Manipulation using LLM and Modular CV Model Integration

ANONYMOUS AUTHOR(S)

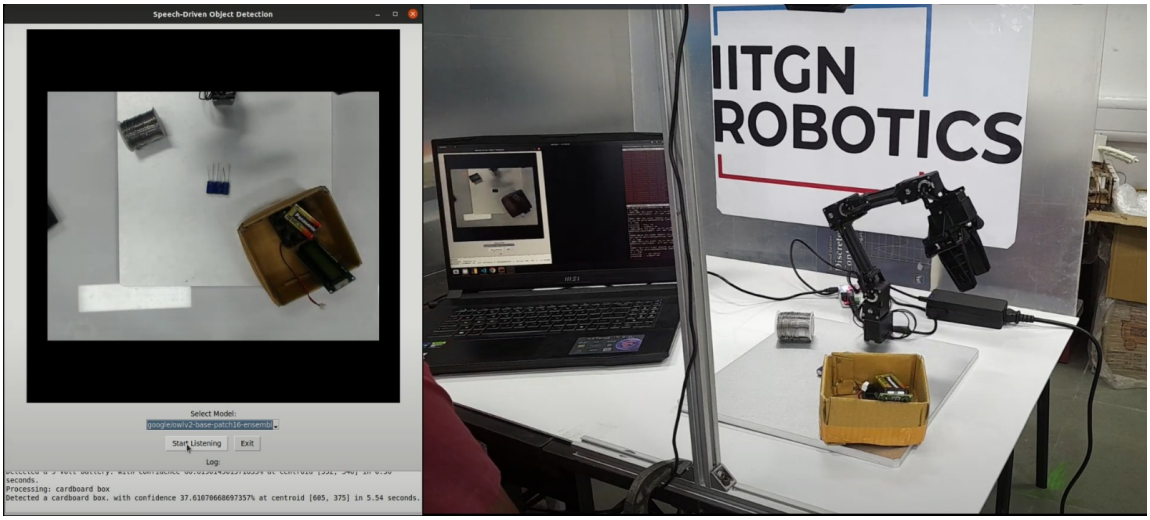


Fig. 1. OpenMANIPULATOR-X's workspace, where the experiments were performed (on the right) and the UI of the pipline (on the left)

Achieving seamless robotic manipulation through voice commands is made possible by advancements in voice recognition, machine learning, and computer vision. A key strength lies in the modularity of the computer vision models, allowing for the integration of various zero-shot object recognition models tailored to specific tasks. This adaptability ensures the system can be easily updated or expanded as new models emerge. The process involves converting voice commands into text, extracting relevant objects using a Large Language Model, and determining the manipulator's joint angles via an Inverse Kinematics algorithm. This approach enhances scalability, flexibility, and future-proofing for advanced robotic manipulation.

CCS Concepts: • **Computing methodologies** → *Motion path planning*; **Vision for robotics**; **Information extraction**; **Object detection**; *Speech recognition*; *Machine translation.*

Additional Key Words and Phrases: Robotic Manipulation, Speech-To-Text, Large Language Models, Human-Robot Interaction, Zero-Shot Object Detection

ACM Reference Format:

Anonymous Author(s). 2018. Manipulation using LLM and Modular CV Model Integration. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 17 pages. https://doi.org/XXXXXXX.XXXXXXX

Manuscript submitted to ACM

1 Introduction

The quest for seamless human-machine interaction has witnessed substantial progress in recent years, driven by advancements in voice recognition technologies, machine learning, Large Language Models (LLMs) and computer vision. Within the domain of robotic manipulation, enabling a robot to execute complex tasks through simple voice commands represents the convergence of these technologies and epitomizes their potential for real-world applications.

A central strength of the proposed system lies in its modular integration of computer vision (CV) models. This modularity facilitates the incorporation of diverse zero-shot object detection models, each optimized for specific object types or tasks, thereby enhancing the system's flexibility and scalability. Such an approach ensures that the pipeline can adapt to various environments and application domains, addressing the limitations of monolithic vision systems.

To demonstrate the practical utility of this research, we propose a versatile application: a robotic assistant designed for electrical laboratories and machine workshops. This assistant is capable of performing manipulation tasks on tools and components within its configuration space (C-space) based on user voice commands. The manipulator used in this study, the OpenMANIPULATOR-X, is a 4-DOF robotic arm capable of precise object manipulation.

1.1 Contributions

This paper introduces a novel modular pipeline that integrates an LLM, zero-shot object detection models, and inverse kinematics for robotic manipulation. The primary contributions of this work include the demonstration of modular computer vision (CV) model integration, enabling versatile and scalable object detection across various tasks. Additionally, the pipeline is validated through practical experiments conducted in diverse environments, showcasing its adaptability and effectiveness in real-world scenarios.

1.2 Paper Structure

The remainder of this paper is organized as follows:

- **Section 2**: *Methodology* details the components of the proposed system, including speech recognition, LLM integration, zero-shot object detection models, inverse kinematics, and text-to-speech processing.
- **Section 3**: *Experimental Validation* describes the experimental setup, datasets used, and results obtained across different scenarios, such as manipulating tools and components in laboratory environments.
- **Section 4**: *Importance of Modularity* emphasizes the advantages of using modular CV models and their impact on the system's adaptability and robustness.
- **Section 5**: *Challenges Faced* discusses the technical and practical difficulties encountered during the development and deployment of the pipeline.
- **Section 6**: *Future Scope and Discussion* explores potential enhancements to the system and broader applications of the proposed approach.
- **Section 7**: *Conclusion* summarizes the findings and implications of the research, highlighting its contributions and future prospects.

2 Methodology

We can easily curate the sequence of steps we must take to ensure that the manipulation task is accomplished by a mere voice command from the user. So, that directly leads us to the voice recognition part of the pipeline. Once, we recognize the voice from the user and convert it into `"speech_text"`, we feed the same to the LLM model. From which

we would get only specific parts of the sentence which is required for the Zero-shot object recognition CV models (in our case objects present in the sentence). The CV models process the LLM's input and gives us the required coordinates to the manipulator (to perform pick-and-place). We would need an IK algorithm to solve the given coordinates and find the joint angles of the manipulator (here 4 joint angles), so that the manipulation can finally be performed successfully. The above sequence of steps can be simply depicted through the below flowchart.

