7th European Conference on Industrial Engineering and Operations Management
Augsburg, Germany, July 16-18, 2024
Publisher: IEOM Society International, USA DOI: 10.46254/EU07.20240272
Published: July 16, 2024

Smart Robot Manipulation using GPT-4o Vision

Ryan Tan, Harry Yang and Sean Jiao
George Walton Comprehensive High School, Marietta, GA 30062, USA
ryan.rt.tan@gmail.com, harryyang07@outlook.com, seansljiao@gmail.com

Liuyang Shan, Chenxi Tao and Roger J. Jiao
School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA
lshan8@gatech.edu

Abstract

This paper explores the application of a robotic arm powered by artificial intelligence (AI) and deep learning to enhance the precision and efficiency of object manipulation tasks. Leveraging OpenAI's GPT-4o image recognition model along with OpenCV, we implement smart computer vision techniques to accurately recognize, locate, and manipulate targeted objects. By automating control and minimizing human intervention, our approach has the potential for improving traditional kinematics-based methods. The integration of AI enables robots to learn from vision data, adapt to various scenarios, and improve over time. Our findings demonstrate the feasibility of utilizing AI and deep learning to achieve high accuracy and efficiency in robot manipulation, with profound implications for manufacturing automation and beyond.

Keywords
Smart manufacturing, Robot manipulation, Large-language model, GPT vision, Machine learning

1. Introduction
Robot manipulation refers to the ability of a robot to interact with objects in its environment, typically through the use of a robotic arm or end-effector (gripper, tool, etc.). This involves a range of tasks such as grasping, moving, placing, and assembling objects. Robot manipulation is a crucial aspect of robotics because it enables robots to perform complex tasks that require precision, dexterity, and adaptability. Robot manipulation is a cornerstone of manufacturing automation, enabling the development of sophisticated and efficient automated systems that can perform a wide range of tasks with high precision and reliability (Anderson and Kurez, 2024).

With this problem in mind, we explore the application of a robotic arm powered by artificial intelligence (AI) and deep learning to recognize, locate, and manipulate targeted objects. The release of OpenAI's GPT-4 model has significantly advanced computer vision capabilities, allowing for seamless access through simple prompts. Its enhanced contextual and language understanding surpasses the limitations and rigidity of previous vision models. This benefit enables the GPT model to be utilized in a variety of scenarios, reducing the need for programs to be specifically tailored to a single use case (Liu et al., 2021).

This research is motivated by the need to improve efficiency and precision in robot manipulation. Manual calibration of robot control not only consumes valuable time but also introduces the possibility of human error. By automating these tasks, we aim to minimize errors, increase throughput, and equip robot manipulation with vision-based intelligence. The application of Large Language Models (LLMs) and deep learning neural networks in robot manipulation presents many advantages over traditional kinematics-based methods (Liu et al., 2024). AI enables robots to learn from vision data, adapt to new use cases, and improve their performance over time as more valid data is used for self-training throughout their lifespan. Deep learning, a subset of AI, allows for the development of

sophisticated algorithms that can accurately interpret and relate complex visual and numerical inputs. This capability is beneficial not only in manufacturing automation for its precision but also in various other scenarios due to its fast adaptability. By integrating AI, robotic arms can perform tasks with higher accuracy, adapt to environmental variations, and reduce the need for human intervention.

In this paper, we investigate the feasibility of a robotic arm system utilizing camera inputs. The camera leverages GPT-4's vision capabilities for identifying object state (orientation, degrees of tilt, etc.) and verifying the target object's hue while utilizing OpenCV for color-based identification and location tracking. The location data from the cameras is formatted to align with a coordinate system mapped onto the camera frame. This data serves as input for our deep learning algorithms, which determine the robotic arm's response and movement towards the target. By combining advanced object recognition and precise location tracking, our smart robot control approach aims to enhance the accuracy and efficiency of robotic manipulation.

2. Related Work on AI in Robot Manufacturing Automation

There has been substantial research and development in the application of AI for robot manufacturing automation. One active area is deep learning for object recognition and manipulation. Levine et al. (2018) discuss using deep learning to enable robotic arms to learn grasping through trial and error, leveraging large datasets for training. Goldberg et al. (2016) develop a framework called Dex-Net that uses deep learning to plan robust grasps by accessing a vast network of 3D object models stored in the cloud. Another research focus is reinforcement learning for adaptive control. For example, Levine et al. (2016) introduce an approach where deep reinforcement learning is used to train robots to perform complex manipulation tasks directly from raw pixel inputs.

Tremendous research work has devoted to collaborative robots and human-robot interaction. A cognitive robot contains problem-solving and reasoning procedures that allow it to make their own informed decisions. Cognitive intelligence enables robots with human-like adaptive abilities, encompassing decision-making and learning abilities fueled by the integration of AI and machine learning. These decisions serve as the basis for planners that correspond to execution of varying actions, as well as failures that are a result of these actions (Karapinar et al., 2012). Cognitive robots are equipped with mechanisms to handle such failures effectively; they can analyze the outcomes of their actions, identify the causes of any issues, and adjust their actions accordingly. In flexible manufacturing autonomous systems, human-robot collaboration allows a human operator to interact with a mobile robot (Li et al., 2024).

Learning high-level cognitive capabilities in robot manipulation tasks remains a critical area of research. Such tasks require robots to understand and execute complex language instructions involving perception, reasoning, and manipulation. Recent approaches leverage LLMs like ChatGPT and PALM to address these challenges. Open-loop models utilize LLMs for offline planning, while closed-loop models update plans based on feedback (Jin et. al., 2023). Incorporating visual observations can enhance control accuracy for complex tasks. It is of great potential to integrate a cognitive dataset and a multi-modal plan generation model to bridge task planning and control. Integrating multiple data modalities, such as textual, visual, and auditory inputs, enhances AI-driven analysis beyond traditional text-focused models (Wang et. al., 2024). This approach allows for more comprehensive task planning and environmental interaction, as demonstrated by projects like Inner Monologue and SayCan, which combine multimodal feedback to generate contextually aware and reliable plans. These advancements indicate a move towards AI systems that can dynamically and autonomously navigate and interact with the real world.

Recent developments on integration of AI models like GPT-4o and similar models in robotics allows for more intuitive interaction and flexible task planning. These models can understand and generate human-like language, which can be used to program and control robots more naturally and efficiently. The application of AI in robot manufacturing automation is an active and rapidly evolving field. The combination of deep learning, reinforcement learning, computer vision, and advanced language models is pushing the boundaries of what robots can achieve in manufacturing settings, paving the way for more intelligent, adaptable, and efficient robotic systems.

3. System Analysis and Design

We propose a smart robot manipulation system using advanced generative AI capabilities. Specifically, the system integrates the UR5 robotic arm and OpenAI's GPT-4 model for advanced image processing and neural network learning. This enables the system to recognize the workpiece object's coordinates and translate this position data into precise robotic manipulation, achieving smart robotic control. By implementing this smart robot control model,

manufacturing automation can effectively handle critical tasks such as perception, task planning, execution of actions, and learning from failures or successes. This study focuses on robot manipulation, exploring the potential of integrating GPT-4 and neural networks into robotic control to enhance object recognition through vision and coordinate extraction. For this purpose, we utilize the GPT-4 Vision capability for robot vision.

