Final Project Proposal

Planning AI Robot Arm Assistant for Tool Handling in Engineering Projects

Submitted to the
Project Committee appointed by the
International School of Engineering (ISE)
Faculty of Engineering, Chulalongkorn University

Project Advisor

Asst.Prof.Paulo Fernando Rocha Garcia, Ph.D.

Submitted By

Kanisorn Sangchai	6538020621
Methasit Boonpun	6538165021
Withawin Kraipetchara	6538191221
Krittin Kitjaruwannakul	6538007521

1/2025: 2147416 Final Project I
Robotics and Artificial Intelligence Engineering (International Program)
International School of Engineering (ISE) Faculty of Engineering,
Chualongkorn University

Abstract

Frequent tool switching in complex engineering projects disrupts workflows and reduces efficiency. Robotic assistants have been proposed as a solution for tool handling, yet many existing approaches rely heavily on end-to-end neural networks that lack interpretability and robustness in dynamic, unstructured environments. Recent research highlights the benefits of combining neural perception with symbolic reasoning, a paradigm known as neuro-symbolic artificial intelligence (NSAI). Togther with large language model (LLMs) and motion planning (TAMP) frameworks. These hybrid methods have shown function in enabling robots to understand natural language, reason about tasks, and execute reliable action in real-world scenario

The objective of this project is to develop a prototype of a Planning AI Robot Arm Assistant capable of supporting engineers task. The system aims to

1. interpret natural language voice commands
2. translate command into interpretable symbolic task plans
3. detect, hold, and deliver engineering tool safely
4. provide transparent reasoning process

By focusing on interpretability, safety, and adaptability, the project addresses current limitations of existing robotic assistants and advances the development of collaborative human-robot systems.

The methodology involves integrating neural and symbolic AI within a system architecture. Speech recognition and natural language processing are used to understand user commands. These inputs are processed by a neuro-symbolic planning algorithm that generates task sequences. The system will be implemented and tested on a Universal Robots (UR) arm equipped with a gripper, depth camera, and microphone. Simulations and real-world experiment will validate the prototype to generalize to unseen scenarios and safely collaborate with human users in engineering environment

Contents

1 Background **4**
- 1.1 Robot Manipulation and Task Planning 4
- 1.2 Large Language Models (LLMs) in Robot Manipulation 5
- 1.3 Neuro-Symbolic Artificial Intelligence (NSAI) in Robot Manipulation 5

2 Objectives **7**

3 Literature Survey and Review **10**
- 3.1 Existing Solutions for Robot Manipulation through Natural Language Instructions 10
  - 3.1.1 Voice Control and Multimodal Speech Recognition for Robot Manipulation 10
  - 3.1.2 Robot Planning 12
- 3.2 Existing Technology for Robot Manipulation through Natural Language Instructions 14
  - 3.2.1 Voice Control and Multimodal Speech Recognition for Robot Manipulation 14
  - 3.2.2 Robot Planning 15
- 3.3 Conclusion 18

4 Project Concept Development with Core Technology **20**
- 4.1 The Current Development of the Project 21
  - 4.1.1 Speech 21
  - 4.1.2 Visual 22
  - 4.1.3 Planning 25
  - 4.1.4 Simulation 25

5 Project Planning **28**
- 5.1 Image Processing 28
- 5.2 Speech Processing 29
- 5.3 Planning Algorithm 29
- 5.4 Interface with Robot Arm 30

6 Theory and Technical Backup **31**
- 6.1 Voice Control Systems (VCS) 31
- 6.2 Multimodal Automatic Speech Recognition 31

6.3 Neuro-Symbolic Artificial Intelligence (NSAI) 32
6.4 Large Language Model (LLM) 34
6.5 Hierarchical Reasoning Model (HRM) 35
6.6 Vision Language Model (VLM) 37
6.7 Vision-Language-Action (VLA) Models 38
6.8 Planning Domain Definition Language (PDDL) 39
6.9 ROSPlan 41
6.10 MoveIt2 43
6.11 Artificial Intelligence in Image Processing 43
6.12 Universal Robots Arm (UR ARM) 46

7 Project Outcome **48**
7.1 Autonomous Planning: 48
7.1.1 Path and Motion Safety 48
7.1.2 Generalization: 49
7.1.3 Human Collaboration: 49

8 Summary and Benefit to Real Industry **50**

9 Roles and Responsibilities **51**

1 Background

In engineering projects, practitioners often switch between multiple tools and instruments. This process disrupts workflow and can introduce inefficiencies or errors. Robotic assistants have the potential to support engineers by automating tool handling and delivery. Achieving this requires effective robotic manipulation and planning, which are typically implemented using deep learning-based, end-to-end neural networks—approaches that may struggle in novel situations and offer limited transparency. By combining neural perception, large language models, and symbolic reasoning, a more reliable and interpretable solution can be created for tool manipulation and delivery.

1.1 Robot Manipulation and Task Planning

Historically, robot task execution was divided into two paradigms: AI task planning and robotics motion planning. Task planning generated abstract sequences of discrete actions using symbolic frameworks such as STRIPS or PDDL, while motion planning computed collision-free paths in continuous configuration spaces. This division proved sufficient in structured factory settings, where tasks and motions could be predefined, but inadequate in unstructured, human-centric environments [1], [2]. Early systems, such as Shakey the Robot, assumed high-level plans could always be refined into feasible motions, an assumption that rarely holds in practice [3]. The core difficulty was that symbolic plans often ignored geometric and kinematic constraints, producing strategies that were logically valid but physically infeasible. Task and Motion Planning (TAMP) overcame these limitations by formulating robot planning as a hybrid discrete–continuous search problem. By combining symbolic reasoning with geometric feasibility checks and introducing an intermediate level for selecting real-valued parameters such as grasps or placements, TAMP effectively bridged high-level planning with low-level execution, employing strategies such as backtracking refinement or interleaved search to prune infeasible plans [1]. Beyond feasibility, optimization-based formulations yield efficient trajectories, while learning-based methods accelerate feasibility checks, guide sampling, and adapt to novel settings [2]. Crucially, TAMP allows robots to revise task strategies when geometric constraints block execution, enabling scalable planning in complex domains [3].