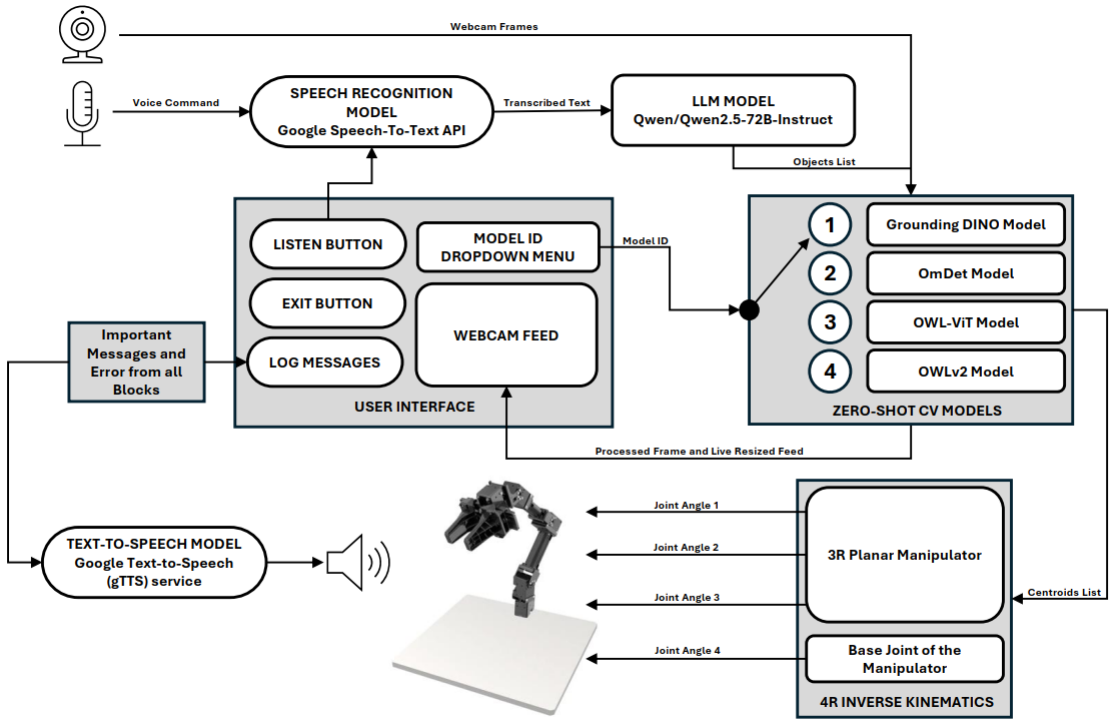


Fig. 2. Flowchart depicting all the blocks present in the pipeline.

From the flowchart we can easily relate to the pseudocode below. The pseudocode is given below for better understanding of the following subsections.

Algorithm 1: Robotic Manipulation Pipeline

Input: Speech recognition, LLM, modular CV models, and inverse kinematics modules are initialized.
Initialize an empty list `detected_centroids` to store object centroids;
Step 1: Initialization;
Announce readiness using text-to-speech (TTS);
Prompt the user to select the required CV model;
Retrieve the selected model ID using modular CV and LLM modules;
Initialize the speech recognizer;
while *True* **do**
  Step 2: Speech Recognition;
  Prompt user to speak via TTS;
  Capture audio input using a microphone;
  if *Speech is successfully transcribed* **then**
    Announce successful recognition via TTS;
  else
    Announce error and prompt the user to try again;
    continue to the next iteration;
  Step 3: Text Processing with LLM;
  Process the `speech_text` to generate actionable commands;
  if *No actionable commands are detected* **then**
    Announce shutdown via TTS;
    Terminate the program and **break** from the loop;
  Step 4: Object Detection Using Zero-shot CV Models;
  foreach *object in the* `ordered_response_list` *from LLM* **do**
    Prepend "a" to the `object_text` for query formatting;
    Perform object detection using the selected CV model;
    Retrieve `centroid`, `confidence`, `processing_time` , and `processed_frame`;
    Display the `processed_frame` temporarily;
    Append the detected `centroid` to `detected_centroids`;
  Step 5: Inverse Kinematics and Pick-and-Place;
  Transform the pixel coordinates of the first and second detected centroids into world coordinates;
  Perform pick-and-place operation using the transformed coordinates;
  Generate a completion message using the LLM and announce it via TTS;
  if *An error occurs during object detection or manipulation* **then**
    Announce the error via TTS;
    continue to the next iteration;

2.1 Speech Recognition Model

Speech recognition models convert audio input, such as sounds from a microphone or pre-recorded files, into text by processing audio signals into meaningful features. They use neural networks to match sounds to words (acoustic models), ensure grammatical correctness with language models, and output the final transcribed text for further use. Some of Speech Recognition Models are DeepSpeech by Mozilla, Whisper by OpenAI and Google Speech-to-Text API.

For our task, we have used Google Speech-To-Text API, which is capable of capturing live audio and processing it in real-time. It supports over 100 languages, including accents like English(India). While performing the task, the language was chosen as "en-IN" to account for the Indian English accent.

2.2 Large Language Model - "Qwen/Qwen2.5-72B-Instruct"

Large Language Models (LLMs) are advanced artificial intelligence systems designed to comprehend, generate, and interact with human language. These models, such as OpenAI's GPT-3 and Google's BERT, are trained on extensive datasets comprising diverse text, enabling them to grasp context, answer questions, and produce human-like text. Their versatility makes them suitable for a range of tasks, including translation, summarization, query answering, and content generation.