As shown in Figure 1, the smart robot manipulation system is composed of image acquisition, object detection, coordinate extraction, neural network processing, and robotic manipulation. During image acquisition, an overhead camera and two side cameras are mounted on the workbench of the robotic arm. These cameras capture images of the robotic arm's gripper working on the workpiece. Having this workstation as a constant working environment allows each perspective to remain consistent throughout its operation. The OpenAI GPT-4 model processes images captured by these cameras to find an object's RGB values. User input from either a human controller or a task planner determines which individual workpiece will be the target. Individual workpieces are labeled as color-coded objects or identified using part numbers in barcodes or QR codes. Finally, the robotic arm performs linear movements to grasp a test tube, which can then be transported to a separate workstation or another defined location.

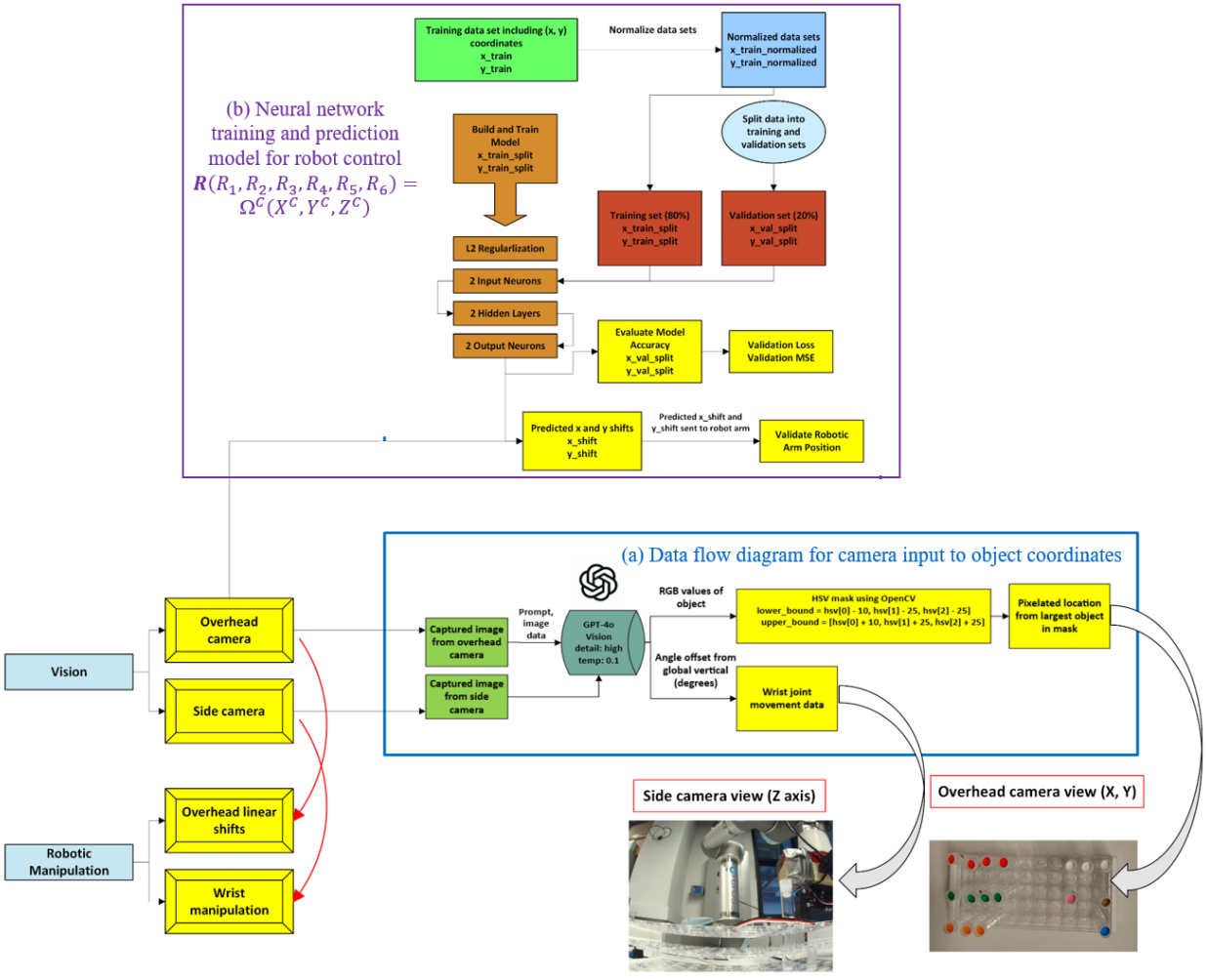


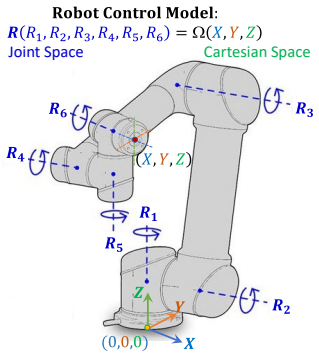
Figure 1. System architecture of smart robot manipulation using GPT-4o Vision

3.1 Problem Formulation of Robot Manipulation

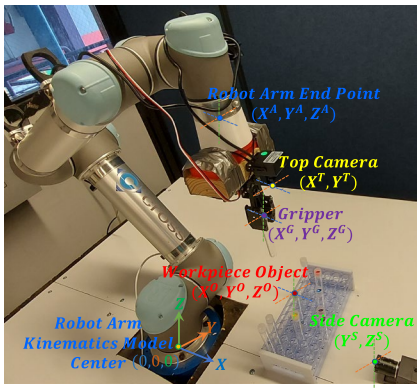
Typically, a robotic arm from a particular manufacturer (e.g., the UR5 in our case) is equipped with robot kinematics models that enable transformation (i.e., calculation) between a spatial position and the movement of the robotic arm. For example, Figure 2(a) shows an inverse kinematics model that calculates the necessary robot joint rotation parameters, $\boldsymbol{R}(R_1, R_2, R_3, R_4, R_5, R_6)$, to reach a particular location characterized by the Cartesian coordinates of the robot head (X, Y, Z), within the robot coordinate plane. This transformation from a desired location in Cartesian space

to the control of six rotation parameters in the robot's joint space involves a robot control model Ω, i.e., $\boldsymbol{R}(R_1, R_2, R_3, R_4, R_5, R_6) = \Omega(X, Y, Z)$. While this 'built-in' robot kinematics model manages to control the robot joint rotations to reach any position, this position is defined as the center point of the robot end, to which a gripper is attached. Since the center of the gripper is where the actual robot operation takes place when picking up the workpiece, an offset Δ^{A-G} exists between the positions of the robot endpoint (X^A, Y^A, Z^A) and the gripper center (X^G, Y^G, Z^G), as shown in Figure 2(b). To achieve a successful robot operation, like grasping a workpiece, the robot joint rotations must make the gripper center reach the target location by compensating for this offset amount on top of the coordinates of the robotic arm endpoint. Thus, the actual robot control model should be $\boldsymbol{R}(R_1, R_2, R_3, R_4, R_5, R_6) = \Omega^G(X^G, Y^G, Z^G)$. This coincides with the typical practice of robot calibration in practical applications, which is often done manually through a tedious trial-and-error process. This results in a fixed setup of the workstation, where the robot joint parameters and locations of the workpiece are predefined properly according to the calibration. However, once the workpiece location changes for different tasks, the entire calibration must be redone.