1.2 Large Language Models (LLMs) in Robot Manipulation

Large Language Models (LLMs), particularly when augmented with visual reasoning as Vision-Language Models (VLMs), are increasingly central to robot manipulation, navigation, and broader embodied AI tasks, transforming complex initial states into desired goal states through sophisticated planning [4], [5]. Beyond directly generating action sequences, LLMs also serve as Modelers, extracting and refining structured planning models such as Planning Domain Definition Language (PDDL) specifications from natural language, which mitigates challenges in long-horizon reasoning and enhances plan reliability [6]. Leveraging world knowledge, reasoning, and decision-making, LLMs facilitate Task Modeling by converting human goals into initial and goal states and decomposing complex tasks into sequential, parallel, or recursive sub-goals. In robot manipulation, LLMs/VLMs improve handling of novel objects and generate motor or programmatic actions for perception, planning, and execution [4]. Furthermore, their impact extends to human-robot interaction (HRI), enhancing intent interpretation and adaptability [5].

Functionally, LLMs enable Domain Modeling, defining essential components like actions, preconditions, and effects, sometimes through demonstrations or iterative refinement [6], and integrate with sensory data to ground language understanding in spatial contexts, ensuring executable and robust plans. Integration with classical planners, hierarchical planning, and search algorithms allows LLMs to translate abstract goals into reliable action sequences guided by world models. Closed-loop feedback systems further enable dynamic adaptation, reducing hallucinations and improving task execution, while fine-tuning on planning-specific objectives enhances correctness and generalization [4], [6].

1.3 Neuro-Symbolic Artificial Intelligence (NSAI) in Robot Manipulation

Neuro-Symbolic Artificial Intelligence (NSAI) is an approach in robot manipulation that combines neural networks' perceptual strengths with symbolic AI's reasoning capabilities to create interpretable, interactive, and generalizable robotic agents in human-centric environments [7], [8]. NSAI enables robots to follow unconstrained natural language instructions for tasks like object picking, grasping, and multi-step manipulation such as pick-and-place

or sorting [9]. Its architecture typically includes a hybrid scene encoder, neural language parser, reasoning/action primitives, symbolic program executor, and concept grounding modules. Language parsers translate instructions into executable symbolic programs, while scene encoders construct object-centric representations [7], [9]. Visual and spatial grounders match language concepts to objects, supporting generalization to unseen scenarios. Symbolic executors perform operations like filtering, querying, and executing manipulation actions [7], and simulators predict target locations for execution [9]. Training often uses weakly supervised, end-to-end curriculum learning with policy gradient methods [9], [10]. This modular design offers interpretability and enables interactive feedback. Challenges remain in integrating continuous neural models with discrete symbolic reasoning [8], with future directions focusing on large language models for zero-shot parsing, vision-language models for grounding, and more complex logical structures for advanced planning [7].

2 Objectives

Drawing from the prior examination of robotic manipulation through natural language command systems, there are multiple challenges and limitations, such as generalization, lack of interpretability, and handling complexity, that pose room for improvements. Thus, the project aims to tackle and explore these challenges and limitations by building a prototype of a personal AI assistant with a robotic arm that can:

1. Understand natural language voice commands

   A primary challenge is enabling robots to effectively ground abstract semantic concepts in precise spatial reasoning from natural language. Current end-to-end neural networks, while capable of learning dexterous skills, often fail to generalize to new goals or quickly learn transferable concepts across tasks when confronted with language that introduces new concepts or slight variations (eg, distinguishing between "red pens" and "blue pens" without extensive new training data) [7]. This highlights a core difficulty in how robots interpret the semantics underlying tasks from human instructions. Furthermore, past language-grounding methods for manipulation were often limited by object-centric representations and struggled to integrate perception and action cohesively based on linguistic input [11]. Overall, robots need to move beyond simple keyword recognition to a deeper, more generalizable understanding of human language in diverse, real-world contexts [7], [9].

2. Translate commands into symbolic task sequences

   Traditional Task and Motion Planning (TAMP) approaches specify tasks using formal representations like PDDL, but these require significant expertise and lack generalizability across different problems [12]. A promising solution involves neuro-symbolic AI, which combines the pattern recognition strengths of neural networks with the logical reasoning and structured knowledge of symbolic AI [8]. This hybrid approach allows for the explicit representation of the underlying reasoning process as symbolic programs, often using a Domain-Specific Language (DSL). Such disentanglement of perception (neural) from reasoning (symbolic) leads to systems that are more sample-efficient , can generalize to unseen concept-task combinations, and enable deeper, com-

positional, and hierarchical reasoning over abstract concepts derived from instructions [7]. Recent advancements has highlighted an alternative approach by exploring guiding LLMs to directly generate code that serves as the robot's TAMP planner and checker , integrating symbolic computation into the planning process while maintaining broad generalizability [12].

3. Detect, grasp, and deliver engineering tools safely to the user

   Accomplishing this requires overcoming challenges in precise spatial reasoning and robust interaction with objects in dynamic, unstructured environments. Current systems, even those showing promise in semantic understanding, may still struggle with fine-grained manipulation that demands high spatial precision, such as handling deformable objects or specific placements. Generalization to novel object instances that were not part of training data remains a hurdle, with models sometimes exploiting biases rather than truly grounding instructions [11]. The process involves object detection, instance segmentation, and grasp synthesis to identify and physically interact with items [7]. Furthermore, ensuring safety during physical interaction is paramount, requiring rigorous validation and addressing issues like collision avoidance and potential biases from pre-trained models that could lead to harmful actions. The "sim-to-real" gap also means that models trained in simulation may require substantial fine-tuning for robust real-world performance [7], [11].

4. Provide interpretable task plans that users can inspect, adjust, or correct

   This addresses the critical problem of "black box" AI, where purely neural systems lack transparency and the ability to explain their decisions. This opacity makes it challenging for non-expert users to diagnose and correct errors in a robot's behavior. Neuro-symbolic AI explicitly tackles this by disentangling reasoning from visual perception and language understanding, leading to fully transparent and interpretable reasoning processes. By generating explicit symbolic programs that represent the robot's plan as a sequence of logical steps, the system provides a formal, interpretable representation of its decision-making. This interpretability is crucial for human-in-the-loop interaction , allowing users

to understand why a robot performs certain actions, inspect the generated task plans, and provide targeted feedback or corrections when a failure occurs. The ability to communicate about failures and ambiguities in a dialogue setting significantly enhances usability and trust in autonomous systems [7], [8].

3 Literature Survey and Review

3.1 Existing Solutions for Robot Manipulation through Natural Language Instructions

3.1.1 Voice Control and Multimodal Speech Recognition for Robot Manipulation

Voice-based interaction has played a central role in enabling natural communication between humans and robots. Two key approaches that have been widely studied are **Voice Control Systems (VCS)** and **Automatic Speech Recognition (ASR)**, both of which contribute to making robot manipulation more intuitive and user-friendly.