For this work, we utilize the LLM "Qwen2.5-72B-Instruct," developed by Alibaba Cloud's Qwen team. This model, part of the Qwen LLM family, is characterized by its architecture of 72.7 billion parameters, which includes 2.7 billion embedding parameters and an 80-layer deep neural network [8]. The model is deployed in two key scenarios to facilitate the robotic manipulation task:

(1) To extract and structure object information from user queries. For instance, given the input query `speech_text`: *"Place the screwdriver on top of the circular plate,"* the model processes the request and returns the objects and their properties in the order of appearance as a comma-separated list: *["screwdriver", "circular plate"]*.
(2) To generate a grammatically correct confirmation statement. For example, for the query `speech_text`: *"Place the screwdriver on top of the circular plate,"* the model produces the statement: *"You have placed the object as requested."*

To integrate this model into our system, we utilize OpenAI's `openai` module. The deployment process involves creating an API key from Hugging Face and configuring the LLM to respond accurately to the specified tasks. The LLM's output in scenario (1) is provided to the zero-shot object detection CV models as a structured list - `"ordered_response_list"`, enabling precise detection and manipulation of objects. Once the manipulation task is completed, the output of scenario (2) is processed by the text-to-speech module to provide an interactive and natural user experience.

2.3 Zero-shot Object Detection CV Models

Zero-shot object detection (ZSD) is a computer vision task where models detect objects in images without prior training on those specific object classes. Given an image and a list of candidate classes, ZSD models output bounding boxes and labels for detected objects. This approach eliminates the need for extensive data annotation and model fine-tuning, making it efficient for applications like image search, object counting, and tracking. In our pipeline we use four different ZSD models, which obtain `"ordered_response_list"` from the LLM and they process the objects present in the list to draw the bounding boxes with certain confidence level or score to eventually calculate the centroid of the specific object asked for.

All of these models follow the same sequence of steps to return us the centroids of the objects and all of them recognize objects better when only a single object is given to it to detect. Hence, define a function which takes "`model_id`", "`object_text`", "`frame_end`" and "`camera_index`" as input parameters and finally returns "`centroid`", "`confidence`", "`processing_time`" and "`processed_frame`" to the user interface.

2.3.1 Grounding DINO Model (tiny variant). The Grounding DINO model integrates the DINO architecture with grounded pre-training to enable open-set object detection. Developed by IDEA-Research, it extends traditional closed-set object detection models by incorporating a text encoder, allowing it to detect objects without prior training on specific classes [4]. The tiny variant of the model achieves notable performance, including an average precision (AP) of 52.5 on the COCO (Common Objects in Context) zero-shot benchmark [5]. This capability makes it particularly effective for diverse applications such as image search and object counting. Its robust generalization across domains without requiring additional training data establishes it as a highly capable tool for zero-shot object detection tasks.

The deployment of this model in Python leverages Hugging Face's `transformers` library, specifically utilizing the `AutoProcessor` and `AutoModelForZeroShotObjectDetection` modules. Initially, input image frames captured using OpenCV are converted from BGR to RGB format, as OpenCV processes images in BGR format, whereas most computer vision models, including Grounding DINO, require images in RGB format. After this conversion, the preprocessed image and the corresponding `object_text` query are fed into the processor, which outputs detection results. These results include bounding boxes around detected objects and their respective confidence scores.

To identify the object with the highest detection confidence, a straightforward algorithm iterates through the bounding boxes to locate the one with the highest confidence score. The bounding box with the highest confidence is visually highlighted in blue to distinguish it from other detections.

The centroid of the detected object is computed to assist in robotic manipulation. This is achieved by calculating the geometric center of the bounding box using coordinate geometry. The centroid plays a crucial role in ensuring that the robotic manipulator can effectively grasp the object while maintaining balance during manipulation tasks.

2.3.2 OmDet Model. The OmDet model, introduced in the work *"Real-time Transformer-based Open-Vocabulary Detection with Efficient Fusion Head"* by Tiancheng Zhao et al., is a state-of-the-art approach for zero-shot (open-vocabulary) object detection. The model leverages the Swin-Tiny-HF architecture, which features a hierarchical structure and shifted windows to reduce computational complexity. This architectural design enables OmDet to deliver efficient and scalable performance across a range of computer vision tasks. The model is particularly effective in detecting objects without predefined labels, ensuring adaptability to diverse applications. Furthermore, its efficient design, coupled with the Swin-Tiny-HF architecture, provides high performance and fast inference, making it well-suited for real-time scenarios where both speed and accuracy are paramount [9].

The deployment of the OmDet model follows a process similar to that of the Grounding DINO model, with the primary difference being the use of the `OmDetTurboForObjectDetection` module to initialize the model. Once deployed, the process for obtaining results, including the highest confidence score, centroids, bounding boxes, and frame outputs, remains identical to the methodology used for the DINO model.

2.3.3 OWL-ViT Model. The OWL-ViT (Vision Transformer for Open-World Localization) is a zero-shot, text-conditioned object detection model proposed by Matthias Minderer et al. The model leverages CLIP (Contrastive Language-Image Pre-Training) as its multi-modal backbone, integrating a ViT (Vision Transformer) for extracting visual features and a causal language model for processing text features. Unlike traditional vision transformers, OWL-ViT removes the final

token pooling layer and attaches lightweight classification and bounding box heads to each transformer output token, enabling efficient and precise object detection [7].

The term *text-conditioned* refers to the model's ability to utilize text queries to guide the object detection process, effectively linking textual descriptions with visual representations. This capability is achieved through the use of contrastive loss, a training strategy that maximizes the similarity between paired image and text embeddings. OWL-ViT's architecture supports open-vocabulary classification, allowing it to detect objects based on textual descriptions without being restricted to predefined labels. The model is trained from scratch and further fine-tuned on detection datasets, enhancing its generalization and detection accuracy [7].

The deployment of the OWL-ViT model follows a process similar to that of the Grounding DINO and OmDet models. However, for initialization, we utilize the `OwlViTProcessor` and `OwlViTForObjectDetection` modules from Hugging Face's `transformers` library. Once deployed, the procedures for obtaining detection results, such as bounding boxes, centroids, and confidence scores, remain consistent with the previously described methods.