To address this challenge, robot vision can be employed to recognize the location of the workpiece and guide the robotic arm to move the gripper to the intended position. However, the object position (X^C, Y^C, Z^C) determined by image recognition is referenced to the physical position of the camera mounted on the robotic arm end. As depicted in Figure 2(b), there exists another offset Δ^{C-G} between the camera and the gripper center. This necessitates an additional calibration step to compensate for this offset, ensuring that the gripper moves to the correct location using $\boldsymbol{R}(R_1, R_2, R_3, R_4, R_5, R_6) = \Omega^G(X^G, Y^G, Z^G)$. These two physical offsets and the corresponding calibrations pose significant challenges for precise robot manipulation. To address this, we propose bypassing these calibration steps by utilizing artificial neural networks to train a smart robot control model. This model would directly determine the robot joint rotation parameters based on image data from the robot's vision system, which correlates with the actual gripper locations. Specifically, we aim to control the robot with $\boldsymbol{R}(R_1, R_2, R_3, R_4, R_5, R_6) = \Omega^C(X^C, Y^C, Z^C)$. To establish the 3-D position of the gripper center, we employ a top-mounted camera on the robotic arm to recognize the horizontal location of objects on the workbench within the camera's coordinate plane, denoted as (X^T, Y^T). Similarly, a side camera can determine the vertical location of objects within its own coordinate plane, as shown in Figure 2(b) where a side camera mounted on the workbench establishes cartesian coordinates in the vertical plane (Y^S, Z^S). For fine-tuning the object's position in the vertical plane, another side camera can gather image data from a different angle, capturing coordinates (X^S, Z^S).



(a) Robotic arm inverse kinematics model



(b) Configuration of a robot manipulation system

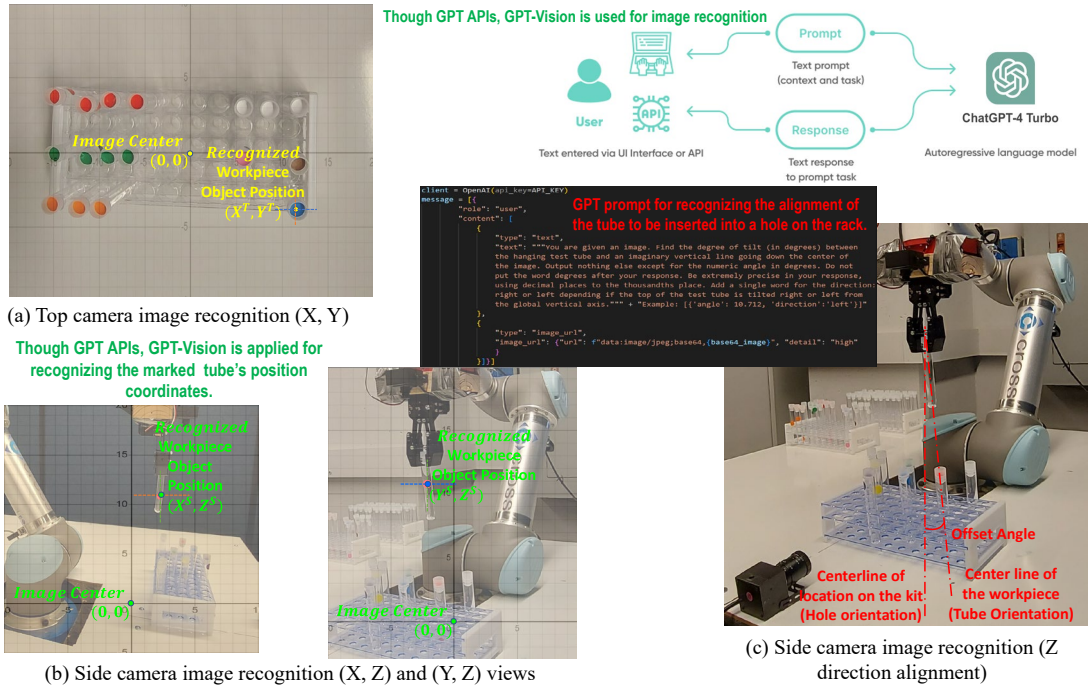
Figure 2. Formulation of a robot control model for robot manipulation in a workstation

3.2 Design of a Smart Robot Control Model using GPT Vision

Our smart robot control model Ω^C effectively bypasses the offsets Δ^{A-G} and Δ^{C-G} inherent in traditional kinematics calibration. Instead, it learns the complex mapping relationships between input (camera image data) and output (joint rotation parameters) autonomously, functioning like a black box. Powered by GPT-enhanced neural network models, this smart robot control system can accurately predict the necessary robot manipulations $\boldsymbol{R}(R_1, R_2, R_3, R_4, R_5, R_6)$ to move the gripper to the target position indirectly identified in camera images (X^C, Y^C, Z^C). This approach enables seamless end-to-end data acquisition, analysis, and action without the need to explicitly handle or calibrate parameters such as (X^A, Y^A, Z^A), (X^G, Y^G, Z^G), Δ^{A-G}, and Δ^{C-G}. The smart robot control model leverages GPT-4o for its

operations. GPT prompts are integrated into the foundational robot control model using Python code and GPT APIs. This approach eliminates the need for manual construction of neural network models, as ChatGPT generates suitable neural network code directly in Python, seamlessly embedding it into the core robot control framework. Our process for integrating GPT capabilities into the basic robot control Python codes follows these steps:

Step 1: Integrate the OpenAI GPT APIs into a Python library by (a) Signing up for an OpenAI API key; (b) Installing the OpenAI Python module using pip, the Python package manager (pip install openai); (c) Importing the OpenAI module into our Python code by adding a line (import openai) at the beginning of the script; and (d) Authenticate the API key, (openai.api_key = "Name_of_API_KEY") (Figure 3).