Traditional Voice Control Systems (VCS). VCS applications have demonstrated the potential of speech as a hands-free, natural interface in multiple domains [13]:

- **Early Childhood Education:** Speech-activated robotics systems enable children to communicate with robots through simple voice commands, fostering intuitive interaction.
- **Smart Home Automation:** By integrating IoT, Fog, and Cloud architectures with speech-to-text and text-to-speech systems, users can control lights and appliances through voice interfaces.
- **Assistance for Individuals with Disabilities:** Low-cost voice-based home automation solutions allow people with paraplegia or quadriplegia to control adjustable beds or other devices with simple commands.
- **Healthcare Applications:** Voice recognition on mobile devices helps medical staff record workflow milestones efficiently, reducing the burden of manual data entry.
- **Traffic Management:** Voice-controlled systems can adjust traffic signals based on simple commands such as "red," "green," or "yellow."

Despite their wide use, traditional VCS are typically limited to *predefined tasks*. They struggle with grounding fine-grained concepts (e.g., color, material, or spatial relations), algorithmic reasoning (e.g., counting, comparing), and often function as black-box systems that require large datasets for

training and lack interpretability [13]. Security and data privacy also remain ongoing concerns.

Automatic Speech Recognition (ASR) and Multimodal ASR. To overcome these limitations, modern systems have moved toward **Automatic Speech Recognition (ASR)**, particularly **multimodal ASR**, which integrates both speech and visual context to improve accuracy in real-world, noisy environments [14]. This approach is better aligned with the conditions faced by physical robots that must interpret spoken rather than written instructions.

Key purposes and mechanisms of multimodal ASR include:

- **Addressing Input Gaps:** While many benchmarks assume text-based instructions, robots deployed in practice must process spoken commands. Multimodal ASR bridges this gap by converting speech into text [14].
- **Reducing Recognition Errors:** Unlike unimodal ASR, which may misinterpret unclear words, multimodal ASR leverages accompanying visual information to resolve ambiguities [14].
- **Example Scenario:** If a robot hears "Put the egg in the ..." but the final word is noisy, unimodal ASR might incorrectly guess "fridge." Multimodal ASR, however, can use visual context (e.g., spotting a microwave nearby) to correctly infer "microwave" [14].

Architecture of Multimodal ASR:

- **Speech Encoding:** Spoken instructions are processed by a speech encoder (e.g., wav2vec 2.0) to extract meaningful features [14].
- **Visual Encoding:** The robot's current visual observation is processed by a visual encoder (e.g., CLIP-ViT) into a feature vector [14].
- **Joint Decoding:** A Transformer-based decoder attends to both the speech and visual encodings, combining them to predict an accurate textual transcription of the spoken command [14].

Together, VCS and multimodal ASR represent complementary solutions for enabling natural language robot manipulation. VCS laid the foundation for voice-driven interfaces in defined contexts, while multimodal ASR extends these capabilities to *open, context-rich environments*, where accurate and adaptive interpretation of spoken instructions is crucial [13], [14].

3.1.2 Robot Planning

CLIPORT
CLIPORT solves various language-specified tabletop tasks, including packing unseen objects and folding clothes. It supports precise manipulation, abstract reasoning, data-efficient few-shot learning, and generalization to seen and unseen concepts. A single multi-task policy handles 10 simulated and 9 real-world tasks, transferring attributes like "pink block" without explicit training. It grounds categories, shapes, and colors without bounding boxes, is less prone to overfitting, and has been validated on a real Franka Panda manipulator. However, it requires an expert to signal task completion, struggles with complex spatial relations, cannot count or maintain history, and grounds verb-noun phrases only within trained contexts. It lacks dexterous 6-DOF manipulation, continuous control, and robustness in partially observable scenes. Execution depends on accurate hand-eye calibration, with open-loop pick-and-place prone to failure if objects move. Grasping novel objects outside training distribution is limited, and biases from CLIP training data may introduce vulnerabilities [11].

Code as Policies (CaP)
CaP translates natural language commands into robot policy code, expressing functions, feedback loops, and parameterized control primitives. It supports spatial-geometric reasoning, precise value assignment, logic structures, and use of external libraries. It generates hierarchical, interpretable code without extra training, adapts to unseen instructions, and outperforms LLM-only reasoning on spatial tasks. CaP handles multilingual commands, emojis, and cross-embodiment tasks, demonstrated on shape drawing, pick-and-place, and mobile robots. Limitations include dependence on available perception APIs and control primitives, difficulty with long or abstract instructions, inability to ensure feasibility or correctness, and brittle cross-embodiment performance. Duplicate object identity may be lost, and complex continuous control can require manual tuning [15].

Code-as-Symbolic-Planner
This method generates code serving as a Task and Motion Planning (TAMP) planner, integrating symbolic computation for efficient plan search and reasoning. It generalizes across discrete/continuous TAMP tasks, simulations, and real-world settings, outperforming other LLM-based planners and Code

Interpreter. It incorporates commonsense reasoning, uses multi-round refinement, and supports multi-robot collaboration. Limitations include inconsistent initial code, suboptimal or time-exceeding solutions for complex tasks, risk of coding errors, assumptions of full environment knowledge, and reliance on manually defined functions for goal translation [12].

Neuro-symbolic Approach
Neuro-symbolic approaches in robot manipulation combine the strengths of neural networks for perception and language understanding with symbolic reasoning for task planning, enabling robots to interpret and execute unconstrained natural language commands for diverse and complex tasks such as grasping, pick-and-place, and multi-step assembly. A significant advantage is their interpretability, offering transparent reasoning processes that help users diagnose and correct robot errors through interactive feedback, or by reconstructing anticipated scene states. These systems demonstrate robust generalization capabilities, efficiently adapting to novel scenes, previously unseen vocabulary, varying object counts, and longer, more intricate multi-step instructions. This often requires minimal supervision, learning from only initial and final world states or through few-shot adaptation for new concepts and real-world transfer. However, a current limitation is their typical reliance on tasks decomposable into sequential primitive steps, with more complex logical structures like conditionals or loops posing a challenge for existing domain-specific languages. [7], [9].