2.3.4 OWLv2 Model. The OWLv2 model, short for Open-World Localization version 2, is a zero-shot, text-conditioned object detection model proposed in *"Scaling Open-Vocabulary Object Detection"* by Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. Building on its predecessor, OWL-ViT, OWLv2 leverages CLIP as its multi-modal backbone, integrating a ViT (Vision Transformer) for visual feature extraction and a causal language model for text processing. Unlike OWL-ViT, OWLv2 employs the ViT-B/16 Transformer architecture, which captures finer details compared to the ViT-B/32 used in OWL-ViT [6].

Similar to its predecessor, OWLv2 removes the final token pooling layer and attaches lightweight classification and bounding box heads. It achieves open-vocabulary classification by replacing fixed classification layer weights with class-name embeddings derived from the text model [6]. OWLv2 introduces improved scalability, enhanced real-time performance, and better generalization to unseen objects, making it a significant step forward in open-vocabulary detection tasks [6]. These advancements are attributed to its refined training process and enhanced text processing pipeline, which together ensure superior efficiency and adaptability for diverse applications requiring zero-shot object detection.

While OWLv2 demonstrates notable improvements over OWL-ViT, its deployment process remains unchanged. The model is initialized using the `Owlv2Processor` and `Owlv2ForObjectDetection` modules from Hugging Face's `transformers` library. Once deployed, the methods for obtaining detection results, including bounding boxes, centroids, and confidence scores, are identical to those used for its predecessors.

2.4 Inverse Kinematics

Inverse kinematics (IK) is a fundamental concept in robotics, used to compute the joint configurations required for a manipulator to achieve a specified end-effector position and orientation. Unlike forward kinematics, which determines the end-effector position based on given joint angles, IK operates in reverse by specifying the target location and calculating the necessary joint angles to reach it. This process often involves solving complex, nonlinear equations, frequently requiring iterative techniques such as Jacobian-based methods. IK is indispensable for robotic arm manipulation, enabling precise motion and positioning for applications in domains such as manufacturing, medical robotics, and autonomous systems.

In practice, IK for robots can be implemented using the `MoveIt!` framework by providing the Unified Robot Description Format (URDF) file to the `MoveIt!` setup assistant. Once configured, the generated package includes controller launch files that enable position control for the robot using the default IK solver (`KDLKinematicsPlugin`). However, in our case, the `MoveIt!` package's IK solver proved to be inefficient, exhibiting significant delays and frequently moving into singular configurations even when the target position was easily reachable by the manipulator (OpenMANIPULATOR-X).

Fig. 3. OpenMANIPULATOR-X, a 4-DOF manipulator with 4R configuration. It can be assumed as a 3R planar manipulator affixed on top of base revolute joint.

To address these limitations, we developed a custom IK algorithm tailored to the OpenMANIPULATOR-X, a 4-DOF manipulator (excluding the 1-DOF gripper). The manipulator is modeled as a combination of a 3R planar manipulator (3-DOF) mounted on a revolute joint (1-DOF). This approach enabled us to achieve more efficient and reliable computation of joint angles, ensuring precise and stable performance for the manipulation tasks.

2.4.1 IK of a 3R Planar Manipulator. The inverse kinematics (IK) equations for a 3R planar manipulator can be derived through geometric and trigonometric principles and are commonly detailed in standard robotics textbooks. These equations can also be obtained by applying the laws of vector addition and trigonometric relationships to the diagram shown in Figure 4.

Using the triangular law of vector addition and the law of sines, the following equations can be derived:

$$\alpha = \cos^{-1}\left(\frac{x_3^2 + y_3^2 - L_{12}^2 - L_{23}^2}{2L_{12}L_{23}}\right), \tag{1}$$

$$\beta = \sin^{-1}\left(\frac{L_{23}\sin\alpha}{\sqrt{x_3^2 + y_3^2}}\right). \tag{2}$$