(a) Top camera image recognition (X, Y)

(b) Side camera image recognition (X, Z) and (Y, Z) views

(c) Side camera image recognition (Z direction alignment)

Figure 3. GPT-powered recognition of workpiece positions in the camera images using cartesian coordinate grids

Step 2: Top camera image recognition using GPT Vision to identify coordinates of the workpiece object, (X^T, Y^T). Figure 4(a) shows an example of recognized object coordinates in one image of the top camera. as shown in To generate text using the OpenAI API, we define a function that takes a prompt as input and returns the generated text. This function uses the openai.Completion.create method to send a request to the OpenAI API with the specified parameters, which include the text prompt, the maximum number of tokens to generate, and the temperature of the generation process, among others. The generated text is then returned as a string, as shown in Figure 4(b).

```
def generate_text(prompt):
    response = openai.Completion.create(
        engine="davinci",
        prompt=prompt,
        max_tokens=1024,
        n=1,
        stop=None,
        temperature=0.7,
    )
    return response.choices[0].text.strip()
```

(a) Define a function to call the GPT API

define a function to call the OpenAI API

```
def get_object_coordinates():
    # Assuming camera is defined somewhere in your code
    ret, frame = camera.read()

    if not ret:
        print("Error: Could not read frame from camera.")
        return []

    filename = 'live_feed_image.png'
    frame_corrected = gamma_trans(frame, 0.5)  # Assuming
gamma_trans is defined somewhere

    cv2.imwrite(filename, frame_corrected)
    cv2.imshow(filename, frame_corrected)
    print("Screenshot saved as " + filename)

    with open(filename, 'rb') as image_file:
        base64_image =
base64.b64encode(image_file.read()).decode('utf-8')


    parameters = {"temperature": 0.1, "model": "gpt-4o",
"messages": message, "max_tokens": 300}
    result = client.chat.completions.create(**parameters)
    coordinates = eval(result.choices[0].message.content)

    return coordinates
```

```
client = OpenAI(api_key=API_KEY)
message = [
{
    "role": "user",
    "content": [
        {
            "type": "text",
            "text": "This is an image. Use the l
provided (a grid) as a guide overlayed over the firs
Identify where the closest object is in the frame us
cartesian coordinates. Format your answer in the for
Python list. Use decimals to the tenths place in eac
coordinates. Return nothing else except for this Pyt
Example: [{'x': 10.2, 'y': 5.6}]"
        },
        {
            "type": "image_url",
            "image_url": {
                "url": f"
_image}",
                "detail": "high"
            }
        },
        {
            "type": "image_url",
            "image_url": {
                "url": f"https://jesus.is-
from.space/r/Grid.png"
            }
        }
    ]
}
]
```

Top Camera:
Text prompt to the OpenAI API

(b) Example of using the OpenAI API for the top camera

Figure 4. Camera image recognition using GPT Vision through the OpenAI API

Step 3: Create a deep learning neural network model using ChatGPT for robot control by the top camera, i.e., $\Omega^{C}{}_{|(X^T,Y^T)}$. The input layer contains two neurons corresponding to (X^T, Y^T). The output layer comprises six neurons expressing rotation parameters R_1, R_2, R_3, R_4, R_5, and R_6, respectively. Figure 5 shows our prompt to ChatGPT and the neural network architecture generated by ChatGPT, along with its implementation in Python.

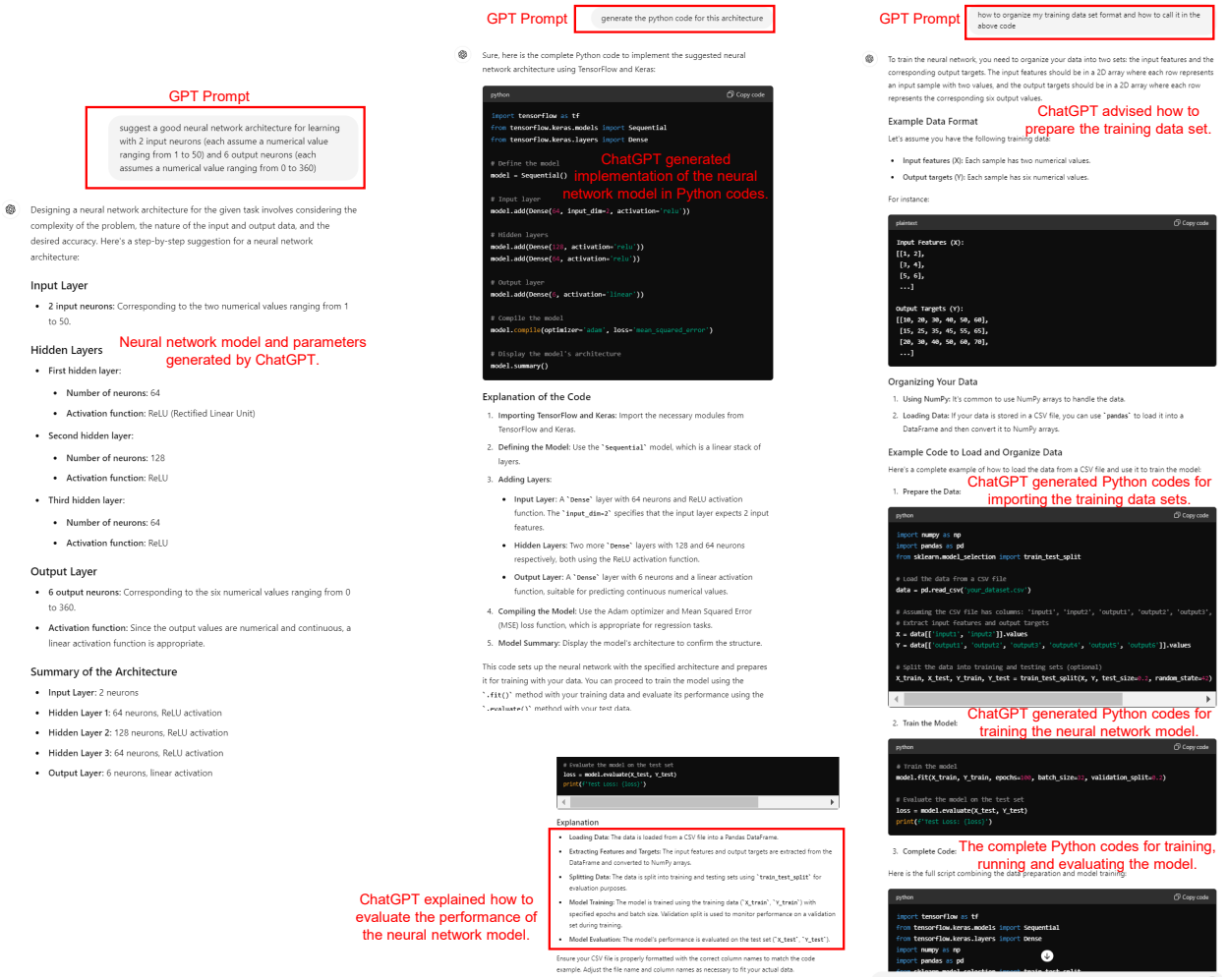


Figure 5. Deep learning neural network model generated by ChatGPT along with its implementation in Python

Step 4: Similar to Step 3, use ChatGPT to generate neural network model for smart robot control by side cameras, i.e., $\Omega^{C}{}_{|(Y^S,Z^S)}$ and $\Omega^{C}{}_{|(X^S,Z^S)}$. Figure 3(b) shows examples of image recognition from the side cameras. The generated side camera prediction models are used for height control of robot movements.

Step 5: Side cameras are also used to fine-tune alignment control of workpiece operations, e.g., the gripper picking up a part from or inserting a part into a location accurately. Again, GPT Vision is embedded in the smart robot control model $\Omega^{C}{}_{|Z)}$using the GPT API. Figure 3(c) shows such an alignment control scenario with an example GPT prompt.

Step 6: Create a multi-view ensemble learning model for integrating image recognition results of top and side cameras for a coherent and holistic smart robot control model, i.e., $\Omega^{C} = \Omega^{C}{}_{|(X^T,Y^T)} \cap \Omega^{C}{}_{|(Y^S,Z^S)} \cap \Omega^{C}{}_{|(X^S,Z^S)} \cap \Omega^{C}{}_{|Z)}$. Similarly, this ensemble learning model is generated using ChatGPT, as shown in Figure 7.