HyCodePolicy
HyCodePolicy interprets high-level natural language instructions for robot manipulation, decomposing them into hierarchical subgoals and synthesizing executable Python programs grounded in object-centric geometric primitives. Programs are executed in simulation with a vision-language model monitoring checkpoints to identify, localize, and infer causes of failures. The system pinpoints root causes, applies targeted code repairs using execution logs and perceptual feedback, and enables self-correcting program synthesis with minimal human intervention. Generated code is treated as an evolving hypothesis, validated and corrected iteratively, enhancing robustness and sample efficiency. HyCodePolicy demonstrates strong zero-shot generalization across structured placement, stacking, and planar manipulation tasks. However, it struggles with articulated object manipulation, fine-grained parameter adjustments, deformable object dynamics, and tasks requiring pre-

cise arm poses, granular spatial understanding, or complex temporal reasoning [16].

LLM-use-API (AuDeRe)
LLM-use-API (AuDeRe) selects appropriate motion planning and control algorithms from natural language task descriptions, environmental constraints, and robot dynamics. Rather than generating trajectories or code directly, it invokes comprehensive planning and control APIs tailored to these insights. It employs iterative LLM-based reasoning with performance feedback to refine algorithm selection and execution, generating high-level execution plans that describe API interactions, input/output relations, and data compatibility. This approach enhances autonomy, reduces manual tuning and expert intervention, generalizes across tasks from tracking to spatiotemporal planning, and consistently outperforms direct trajectory or code generation in success rates, query efficiency, and error reduction. Limitations include the non-triviality of direct API invocation, which requires integration code for compatibility. Performance degrades with higher LLM temperatures, larger models increase runtimes, and errors such as syntax, timeouts, and task failures can still occur despite reductions [17].

3.2 Existing Technology for Robot Manipulation through Natural Language Instructions

3.2.1 Voice Control and Multimodal Speech Recognition for Robot Manipulation

Voice Control Systems (VCS) have transformed human-computer interaction by enabling intuitive, hands-free communication through natural language commands. Traditionally, VCS relies on Automatic Speech Recognition (ASR) to transcribe speech into text and Natural Language Processing (NLP) to interpret and execute instructions. These systems are widely used in digital assistants, smart homes, and IoT devices, enhancing accessibility, convenience, and efficiency in everyday tasks. In robotics, VCS lowers barriers to interaction by allowing users, including children and individuals with limited mobility, to control robots naturally through speech [13]. Despite these advantages, traditional ASR systems remain limited in noisy, ambiguous, or dynamic environments. They often struggle with fine-grained contextual information or visually grounded references, which are critical for

successful robot manipulation. To address these challenges, multimodal ASR has emerged as an extension of VCS, incorporating both speech signals and visual context to ground linguistic input in the robot's perceptual environment [14], [18].

Core Architecture of Multimodal ASR:

- **Speech Encoder:** Extracts audio features from spoken instructions, often using pre-trained models such as wav2vec 2.0.
- **Visual Encoder:** Processes visual observations from the robot's environment (e.g., CLIP-ViT).
- **Language Decoder:** A Transformer-based decoder that jointly attends to both speech and visual features to generate accurate transcriptions.

By combining speech with visual context, multimodal ASR achieves higher transcription accuracy, especially for visually salient words such as object names or spatial instructions. It also improves task success rates by reducing errors from misheard commands and generalizes better across new environments and diverse speakers. Benchmarks such as ALFRED have demonstrated that multimodal ASR enables robots to more reliably follow natural language instructions in simulated household environments [13], [14], [18]. By uniting the accessibility of traditional VCS with the robustness of multimodal ASR, robots gain the ability to interpret and execute commands more effectively in uncertain and dynamic conditions. This evolution marks a critical step toward human-robot interaction that is both natural and contextually grounded [13], [14], [18].

3.2.2 Robot Planning

Robot manipulation tasks, particularly in unstructured human environments, necessitate that robots can understand and execute natural language instructions from non-expert users [11]. Historically, robot planning relied on segregated high-level artificial intelligence (AI) task planning and low-level motion planning, with traditional methods often using hand-coded symbols or relying on strict hierarchical decompositions [9], [12]. However, these approaches struggled to effectively combine discrete task decisions with continuous geometric and kinematic considerations, limiting their applicability in dynamic,

human-centric settings [11], [12]. Recent advancements have led to diverse implementations and approaches, broadly categorized as:

I. **Traditional / Early Language Grounding**
Traditional methods typically map natural language phrases to predefined symbolic representations of robot states and actions, assuming a symbolic description of the environment and actions. They often rely on symbols hand-coded by domain experts [2] or logical parses that translate language into motion constraints or control actions, and generally lack the flexibility to autonomously learn task semantics [9]. While these methods provide clear symbolic interpretability, they are inherently limited in their generalizability—new goals (e.g., switching from red pens to blue pens) require new training or explicit user input [10].

Interpretability is high since the reasoning is explicit and symbolic, but they often fail to integrate discrete task planning with continuous motion feasibility [1], restricting use in dynamic, real-world settings. Data efficiency is poor, as pre-annotated datasets are needed to map phrases to symbols. Additionally, scalability is limited since symbolic operators are predefined, and the frameworks cannot easily adapt to unseen tasks.

II. **Language-Conditioned End-to-End Systems (e.g., CLIPORT)**
These systems combine large-scale pre-trained vision-language models (like CLIP, for broad semantic grounding—the "what") with architectures specialized for spatial precision (like Transporter—the "where"). Transporter Networks are highly sample-efficient, end-to-end models for robotic manipulation that avoid object-centric assumptions, exploiting spatial symmetries and generalizing well to unseen objects and configurations [19]. CLIPORT builds on this by integrating language-conditioned policies and semantic understanding via a two-stream design, enabling grounding of categories, shapes, colors, and text attributes without requiring hand-designed perception pipelines. They show strong data efficiency, can learn multi-task policies, and generalize across seen and unseen semantic concepts, transferring learned attributes to new tasks [11].

However, challenges remain. CLIPORT struggles with fine-grained reasoning about relationships (e.g., "middle square hole" with unseen

shapes), counting, or verb-noun generalization beyond training data. Its execution is limited to open-loop pick-and-place primitives, making it brittle in dynamic or dexterous tasks. Interpretability is partial: while the semantic prior is clear, the underlying reasoning process remains mostly opaque. Calibration errors or biases in training data can degrade performance, and extending to high-DOF manipulation is non-trivial [11].