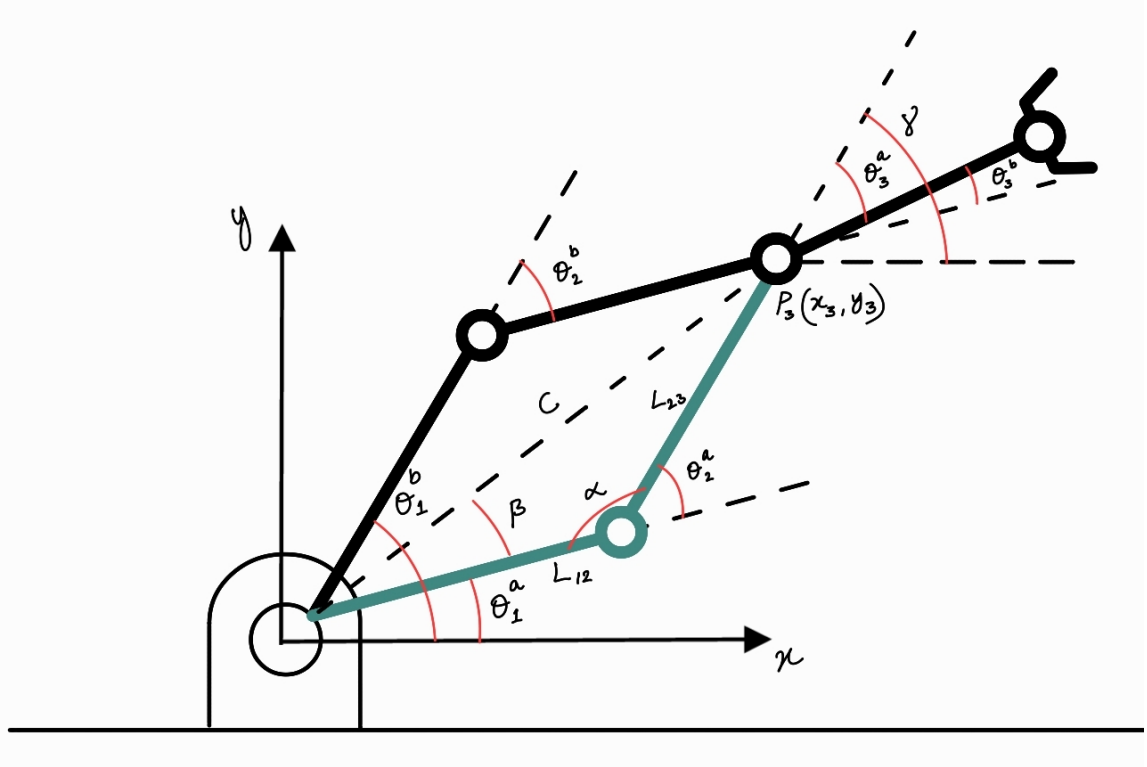


Fig. 4. 3R Planar manipulator diagram with joint angles and link lengths and the two possible orientations for a single position are differentiated using colors.

From the geometry in Figure 4, the position of point P_3 in Cartesian coordinates can be expressed as shown below [3].

$$P_3 = \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} = \begin{bmatrix} x_e - L_{34}\cos(\gamma) \\ y_e - L_{34}\sin(\gamma) \end{bmatrix}. \tag{3}$$

Using these relationships, the joint angles can be determined. For most configurations, the manipulator can assume two orientations: elbow-up (a) or elbow-down (b) [3]. For the purposes of this study, which focus on pick-and-place tasks, the elbow-down configuration is selected exclusively. The joint angles for this configuration are calculated as follows:

$$\theta_1^b = \tan^{-1}\left(\frac{y_3}{x_3}\right) + \beta, \tag{4}$$

$$\theta_2^b = -(180^\circ - \alpha), \tag{5}$$

$$\theta_3^b = \gamma - \theta_1^b - \theta_2^b. \tag{6}$$

2.4.2 IK of the Base Joint. The base joint angle, θ_{base}, can be determined using the coordinates of the end-effector (x_e, y_e). The angle is given by:

$$\theta_{\text{base}} = \tan^{-1}\left(\frac{y_e}{x_e}\right). \tag{7}$$

Equations (4), (5), (6), and (7) are integrated to compute all the joint angles required for the OpenMANIPULATOR-X manipulator.

2.4.3 Coordinate Transformation. To compute the joint angles, the end-effector coordinates must be transformed from the camera's base frame to the manipulator's base frame, and the pixel coordinates must be converted to meters. This transformation is performed using the following equations:

$$x_m = (y_p - 15) \cdot 0.000625, \quad y_m = (x_p - 435) \cdot 0.000625, \tag{8}$$

where x_m and y_m are the coordinates in meters, x_p and y_p are the pixel coordinates in the camera frame. The constants 15 and 435 represent the reference pixel coordinates, and the scaling factor 0.000625 converts pixel units to meters. With the transformed coordinates, the joint angles for any position within the manipulator's configuration space (C-space) are computed. Joint control is then performed, which provides faster and more accurate positioning compared to the `MoveIt!` package's default position control.

2.5 Text-to-Speech Model

To provide auditory feedback to the user, we utilized the Google Text-to-Speech (gTTS) service, which converts textual input into speech. The gTTS service supports over 100 languages and accents, offering flexibility for diverse applications. It also allows customization of speech speed through the `slow` and `speed_factor` parameters, enabling control over the pace of the generated speech.

In this study, the text-to-speech model is employed to deliver feedback such as "Speech Recognized!" upon successful speech recognition. Additionally, it provides error messages for scenarios where recognition fails, such as "Could not understand the audio. Please try again!" This integration ensures a user-friendly and interactive system, enhancing the overall experience.

3 Experimental Validation

3.1 Task Description

We test the working of our pipeline by placing the manipulator in three different scenarios:

- colored blocks of different shapes
- electrical lab environment
- mechanical lab environment

As mentioned earlier we give the manipulator natural voice commands and it assists you in the lab by completing the assigned task through manipulation (pick-and-place).

This video [2] (watch in 1.75x) shows the performance of the manipulator under different scenarios and different CV models.

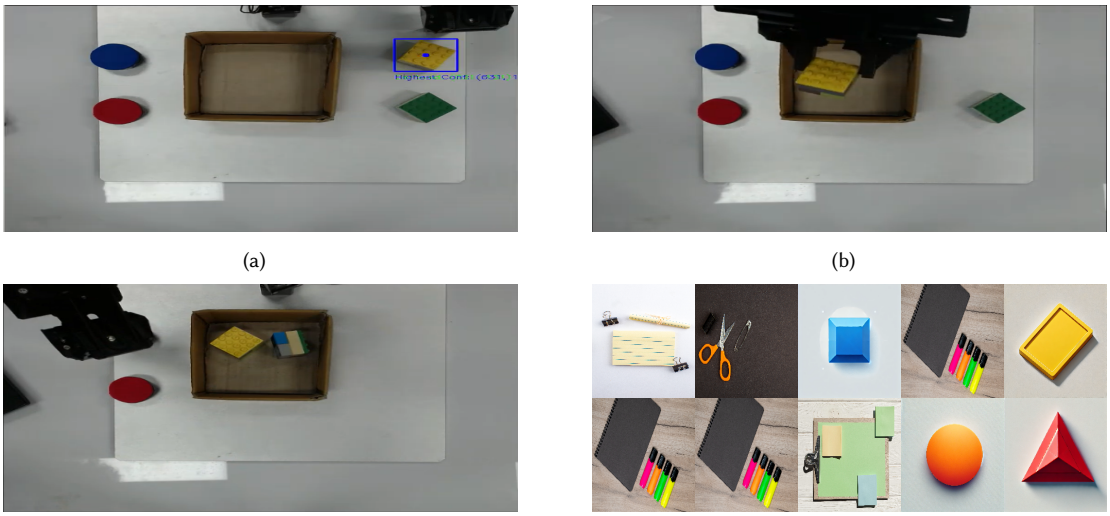
3.2 Dataset Used

It is also well recommended to perform benchmark tests on these CV models to get a clear idea on how well or worse it performs under these different scenarios. So, we create a sample dataset of 50 images, which consists of 20 images of electrical or electronic components and tools, 20 images of mechanical tools and 10 images of simple shapes of different colors and popular stationary. We feed the sample dataset to all of these Zero-shot object detection CV models.