4. GPT Vision Powered Image Recognition

The vision capabilities of GPT-4o enable sophisticated processing and analysis of visual data, which is crucially dependent on handling complex visual patterns and contextual information effectively and consistently. GPT vision processes a base64-encoded image from either an overhead or side camera. Alongside the image, a prompt is provided, which is essential for guiding the model to accurately identify the desired object and its color within the image context. Through prompt engineering, users or task planning models can specify inputs to focus the model on specific objects or locations. Setting a format with relevant examples and adjusting the model's temperature value ensures consistent

results that are usable across the system. Moreover, maintaining a consistent frame of reference for each captured image ensures any errors are predictable and easily correctable.

To establish a Cartesian coordinate grid for camera image recognition, we employ a color mask in the HSV (Hue, Saturation, Value) color space. The HSV range is divided into three channels: Hue (0-180), Saturation (0-255), and Value (0-255). A mask is created based on these HSV values and overlaid on the raw captured image. This allows a color detection algorithm to isolate the largest object defined by the mask and determine its coordinates. Figures 3(a)-(b) depict Cartesian coordinate grids overlaid on images captured by the top and side cameras. One limitation of the current GPT Vision model is its accuracy in providing relative or absolute object locations within an image. Therefore, these tasks are delegated to an OpenCV masking function, which precisely calculates the coordinates of specified colors in the image. Figure 1(a) illustrates the process of locating an object on a scaled 40x20 Cartesian coordinate grid. OpenCV plays a pivotal role in image processing tasks such as masking and coordinate extraction, ensuring precise determination of color-specific locations within the image. By integrating GPT-4o with OpenCV, we achieve accurate object coordinate definitions on a scaled grid, enhancing spatial data accuracy beyond the capabilities of GPT-4o alone.

One challenge lies in using relative Cartesian coordinates (X^C, Y^C, Z^C) for manipulating the robotic arm, which operates based on six joint rotation parameters $\boldsymbol{R}(R_1, R_2, R_3, R_4, R_5, R_6)$. To address this, a neural network is trained to correlate these coordinate pairs with linear movements along the horizontal axes, measured in meters. For instance, the UR5 library, URBasic, facilitates linear robot movement using its native functions. These functions accept positional changes for each rotation axis as inputs, ensuring that the end effector remains parallel to the workstation surface during motion. Additionally, since the approach involves relative changes rather than absolute joint angles, it can be applied universally across different workstations. Figure 1(b) outlines the workflow of training the neural network and its predictive model for robot control $\boldsymbol{R}(R_1, R_2, R_3, R_4, R_5, R_6) = \Omega^C(X^C, Y^C, Z^C)$.

5. GPT-Powered Ensemble Learning for Fusion of Multi-View Image Recognition

In the fusion of multi-view data, traditional machine learning approaches like discriminant analysis, kernel machines, and spectral clustering often concatenate multiple views into a single unified view for learning purposes. However, this approach can lead to overfitting, especially with limited training samples, and lacks physical interpretability as each view possesses unique statistical properties and semantics. In contrast, ensemble learning approaches advocate for using multiple functions to model each view separately and collectively optimizing them to leverage the diverse perspectives of the input data, thereby enhancing learning performance (Zhang and Ma, 2012). Ensemble methods are meta-algorithms that combine several machine learning techniques into a unified predictive model to mitigate variance (bagging), bias (boosting), or enhance predictions (stacking). Originally developed to reduce variance and improve accuracy, ensemble learning trains combinations of multiple models (referred to as base learning models or ensemble members), resulting in superior predictive performance compared to individual models (Parikh and Polikar, 2007), particularly when the errors made by these base models are minimally correlated (Zhang and Ma, 2012).

Furthermore, the effectiveness of ensemble learning in reducing generalization error varies across different training datasets. To address diverse semantic aspects in product usage datasets, we advocate for learning robot control models using ensembles of heterogeneous features. However, using identical training datasets for all individual models may limit the diversity among base models, whereas using entirely different datasets may compromise the accuracy of individual base models. Therefore, constructing well-balanced training datasets that strike a suitable trade-off between accuracy and diversity is crucial for effective ensemble learning. To tackle these challenges, we propose a novel architecture called Multi-View Ensemble Learning (MEL), depicted in Figure 6. MEL aims to integrate multiple views of data while maintaining diversity among base models, thereby enhancing the robustness and performance of ensemble learning approaches.

5.1 Data Preprocessing and View-wise Classification

Encouraging diversity in training datasets for individual learning models is crucial for reducing variability in ensemble learning. We acknowledge that unrepresentative samples can significantly impact model accuracy, leading to frequent misclassification patterns. To address this, we propose a novel approach that preprocesses the dataset into more homogeneous clusters corresponding to multiple data views, thereby enhancing the stability of predictions in inverse relationships. Our rationale is to promote diversity by increasing homogeneity within reclassified sample clusters while maintaining heterogeneity between different cluster groups.