III. **Neuro-Symbolic Approaches**

Neuro-symbolic methods integrate neural perception modules with symbolic reasoning engines. They often translate natural language into executable symbolic programs composed of both neural (e.g., visual grounding) and symbolic (e.g., counting, logic) primitives. This enables them to combine robust perception with transparent, structured reasoning. Examples include the DeepSym framework for autonomously discovering symbols, neurosymbolic architectures for coupling vision and reasoning, and program-generating models for manipulation [8].

NSAI systems enhance interpretability, robustness, and trustworthiness, while also facilitating learning from less data. By disentangling perception and language understanding (neural) from reasoning (symbolic), neurosymbolic systems offer a formal, interpretable representation of the underlying reasoning process [7]. They are also highly sample-efficient, capable of learning from weak or "natural" supervision (initial/final states rather than dense annotations) [8], [9].

However, limitations include reliance on fixed domain-specific languages (limiting concept coverage), susceptibility to perception errors in cluttered scenes, and difficulty scaling to more complex logic (loops, conditionals) [7], [9]. Integration of neural and symbolic modules introduces computational challenges such as state-space explosion and longer training times, and standardization across frameworks remains an open problem [8].

IV. **Large Language Model (LLM)-Based Approaches**

LLM-based methods leverage the broad commonsense reasoning of large pre-trained language models for robot planning. Some approaches map natural language into symbolic planning representations (e.g., Text2Motion [20]), while others use LLMs to choose from pre-defined

primitives (e.g., Code-as-Policies [15]). A recent paradigm, Code-as-Symbolic-Planner, queries the LLM to generate executable code that acts as both the planner and verifier, explicitly integrating symbolic computation [12].

Their generalization is strong due to pretraining on vast text/code datasets. LLMs provide natural language transparency and can even generate interpretable code that reflects explicit reasoning chains [12], [15]. Data efficiency benefits from pretraining, though TAMP-specific fine-tuning or self-checking guidance frameworks are often required [4].

However, challenges include brittleness in complex optimization scenarios, inconsistency in generated code, and performance degradation with increasing task complexity. Earlier LLM methods that lacked symbolic computation were unreliable for tasks with numeric constraints [12]. Even with code generation, outputs may be suboptimal or erroneous, requiring iterative refinement. Furthermore, real-world translation of problems into solvable code remains difficult, and generated code can introduce execution risks [4].

3.3 Conclusion

The field of robot manipulation and planning through natural language is moving towards the main goal of enabling robots to seamlessly translate human intent into precise, executable actions. Current progress demonstrates the promise of LLMs and MLLMs in bridging abstract linguistic instructions with the concrete demands of real-world robotics. By combining high-level semantic understanding with fine-grained spatial reasoning, generating interpretable code as flexible policies, and leveraging neuro-symbolic reasoning for systematic generalization, researchers are building the foundations for robots that can adapt to diverse tasks and environments. These approaches, when coupled with modular design and iterative feedback loops, highlight a path toward truly intelligent and collaborative robotic assistants.

For this project, we aim to contribute to this progress by addressing some of the key limitations in the field. In particular, we will explore extending manipulation capabilities toward more complex object handling based on high-level commands, while also investigating methods for improving robustness and adaptability under real-world uncertainty. We see opportunities in integrating neural and symbolic reasoning, advancing perception systems

for richer grounding, and experimenting with human-in-the-loop strategies for preference alignment and safe deployment. By targeting these areas, We hope to push the field closer to building robots that are not only capable and efficient but also trustworthy collaborators in human-centered environments.

4 Project Concept Development with Core Technology

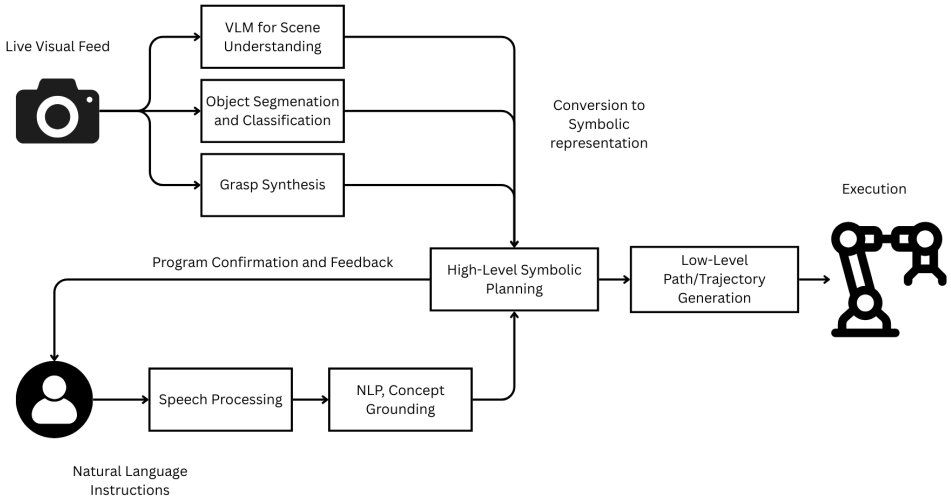

Figure 1: System Architecture Overview

The project combines:

- **Neural AI**: Object detection, segmentation, grasp synthesis, automatic speech recognition, and natural language understanding.
- **Symbolic AI**: Constraints enforcement to ensure executability. Path and trajectory generation.
- **Robotics**: A robot arm with a gripper executing precise manipulation under safe handover protocols.

This neuro-symbolic hybrid design ensures adaptability, efficiency, and user transparency.

Overall Workflow:

1. Users instruct the robot through high level, natural language, vocal commands.

2. The vocal command is converted into text using Automatic Speech Recognition (ASR) pipeline.

3. Semantics and discrete concepts are extracted from the text commands using Natural Language Processing (NLP) techniques, resulting a detailed description (for LLMs).

4. The current state of the work area is perceived through a camera.

5. The live visual feed is processed by applying object detection, segmentation, grasp synthesis, and visual understanding.

6. Visual and textual information are converted into symbolic representation by utilizing LLMs with access to visual perception tools and query users for additional details.

7. The symbolic high-level planner decomposes the high-level task into sequential sub-tasks.

8. The user can confirm, reject, or adjust the program proposed by the algorithm. Creating a closed-loop feedback system that allows more control.

