brief summary of the project and its objectives.
2. Simulation Setup (10 points): Describe how you set up and ran the RL simulation for the robotic arm. Include screenshots or images of the simulation in action.
3. Code Explanation (20 points): Provide an explanation of the key sections of the provided Python code that implements the RL algorithm and controls the robotic arm's learning process.
4. Observations and Results (30 points): Discuss your observations during the simulation, focusing on how the robotic arm's performance improved over time. Include graphs or data that track key metrics such as the reward function, time to reach the target, and success rate over the course of training.
5. Conclusion (20 points): Summarize your conclusions about the effectiveness of the RL process. Was the robotic arm able to learn and improve its movements efficiently? Provide data or examples to support your conclusion.
6. Appendix: Code (10 points): Include the full Python code used to run the simulation, with comments explaining key sections of the code.

Fig. 6. Lab assignment #5.