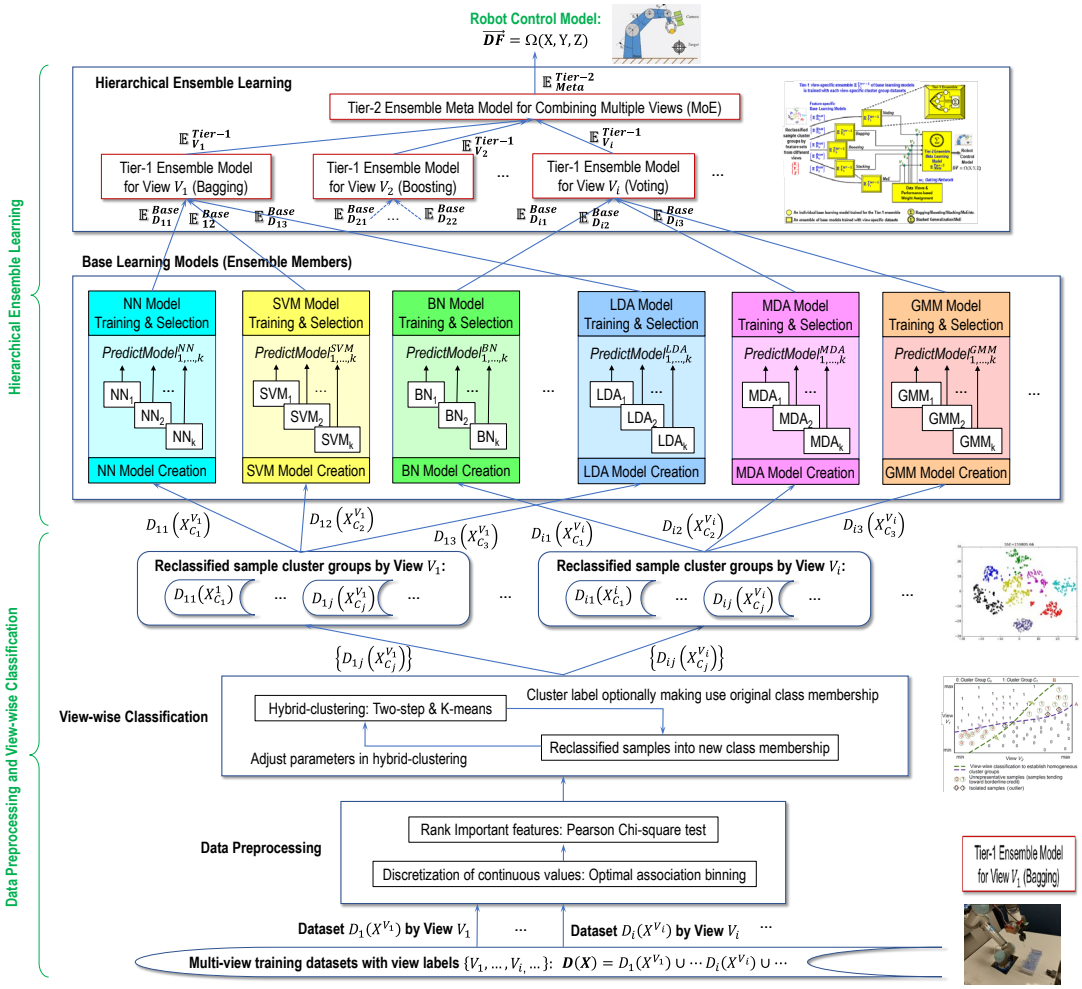


Figure 6. Systems architecture of the multi-view ensemble learning decision process

Initially, our Multi-View Ensemble Learning (MEL) method begins with preprocessing the raw dataset, discretizing continuous values into discrete states suitable for prevalent learning paradigms that handle categorical data, such as CART decision trees, C4.5, Bayesian networks, and instance-based learning. We employ optimal binning associated with feature selection using a Pearson Chi-square-based importance measure. Subsequently, MEL refines sample quality through view-wise classification, labeling the training dataset according to different data views $\{V_i\}$. An advantage of this approach is that all samples from a single view, $D_i(X^{V_i})$, can be further categorized into multiple cluster groups, $C_j = \left\{D_{ij}\left(X_{C_j}^{V_i}\right)\right\}$, achieving a balanced mix of heterogeneity across cluster groups and homogeneity within each cluster group and across the same data view. Each cluster group is defined based on a subset of features from a specific data view. The classifiers we develop aim to correctly assign target view memberships based on sample feature values. Our method of view-wise classification employs a hybrid approach combining two-step clustering and K-means clustering, which preprocesses input samples effectively by clustering, ensuring appropriate view memberships. This preprocessing step leverages data-driven perspectives to enhance the quality of input samples, screening out unrepresentative samples to support robust model building processes.

5.2 Hierarchical Ensemble Learning

An MEL system typically comprises three main components: (i) data sampling/selection, (ii) training base learning models, and (iii) combining base models. We propose integrating base model training and ensemble formation within a unified framework of hierarchical ensemble learning, illustrated in Figure 7. For training base models, we establish

a machine learning library that encompasses a range of learning algorithms, progressing from simple parametric to nonparametric statistical methods. In scenarios involving time-dependent product operating data, parametric methods such as Linear Discriminant Analysis, Multiple Discriminant Analysis, and Gaussian Mixed Models can effectively serve as predictive models. For symbolic or categorical data, nonparametric statistical methods or artificial intelligence techniques are recommended for robust predictive reinforcement. For instance, neural networks often outperform traditional classifiers, Bayesian Networks adeptly capture complex variable relationships, and Support Vector Machines serve as robust benchmark techniques. Considering the semantic and statistical characteristics inherent in each data view, appropriate learning paradigms are selected from the model library to develop feature-specific base learning models, denoted as $\left\{\mathbb{E}_{D_{ij}}^{Base}\right\}$. These models are trained using distinct view-specific sample cluster groups, $D_i(X^{V_i}) = \left\{D_{ij}\left(X_{C_j}^{V_i}\right)\right\}$. Therefore, our MEL architecture employs a two-stage hierarchical ensemble method to combine base learning models effectively. In practical scenarios involving product usage, different contexts may be implied by each view of product operating data. This diversity necessitates the use of multiple base learning models within a single view-specific training dataset. To enhance the diversity and performance of ensemble learning, we implement a Tier-1 ensemble stage to aggregate all base feature-specific learning models from the same view, resulting in a view-specific ensemble model, $\mathbb{E}_{V_i}^{Tier-1}$. Subsequently, a Tier-2 ensemble stage combines these Tier-1 ensemble models from different views. At the Tier-1 ensemble stage, various ensemble strategies (such as bagging, boosting, stacking, or voting) are applied based on the specific contextual implications of each feature set within a particular data view. The resulting Tier-1 ensemble models then serve as the foundational base models for Tier-2 ensemble, which aggregates different views into a meta model, $\mathbb{E}_{Meta}^{Tier-2}$.

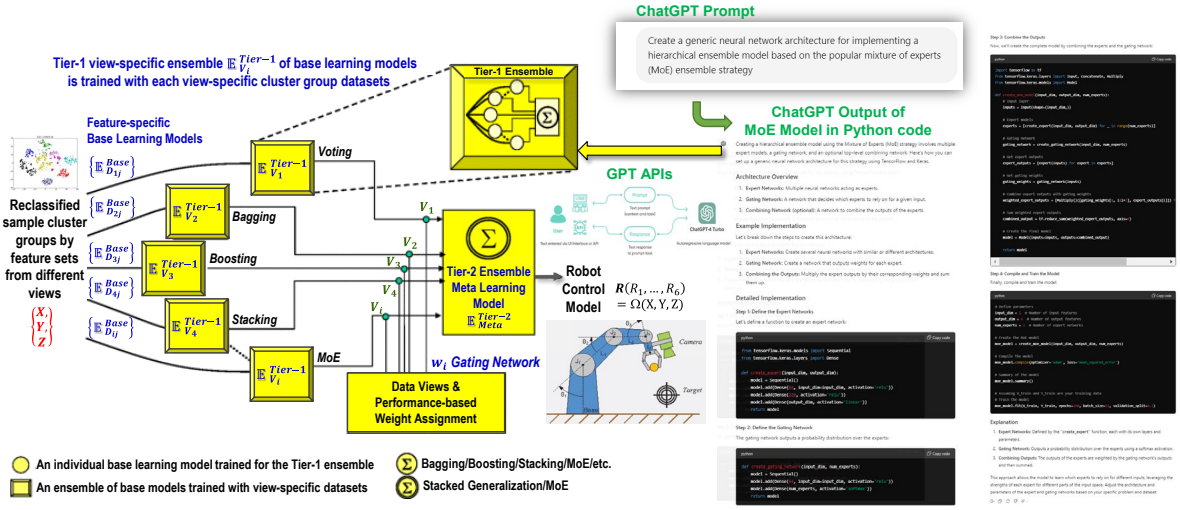


Figure 7. An MoE-based hierarchical ensemble learning model for data fusion implemented using ChatGPT

We develop a hierarchical ensemble model based on the mixture of experts (MoE) ensemble strategy, a concept similar to stacked generalization (Jacobs et al., 1991; Jordan and Jacobs, 1994; Wolpert, D. H., 1992). Illustrated in Figure 7, our MoE-based hierarchical ensemble method trains a set of Tier-1 base learning models using appropriate sampling techniques. These individual models are then combined using a weighted combination rule or another suitable ensemble strategy, where the weights are determined by a gating network. Typically, this gating network is trained using algorithms such as expectation-maximization (Jordan and Xu, 1995) or confidence-weighted voting on the original training data. The gating network dynamically assigns weights based on the input data, allowing the MoE ensemble to effectively learn which parts of the feature space are best represented by each ensemble member. To implement this MoE ensemble learning model, we utilize ChatGPT to generate a generic neural network model in Python code, as depicted in Figure 7. ChatGPT is instrumental in constructing the model and generating Python code in response to our prompts. This MoE model is seamlessly integrated into the robot control model using GPT APIs.