The data accumulation of these CV models can be seen in through the Excel worsheet [1].

3.3 Experimental Results

3.3.1 Colored Blocks of Different Shapes. In the video [2] (watch in 1.75x) we ask the robot to place the yellow square object inside the cardboard box followed by asking it to place the green square object, the blue circular object and the red circular object inside the cardboard box. We clearly see it perform the manipulations seamlessly without any issues.



(a) (b) (c) (d)

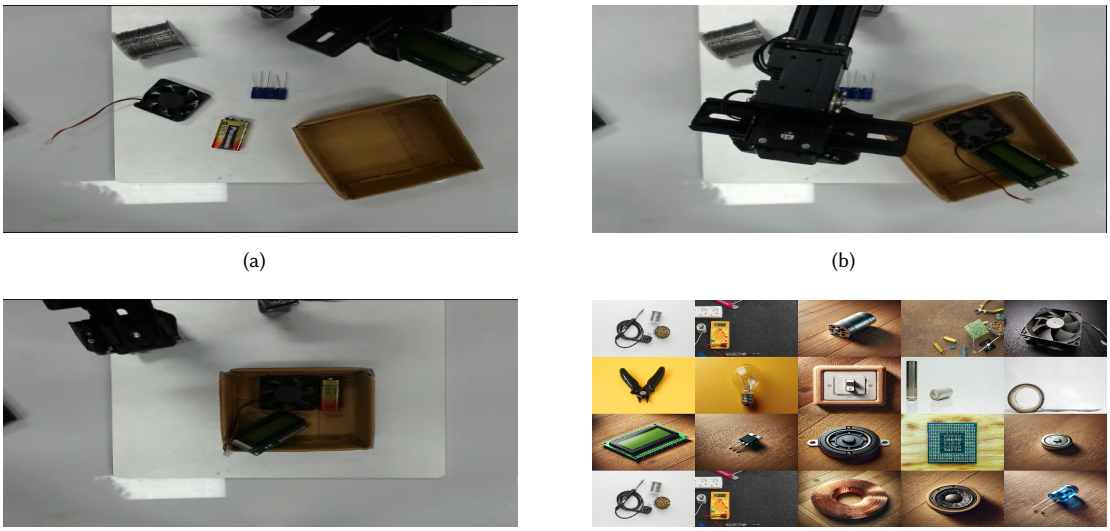
Fig. 5. (a) The Robot is searching for a yellow square. (b) The robot places the Yellow Square in the cardboard box. (c) The robot places the red square in the cardboard box. (d) A sample image dataset consisting of 20 images of shapes, colors, and stationery items used to benchmark the model, via free licensed images (https://www.freepik.com/), (https://www.vecteezy.com/), and (https://unsplash.com/).

Table 1. Testing Models on Shapes, Colours, and Stationery Items.

Models	Accuracy	Average Processing Time (s)	Mean Confidence (Correct Predictions)
Grounding DINO	70.00%	10.489	80.81%
Omdet Turbo	80.00%	2.347	68.15%
OwlV2	90.00%	15.247	60.43%
OwlVit	80.00%	2.157	34.18%

After testing the four models on different shapes and colors, *OwlV2* was the most accurate. *OwlVit* emerged to be the fastest model, and *Grounding DINO* proved to be the most confident model in predictions.

3.3.2 Electrical Lab Environment. In the video [2] (watch in 1.75x) we ask the robot to place all the electrical or electronic tools and components in a specific order as shown in the video, inside the cardboard box. We clearly see that it performs the manipulations seamlessly without any issues.



(a) (b) (c) (d)

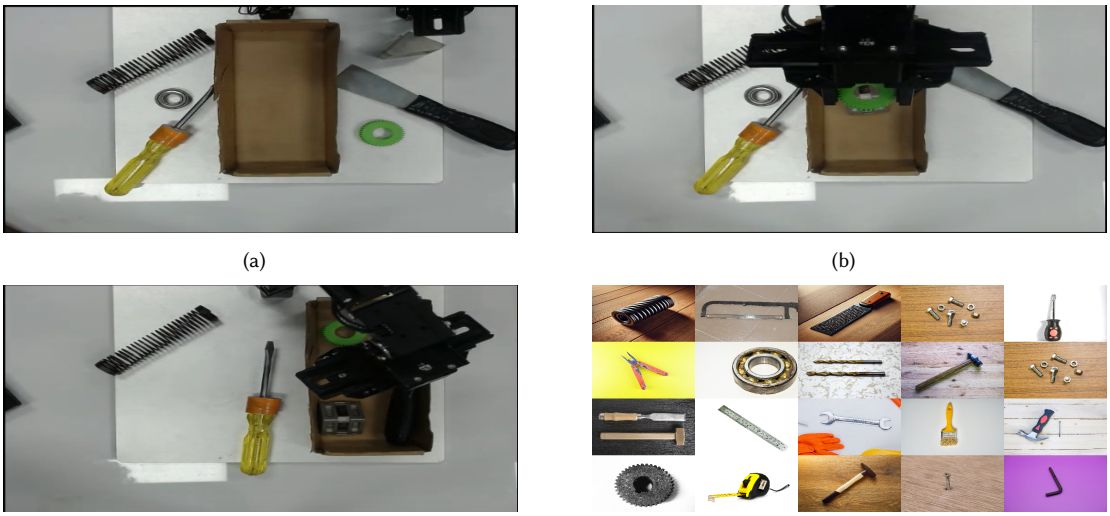
Fig. 6. (a) The Robot is searching for a 9V Battery, (b) The Robot is placing a Blue Capacitor in the Cardboard Box, (c) The Robot has successfully placed a green LCD display in the Cardboard Box, (d) A sample Image dataset consisting of 20 images of Various Electrical Tools and Components used to perform Benchmark Tests on the Computer Vision Models, via free licensed images (https://www.freepik.com/), (https://www.vecteezy.com/), and (https://unsplash.com/).