9. Once a plan is confirmed by the user, each sub-task is executed.

4.1 The Current Development of the Project

4.1.1 Speech

To establish the Automatic Speech Recognition (ASR) foundation for robot manipulation, a speech-to-text pipeline was developed using the AssemblyAI API. In this prototype, example audio inputs such as harvard.wav were uploaded to a secure cloud endpoint, processed by AssemblyAI's transcription service, and retrieved once the transcription was complete. This workflow demonstrates how spoken instructions can be reliably converted into machine-readable text, which can subsequently be interpreted by Natural Language Processing (NLP) modules. By providing a robust and modular ASR layer, the system lays the groundwork for future integration with multimodal approaches, where linguistic input is contextualized with visual information from the robot's environment. Building upon this foundation,

the next stage of development will focus on extending the system into real-time ASR, enabling immediate transcription of speech and thereby allowing more natural, responsive interaction between humans and robots.

Figure 2: Example of the vcs pipeline using AssemblyAI with sample input ("harvard.wav")

4.1.2 Visual

We decided to implements modular vision pipeline consisting of four sequential stages: object detection and segmentation, object classification, grasp synthesis, and scene understanding. Each stage is designed to progressively enrich the robot's understanding of its environments, with the ultimate goal of enabling reliable manipulation and task executive.

Our exploration has focused on leveraging foundation models such as the Segment Anything Model (SAM) for segmentation and CLIPs for classification, while ensuring modularity that allows integration with grasping algorithm (e.g., GraspNet) and higher-level reasoning (e.g. Vision-Language Models for scene understanding) [21–24].

In this time period, we implemented and tested the first two stages of our vision pipeline: object segmentation and object classification

Object Segmentation

We used the SAM model to segment objects in tool-rich images. SAM successfully generated fine-grained masks and bounding boxes, producing over 50 detected regions per image as show in Figure 3.

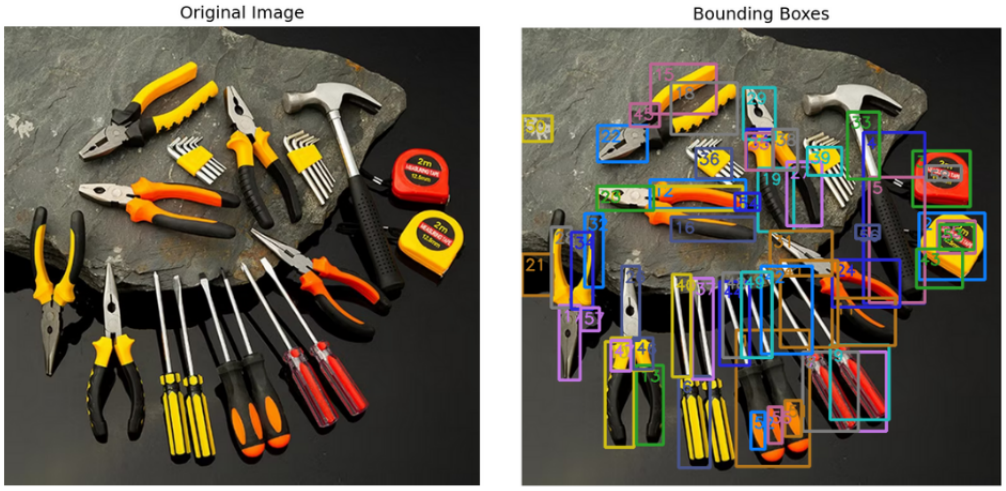

Figure 3: Segmentation results using the Segment Anything Model (SAM). Elongated objects (e.g., pliers) are often over-segmented into multiple masks

However, elongated objects such as pliers were often split into multiple masks. To address this, we implemented a mask merging strategy based on bounding box overlap and mask adjacency. This reduced redundant detections and allowed us to consolidate fragmented masks into a single object representation.

Object Classification

We integrated CLIP embeddings with a curated tool embedding set covering 15 tool categories. After applying mask refinement, the cropped object regions fed into CLIP showed improved consistency. For example, screwdriver and pliers that were previously classified as multiple partial objects now produced stable predictions with higher confidence scores.

By combining SAM segmentation with our refinement step and CLIP classification, we achieved more reliable object labeling across the dataset. Classification accuracy improved in terms of both stability (fewer duplicate predictions) and confidence values. We will extend the refinement module with embedding similarity checks to further improve mask consolidation. This will prepare the data flow for the third stage of the pipeline which is grasp synthesis.

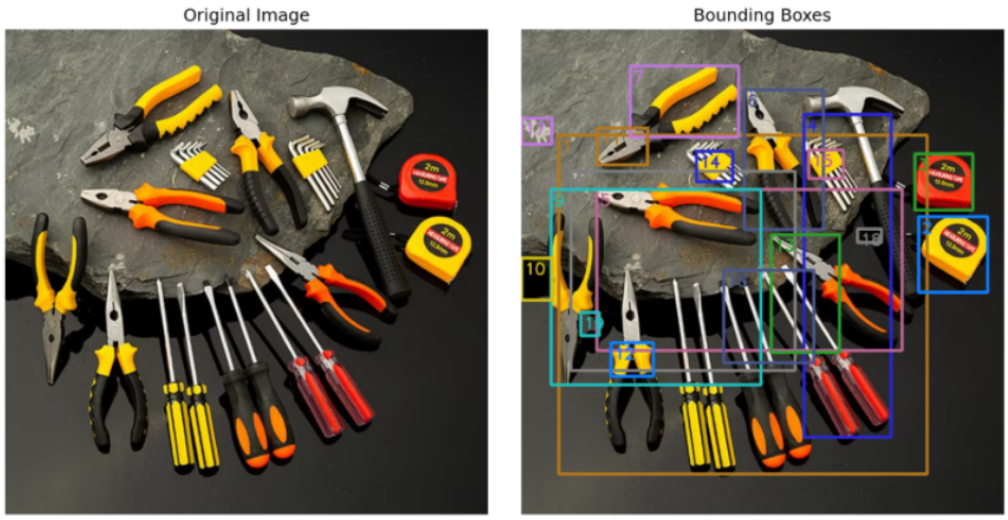

Figure 4: Segmentation results after applying the mask merging strategy. Over-segmented masks are combined into a unified representation