6. Case Study of Robotic Bio-manufacturing Kitting Operations

Robotic kitting automation plays an important role in the biomanufacturing industry by encouraging suppliers to create robot-friendly containers, thereby reducing packaging waste. Subcontracted kitting has the potential to deliver consumables in bulk, minimizing overall packaging. This approach allows kitting companies to supply local drug companies with kits in reusable containers. To evaluate the effectiveness of our smart robot manipulation model, we established a testbed in this context. Figure 8 depicts our experimental setup, simulating a robot kitting automation process. The robot task begins by picking up materials from two incoming supply zones, each containing tubes labeled differently to fulfill various job orders. The locations of these supply zones are fixed and predefined in the UR5 basic control model, allowing the robotic arm to swiftly navigate to the designated zones. Similarly, the kitting zone (rack) is positioned at a predefined location on the workbench, so that the robotic arm can move there first, before camera image recognition is engaged to accurately identify the objects or locations for further action.

We designed several benchmark experiments to evaluate the performance of our smart robot control model. In the first test case, our objective is to train a neural network that correlates cartesian coordinates to the six degrees of freedom representing the final position where the robotic arm will move before picking up a test tube, specifically $\boldsymbol{R}(R_1, R_2, R_3, R_4, R_5, R_6) = \Omega^C{}_{|(X^C, Y^C, Z^C)}$. Each object's position is identified using one top camera and two side cameras simultaneously. The neural network model is designed to learn mappings from spatial positions to corresponding joint rotations, creating an integrated model where image inputs from the three cameras are processed collectively. To construct the training dataset, we manually position the UR5 robotic arm to specific target locations where the workpiece objects are located, i.e., (X^O, Y^O, Z^O). For each of these positions, we record the corresponding six joint rotation values necessary to reach them. This dataset serves as the training input for the neural network model, which is designed to predict the values of the six joint rotation parameters based on input coordinate pairs identified in the camera images, as illustrated in Figure 3(a)-(b).

However, predicting the joint angles accurately based on cartesian coordinates posed a significant challenge for the model, especially for specific locations. Moreover, this approach would only be effective for a specific section of the workstation. Improving the neural network's accuracy would necessitate an impractical amount of additional data points for each environment. Therefore, we are considering a two-step smart robot control strategy: first, moving the gripper horizontally to the (X^C, Y^C) position, followed by vertical adjustment along the Z-axis. In this approach, the top camera neural network model correlates two input neurons to two output neurons denoting X and Y shifts. Theoretically, this pragmatic model, $\Omega^C{}_{|(X^T, Y^T)} \cap \Omega^C{}_{|(Y^S, Z^S)} \cap \Omega^C{}_{|(X^S, Z^S)}$, is expected to be more efficient in training as it incorporates domain knowledge specific to machine learning.

We further consider a third benchmark case, which is a more traditional kinematics-based approach and less of a smart model. This approach uses the robot kinematics model, Ω^C, to determine the position of the robotic arm end, (X^A, Y^A, Z^A), and then employs a neural network model to predict the offset from (X^A, Y^A, Z^A) to the target position (X^C, Y^C, Z^C). We refer to this strategy as a semi-smart robot control model, $\Omega^A{}_{|(X^A, Y^A, Z^A)} \cap \Omega^{A-C}{}_{|(X^C, Y^C, Z^C)}$. We also use ChatGPT to generate this neural network model with three input neurons and three output neurons.

While systematic neural network evaluation procedures/functions generated by ChatGPT could be used to compare learning and prediction performance, we propose using more meaningful performance indicators aligned with the specific domain problem context. For evaluating the prediction performance of the neural networks in smart robot control, we define performance measures based on the success rate of completing specific robot manipulation tasks. For instance, these tasks include accurately picking up a tube from the rack without interfering with nearby tubes or correctly placing a tube into the designated hole on the rack without misalignment. Furthermore, to analyze the learning performance of the neural networks, we focus on comparing the size of the training dataset required for the robot control model to achieve the desired success rate in task fulfillment.

Table 1 presents the results of our experiments for performance comparison. The semi-smart model overall shows inferior performance compared to the other two models. This can be attributed to the inherent uncertainties in the built-in robot kinematics model Ω^A, as well as the practical challenges in calibrating the offsets Δ^{A-G} and Δ^{C-G}. Nonetheless, using neural network learning for Ω^{A-C} can streamline the tedious, manual, and error-prone calibration process. As shown in the table, the all-in-one model performs better than the semi-smart model but falls short

compared to the two-step model. Both models implement our smart robot control strategy powered by GPT Vision. The all-in-one model treats smart robot control as a pure machine learning problem, overlooking the domain-specific context. In contrast, the two-step model enhances performance by integrating domain knowledge. This aligns with the consensus on the hallucination problem of LLMs' in that they struggle with understanding cause-and-effect relationships and logical flow, for which domain knowledge matters.

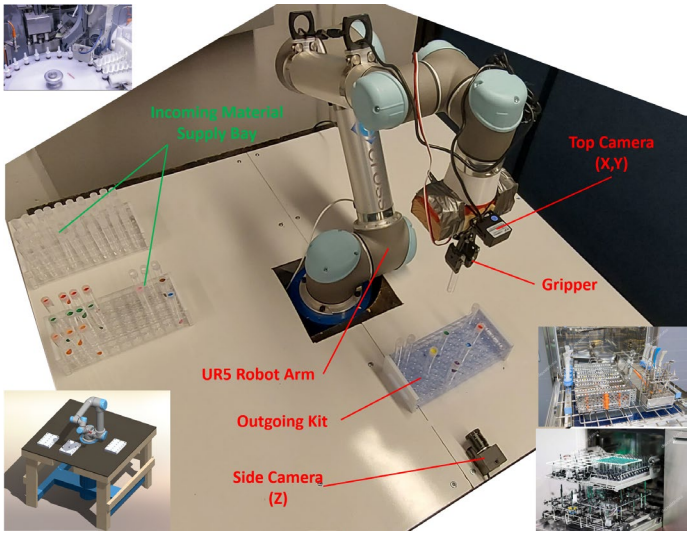


Figure 8. Robot automation testbed setup for bio-manufacturing kitting operations