Table 2. Comparison of models for detecting Electrical Tools.

Models	Accuracy	Average Processing Time (s)	Mean Confidence (Correct Predictions)
Grounding DINO	70.00%	13.524	80.72%
Omdet Turbo	95.00%	2.158	68.10%
OwlV2	95.00%	17.008	49.70%
OwlVit	75.00%	1.145	28.73%

While testing on Electrical Tools, *Omdet Turbo* and *OwlV2* showed the highest accuracy. *OwlVit* was the most accurate model, and *Grounding DINO* was the most confident model for predictions.

3.3.3 Mechanical Lab Environment. In the video [2] (watch in 1.75x) we ask the robot to place all the mechanical tools in clockwise order as shown in the video, inside the cardboard box. We clearly see that it performs the manipulations seamlessly without any issues.



(a) (b) (c) (d)

Fig. 7. (a) The model detects a green gear to be placed in the cardboard box. (b) The robot has placed the green gear in the cardboard box. (c) The robot has placed the black chisel in the cardboard box. (d) A sample image dataset of 20 images of various mechanical tools used to benchmark computer vision models,via free licensed images (https://www.freepik.com/), (https://www.vecteezy.com/), and (https://unsplash.com/).

Table 3. Comparison of models to detect Mechanical Tools.

Models	Accuracy	Average Processing Time (s)	Mean Confidence (Correct Predictions)
Grounding DINO	80.00%	14.076	81.25%
Omdet Turbo	75.00%	2.372	60.34%
OwlV2	80.00%	14.883	47.04%
OwlVit	65.00%	1.175	28.44%

While testing over Mechanical Tools, *Grounding DINO* and *OwlV2* exhibited the maximum accuracy. *OwlV2* was the fastest model and *Grounding DINO* was the most confident model for predictions.

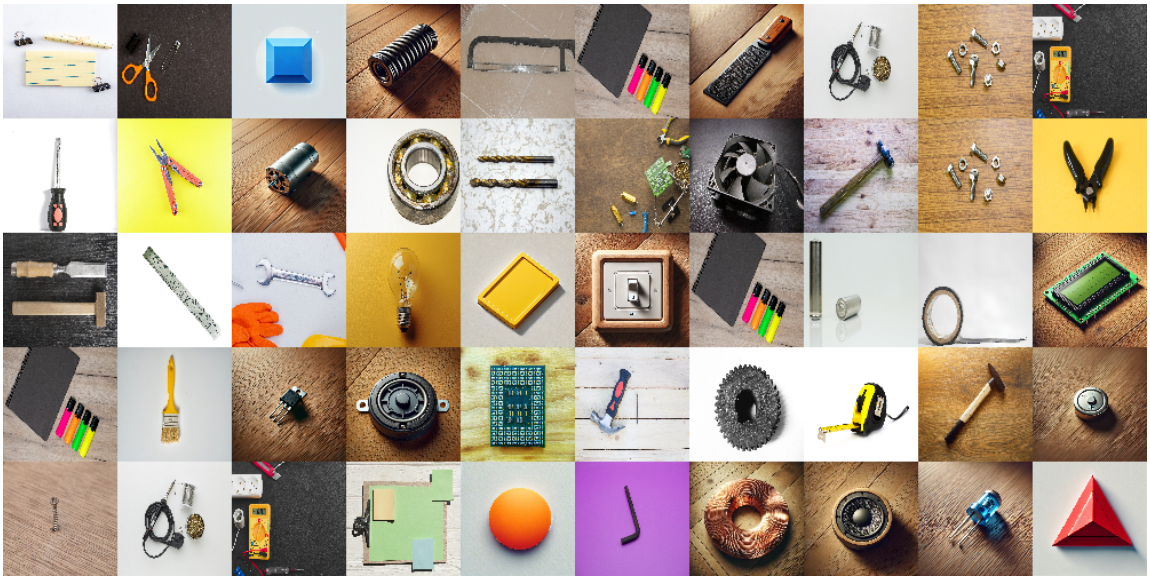


Fig. 8. A sample image dataset consisting of 50 images used to perform Benchmark Tests on the Computer Vision Models to judge the overall performance of the Models, via free licensed images (https://www.freepik.com/), (https://www.vecteezy.com/), and (https://unsplash.com/).

Table 4. Overall comparison of models.

Models	Accuracy	Average Processing Time (s)	Mean Confidence (Correct Predictions)	Confidence-Weighted Accuracy
Grounding DINO	74.00%	13.138	80.97%	59.92%
Omdet Turbo	84.00%	2.282	65.34%	54.88%
OwlV2	88.00%	15.806	50.93%	44.81%
OwlVit	72.00%	1.360	29.84%	21.48%

3.3.4 Overall Performance of Models. Overall, *OwlV2* was the most accurate model. *OwlVit* was the fastest model, and *Grounding Dino* was the most confident model.

For the highest accuracy, *OwlV2* is preferred, albeit at the cost of speed and confidence. This is because it uses a deeper and more complex architecture, which increases accuracy but reduces speed. For good confidence and accuracy, *Grounding DINO* excels, making it reliable but slightly slower. Its use of additional attention mechanisms or object-centric processing explains its high confidence but also incurs a speed penalty. The fastest model was *OwlVit*, but it consistently performed poorly in accuracy and confidence, making it ideal only for high-speed applications. This is due to its relatively lighter architecture. *Omdet Turbo* was a well-balanced model, as it uses a moderately heavy architecture, making it suitable for tasks that require a balance between speed and accuracy.

4 Why Modularity of CV Models is Important?

The modularity of computer vision (CV) models is crucial for advancing the development and application of Robot-LLM systems that integrate language models with visual perception. Modular CV models offer flexibility, scalability, and adaptability, which are essential in addressing the diverse challenges faced in robotics and multi-modal AI systems.