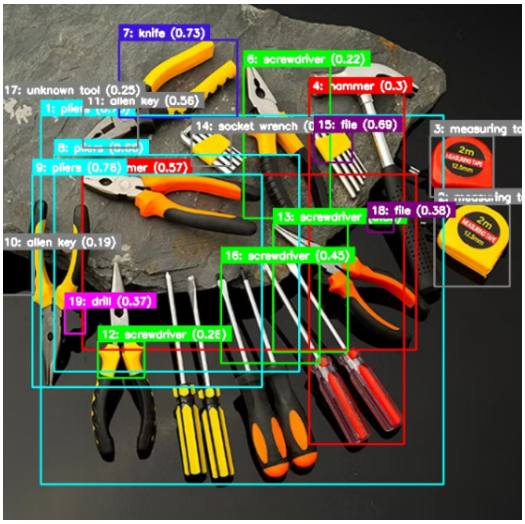

Figure 5: Object classification results using CLIP model

4.1.3 Planning

Symbolic Representation
We decided to use PDDL as our symbolic representation for our problem. Planning Domain Definition Language (PDDL), which is discussed in further details in Section 6.8 is a standardized language for modeling automated planning problems in artificial intelligence. Our exploration has focused on using large language models (LLMs) and multimodal language models (MLLMs) to automatically generate PDDL code from high-level task inputs. The idea is to reduce the gap between human task descriptions (in natural language or images) and the formal specifications required by planners.

In this approach, We designed a workflow where:

- Environment images are processed by an MLLM to extract structured details (e.g., objects, spatial relations, constraints), which are then translated into the domain model of PDDL.
- Textual task instructions are provided to the LLM, which generates the corresponding goal conditions and relevant actions.
- A LangChain-based agent orchestrates the process, giving the LLM access to multiple tools: one for querying the image model for clarification of environment details, and another for interactively querying the user to resolve ambiguities in the textual instructions.

Once the PDDL code is generated, We use the Fast Downward planner to find the solution for the defined problem.

Through this exploration, We have begun to identify the challenges of automating PDDL generation. These include ensuring syntactic correctness, capturing accurate action preconditions and effects, and handling underspecified or ambiguous task instructions. At the same time, the results demonstrate the potential of combining LLM reasoning, multimodal perception, and interactive tool use to streamline the process of translating high-level descriptions into structured planning problems.

4.1.4 Simulation

The simulation environment was set up on a shared computer located in room 9019-1, 9th floor, Chulapat 14 Building, as shown in Figure 6. This room also houses the UR robotic arm that will be used for physical experiments

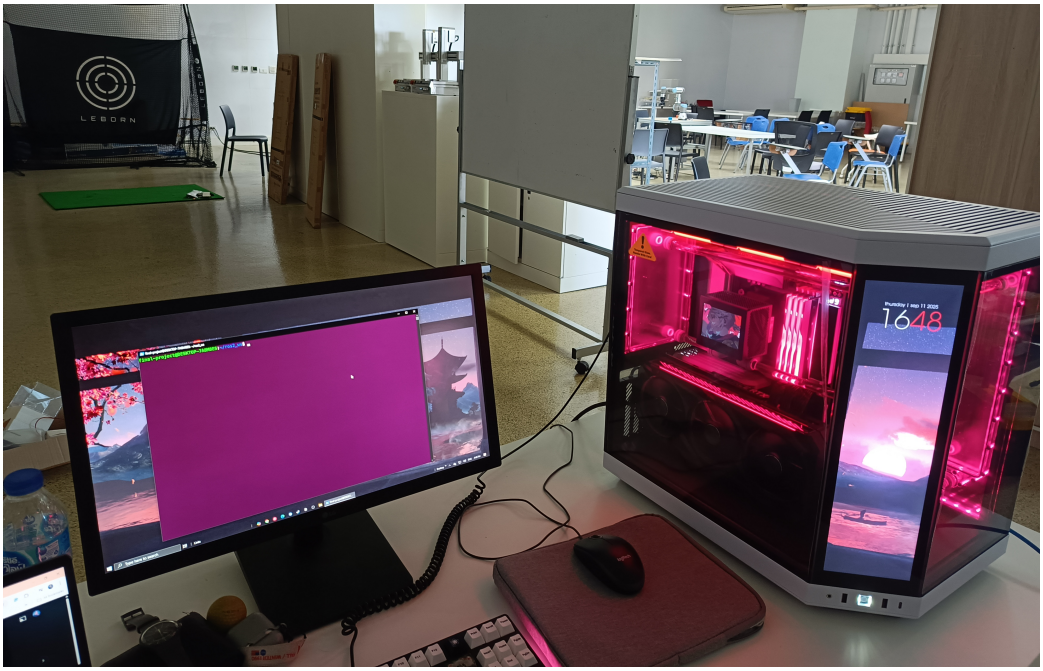

Figure 6: Shared Computer Setup at Chulapat 14

in the following semester. For the current semester, however, the focus is entirely on simulation.

To simulate robot control, we installed Isaac Sim on the shared computer. Since the long-term plan involves controlling the UR arm through ROS2, we configured the Isaac Sim ROS2 bridge, which allows the simulated robot to be controlled via ROS2 nodes. The simulation stack was deployed in a hybrid environment: Isaac Sim running on Windows, while ROS2 Humble was installed and run inside Ubuntu 22.04 on WSL2 on the same machine.

Isaac Sim provides built-in models for the UR3e robotic arm and the Robotiq gripper, as shown in Figure 7, both of which are the intended hardware components for future deployment. To maintain consistency between simulation and hardware, we also set up the official Universal Robots ROS2 driver [25]. The driver was successfully installed and tested; however, its documentation currently focuses on integration with URSim, the manufacturer's own simulator, rather than with Isaac Sim. As a result, integrating the ROS2 driver with Isaac Sim presents additional challenges that we are still investigating.