Table 1. Experiment results for performance comparison

Testing Cases	Successful rate of fulfilling 50 tasks	Size of training data set needed to achieve 95% rate of successful task fulfillment
All-in-one model: $\Omega^{C}{}_{\|(X^C,Y^C,Z^C)}$	94%	115
Two-step model: $\Omega^{C}{}_{\|(X^T,Y^T)} \cap \Omega^{C}{}_{\|(Y^S,Z^S)} \cap \Omega^{C}{}_{\|(X^S,Z^S)}$	99%	22
Semi-smart model: $\Omega^{A}{}_{\|(X^A,Y^A,Z^A)} \cap \Omega^{A-C}{}_{\|(X^C,Y^C,Z^C)}$	88%	250

7. Concluding Remarks

By harnessing OpenAI's GPT-4o model in conjunction with computer vision capabilities like OpenCV, this study illustrates great potential in object recognition, localization, and manipulation within robotic arm systems. The GPT-powered smart robot control approach not only automates manual calibration processes but also reduces errors, enhances throughput, and enhances adaptability to diverse environments. By integrating AI and deep learning, robots can autonomously learn from visual data, continually improving their performance over time. This represents a promising shift towards more intelligent and efficient automated manufacturing. These findings highlight the transformative potential of AI in robotics, offering extensive applications across various industrial sectors beyond traditional kinematics-based methods.

References

Anderson, J. and Kurez, E., Mastery in motion: Analyzing state-of-the-art approaches to robot manipulation. *EasyChair*, No. 12329, 2024.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E., Adaptive mixtures of local experts, *Neural Computation*, vol. 3, no. 1, pp. 79-87, 1991.

Jin, C., Tan, W., Yang, J., Liu, B., Song, R., Wang, L., and Fu, J., AlphaBlock: Embodied finetuning for vision-language reasoning in robot manipulation, Available online: https://doi.org/10.48550/arXiv.2305.18898, 2023.

Jordan, M. J. and Jacobs, R. A., Hierarchical mixtures of experts and the EMalgorithm, *Neural Computation*, vol. 6, no. 2, pp. 181-214, 1994.

Jordan, M. I. and Xu, L., 1995, Convergence results for the EM approach to mixtures of experts architectures, *Neural Networks*, vol. 8, no. 9, pp. 1409-1431, 1995.

Karapinar, S., Altan, D., and Sariel-Talay, S., A Robust Planning Framework for Cognitive Robots, *AAAI 2012 Workshops*, pp 102-108, Toronto, Canada, 2012.

Levine, S., Finn, C., Darrell, T., and Abbeel, P., End-to-end training of deep visuomotor policies, *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1-40, 2016.

Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., and Quillen, D., Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection, *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421-436, 2018.

Li, S., Chu, Y., Zheng, P., Wang, X.V., and Wang, L., Mutual-cognition for proactive human-robot collaboration: A mixed reality-enabled visual reasoning-based method, Available online: https://www.tandfonline.com/doi/full/10.1080/24725854.2024.2313647, 2024.

Liu, J., Yuan, Y., Hao, J., Ni, F., Fu, L., Chen, Y., and Zheng, Y., Enhancing Robotic Manipulation with AI Feedback from Multimodal Large Language Models. *arXiv preprint,* arXiv:2402.14245, 2024.

Liu, Z., Dai, H., Zhao, L., Ge, B., Li, X., Liu, T., and Zhang, Shu., Large Language Models for Robotics: Opportunities, Challenges, and Perspectives, *Journal of Latex Class Files,* vol. 14, no. 8, 2021.

Mahler, J., Pokorny, F.T., Hou, B., Roderick, M., Laskey, M., Aubry, M., Kohlhoff, K., Kröger, T., Kuffner, J., and Goldberg, K., Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards, *IEEE Int. Conf. on Robotics and Automation*, pp. 1957-1964, 2016,

Wang, J., Wu, Z., Li, Y., Jiang, H., Shu, P., Shi, E., Hu, H., Ma, C., Liu, Y., Wang, X., and Yao, Y., Large language models for robotics: Opportunities, challenges, and perspectives. *arXiv preprint*, arXiv:2401.04334, 2024.

Wolpert, D. H., Stacked generalization, *Neural Networks*, vol. 5, no.2, pp. 241-259, 1992.

Zhang, C. and Ma, Y., *Ensemble machine learning: Methods and Applications,* Springer, 2012.

Biographies

Ryan Tan is a senior at George Walton Comprehensive High School in Marietta, GA, USA pursuing the Walton STEM Academy Engineering pathway. In Summer 2024, he conducted research on AI for robot manufacturing in collaboration with undergraduate and graduate research students and the professor in the Advanced Design and Manufacturing Systems (ADAMS) lab in the School of Mechanical Engineering at Georgia Tech. His research interests include industry applications of neural networks, applications of artificial intelligence in manufacturing automation, and aerospace engineering. More information about him: https://www.linkedin.com/in/ryantan-rt/

Harry Yang is a senior at George Walton Comprehensive High School in Marietta, GA, USA pursuing the Walton STEM Academy Engineering pathway. In Summer 2024, he conducted research on AI for robot manufacturing in collaboration with undergraduate and graduate research students and the professor in the Advanced Design and Manufacturing Systems (ADAMS) lab in the School of Mechanical Engineering at Georgia Tech. His research interests include electrical systems engineering, AI applications in bio-manufacturing, and manufacturing automation.

Sean Jiao is a senior at George Walton Comprehensive High School in Marietta, GA, USA. In Summer 2024, he conducted research on AI for robot manufacturing in collaboration with undergraduate and graduate research students and the professor in the Advanced Design and Manufacturing Systems (ADAMS) lab in the School of Mechanical Engineering at Georgia Tech. His research interests include AI applications in bio-manufacturing, decision sciences, production planning and control, and systems engineering. More information about him: https://www.linkedin.com/in/sean-jiao-aaa626233/

Liuyang Shan is a BS/MS student in the School of Mechanical Engineering at the Georgia Institute of Technology, USA. He is currently finishing his B.S. degree and preparing for his M.S. studies. His research interests focus on the applications of AI in advanced manufacturing systems and automation.

Chenxi Tao is currently a graduate student in the School of Mechanical Engineering at Georgia Institute of Technology, USA. His research interests include large language models for industrial applications, artificial intelligence, robot learning, design optimization, operation planning, decision making, and data analysis. He recently graduated from Columbia University in New York City, USA, with an M.S. in Mechanical Engineering. He also holds a B.Eng. in Mechanical Engineering from Shanghai Maritime University, China.

Roger Jiao is the editor-in-chief of Journal of Engineering Design and an associate professor of mechanical and industrial engineering at Georgia Tech, USA. Prior to joining the School of Mechanical Engineering at Georgia Tech in December 2008, he was an Assistant Professor and then Associate Professor in the School of Mechanical and Aerospace Engineering at Nanyang Technological University, Singapore. Before his career in Singapore, he was a Visiting Scholar in the Department of Industrial Engineering and Engineering Management at Hong Kong University of Science and Technology from 1998 to 1999. From 1993 to 1994, he was a Lecturer of Industrial Engineering in the School of Management at Tianjin University, China. From 1988 to 1990, he worked as an Associate Lecturer in the

Department of Industrial Design at Tianjin University of Science and Technology, China. More info about his research: https://scholar.google.com/citations?user=9yikEHAAAAAJ&hl=en&oi=ao.