(1) Modularity facilitates customization and reuse. By designing CV systems as interchangeable components, developers can easily tailor specific modules for tasks like object detection, scene segmentation, or gesture recognition without rebuilding the entire pipeline. This approach enables researchers to reuse pre-trained modules in different contexts, reducing development time and computational costs.
(2) Modularity enhances scalability. Complex robotics applications often require incremental updates or additions to their vision capabilities. A modular architecture allows seamless integration of new functionalities. This scalability is critical in dynamic environments where robotic systems need to adapt to evolving requirements.
(3) Modularity supports fault isolation and debugging. In large-scale systems, identifying the source of errors can be challenging. Modular CV architectures enable developers to isolate and diagnose specific components, ensuring more efficient troubleshooting and maintenance.

If in future, we need to add a new, better zero-shot object detection CV model to our pipeline we just need to add or change only the "`process_frame`" function. Once, that is done we are good to go. This makes our implementation modular, which is crucial for all the reasons given above. Finally, modularity promotes collaboration across research domains. Teams can independently develop and improve modules, fostering innovation and rapid prototyping.

5 Challenges Faced

- Finding the right LLM which is simple to deploy and works everytime for the same repetitive instruction it has been given.
- Writing modules for every block in the pipeline (efficient IK, Modular CV, pick-and-place, speech recognition, user interface, text-to-speech) which would be finally imported in the UI code.
- Overlay of live webcam feed and the display of "`processed_frame`" together in the same window of the user interface.
- Devising a single common function to detect objects and return centroids for all of the zero-shot object detection models.
- Figuring out to how to divide the inverse kinematics problem among the parts of the OpenMANIPULATOR-X manipulator, to solve it efficiently and easily.

6 Future Scope and Discussion

The integration of robotics and Large Language Models (LLMs) presents significant potential, given that both fields are among the fastest-evolving domains in modern technology. The rapid adoption of systems like OpenAI's ChatGPT into everyday life underscores the transformative impact of LLMs. Extending this capability to interact effectively with the physical environment through robotic systems offers unparalleled opportunities for societal impact. A technology pipeline, such as the one proposed in this study, has the potential to bridge the gap between digital intelligence and real-world application, surpassing the current limitations of LLMs in generating textual content.

One of the most impactful applications of such a pipeline is in assistive technologies, particularly for individuals with disabilities. For instance, the world's blind population could benefit significantly from a system capable of interpreting

visual inputs and executing contextual physical tasks. Beyond the blind community, individuals with varying disabilities could use this cross-domain innovation to perform everyday activities more efficiently. Our experimentation, which employed a standard webcam and a 4-DOF manipulator, highlights the versatility of such systems. The addition of a depth-sensing camera, for example, could allow manipulators to perform precise tasks like placing tools directly into a user's hands.

The modular design of the pipeline ensures that it remains future-proof. Faster, more accurate CV or LLM modules can be seamlessly integrated without requiring a complete redesign. Another exciting avenue lies in industrial automation, where modular CV-LLM systems could optimize assembly lines by integrating complex decision-making capabilities with robotic dexterity. These advancements underscore the transformative potential of modular, cross-domain pipelines in addressing real-world challenges across industries.

7 Conclusion

The integration of robotics and Large Language Models (LLMs) with modular computer vision (CV) architectures represents a transformative step in advancing multi-modal AI systems. Our research highlights the importance of modularity in ensuring adaptability, scalability, and interoperability, allowing the pipeline to evolve with advancements in both CV and LLM technologies. Through experimentation with a simple setup, we demonstrated the pipeline's potential in real-world applications, from assistive technologies for individuals with disabilities to broader industrial automation tasks. The modular design ensures future readiness, enabling seamless integration of more accurate and efficient models to meet emerging challenges.

This innovative cross-domain approach bridges the gap between digital intelligence and physical interaction, paving the way for a wide range of use cases. Ultimately, this research underscores the immense potential of combining LLMs with robotics, offering a pathway toward smarter, more inclusive, and highly adaptable systems for diverse real-world environments.

References

[1] Anonymized. 2024. Data Accumulation of CV Models. Excel Sheet. https://docs.google.com/spreadsheets/d/1LGY4fcigky1658t3-OpXl4GrfP7dnuGN/edit?gid=1779260325 Accessed: December 15, 2024.

[2] Anonymized. 2024. Robot_LLM_Modular_CV_anonymous.mp4. Video. https://drive.google.com/file/d/1QHUNmvWTMWS5MtPXRnRxxIPxhEmTEuAL/view Accessed: December 15, 2024.

[3] Juecoree. 2020. Forward and Reverse Kinematics for 3R Planar Manipulator. https://hive.blog/hive-196387/@juecoree/forward-and-reverse-kinematics-for-3r-planar-manipulator Accessed: 2024-12-15.

[4] Allan Kouidri. 2023. Grounding DINO: Leading the Way in Zero-Shot Object Detection. https://www.ikomia.ai/blog/grounding-dino-zero-shot-detection-explained?form=MG0AV3 Accessed: 2024-12-15.

[5] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. 2023. Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection. arXiv:2303.05499 [cs.CV]

[6] Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. 2023. Scaling Open-Vocabulary Object Detection. arXiv:2306.09683 [cs.CV]

[7] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, Xiao Wang, Xiaohua Zhai, Thomas Kipf, and Neil Houlsby. 2022. Simple Open-Vocabulary Object Detection with Vision Transformers. *arXiv preprint arXiv:2205.06230* (2022).

[8] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024. Qwen2 Technical Report. *arXiv preprint arXiv:2407.10671* (2024).

[9] Tiancheng Zhao, Peng Liu, Xuan He, Lu Zhang, and Kyusong Lee. 2024. Real-time Transformer-based Open-Vocabulary Detection with Efficient Fusion Head. *arXiv preprint arXiv:2403.06892* (2024).

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009