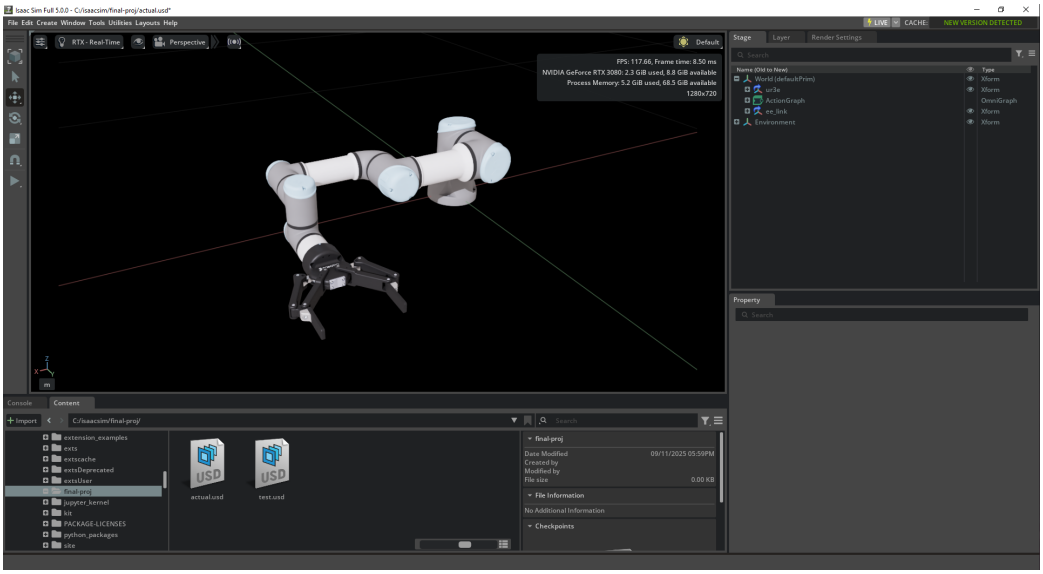

Figure 7: UR ARM 3e with Robotiq Gripper in Isaac Sim

In summary, the simulation environment is now operational with Isaac Sim and the ROS2 bridge, providing a foundation for testing planning and control pipelines in a simulated setting before moving to physical hardware next semester. The next milestone will be establishing integration between the official UR ROS2 driver and Isaac Sim, which will ensure compatibility between the simulated workflows and the physical UR arm.

5 Project Planning

The project is structured around four primary technical area of concentration exluding any relevant paperwork tasks. These align seamlessly with the core enhancements, which will all be implemented in parallel: **image processing, speech processing, planning algorithms, and robotic arm integration**. The planned timeline is structured into weekly milestones to ensure iterative progress and early validation

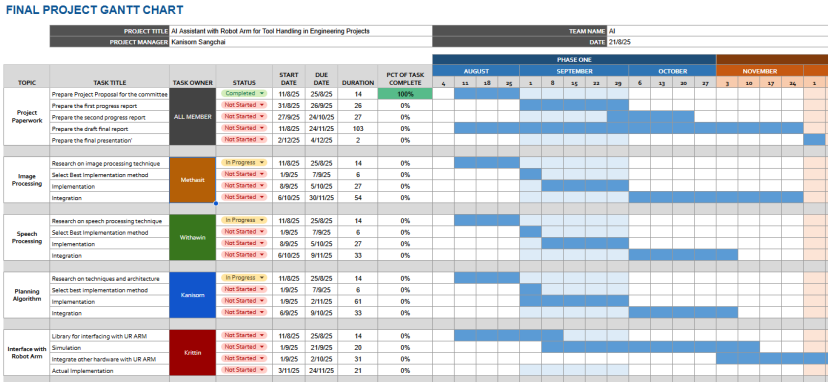

Figure 8: A proposed project timeline scheduled for the project's implementation in this semester

5.1 Image Processing

[2 Weeks] Research on image processing techniques

- A literature review and exploration of existing approaches will be conducted to identify suitable image processing methods for robotic applications.

[1 Week] Select best implementation method

- Based on the research findings, the team will decide on the most promising method to pursue further development.

[4 Weeks] Implementation

- The chosen method will be implemented, with iterative testing to verify accuracy, robustness, and suitability for integration with the robot system.

[8 Weeks] Integration

- The implemented solution will be connected to the overall system, ensuring compatibility and real-time performance with the robotic platform.

5.2 Speech Processing

[3 Weeks] Research on speech processing techniques

- A comprehensive study on speech-to-command systems will be carried out to identify feasible methods for tool-handling tasks.

[1 Week] Select best implementation method

- The most effective speech processing model or framework will be selected after evaluating feasibility, latency, and accuracy.

[5 Weeks] Implementation

- The selected method will be developed into a working module, with error handling and adaptability to diverse inputs.

[5 Weeks] Integration

- The module will be integrated into the robot system to ensure smooth operation, including communication between the speech module and task-planning system.

5.3 Planning Algorithm

[3 Weeks] Research on techniques and architectures

- A review of different approaches and existing planning algorithms will be conducted to identify suitable architectures.

[1 Weeks] Select best implementation method

- Candidate techniques will be compared, and the optimal method will be chosen for development.

[5 Weeks] Implementation

- The selected approach will be implemented with an emphasis on ensuring symbolic reasoning and adaptability to real-world tasks.

[5 Weeks] Integration

- The algorithm will be integrated into the system to handle planning tasks, translating perception (image/speech) into robotic execution commands.

5.4 Interface with Robot Arm

[5 Weeks] Develop library for interfacing with UR Arm

- A dedicated library will be created to enable seamless communication with the UR robotic arm.

[10 Weeks] Simulation

- Virtual simulations will be performed to verify system stability and safety before real-world trials.

[5 Weeks] Integrate other hardware with UR Arm

- Additional supporting hardware will be tested and interfaced to ensure full robotic tool-handling capabilities.

[2^{nd} semester] Actual implementation

- The final robotic interface will be tested and deployed on the physical UR Arm, completing the integration.

6 Theory and Technical Backup

6.1 Voice Control Systems (VCS)

Voice control systems have significantly transformed human-computer interaction by enabling users to interact with interfaces using voice commands and natural language. These systems leverage automatic speech recognition (ASR) to convert voice recordings into text, and advancements in deep learning have greatly enhanced their capabilities, alongside natural language processing. VCS offers numerous benefits, including saving users time and resources by providing quick services through an efficient interface without requiring extensive additional devices. They foster natural voice communication between users and devices, enhancing user comfort and creating a more intuitive interface. The adoption rate of voice-activated digital assistants has increased substantially, leading to a wide array of Internet of Things (IoT) devices incorporating voice-activated user interfaces for tasks ranging from creating grocery lists to controlling smart homes. Beyond smart gadgets and IoT, VCS technology is also applied in automotive systems to prevent accidents by allowing drivers to focus on their surroundings. VCS significantly improves the quality of life for various groups, including elderly people, children under four, and individuals with physical disabilities. They can provide complete control over home or office appliances, facilitate traffic regulation, and enhance healthcare systems [13].

6.2 Multimodal Automatic Speech Recognition

Multimodal Automatic Speech Recognition (ASR) enhances speech transcription by integrating visual context alongside audio signals, which is crucial for tasks such as embodied agents and human-robot interaction. By leveraging both modalities, multimodal ASR improves robustness and accuracy over traditional unimodal systems, achieving lower Word Error Rate (WER) and higher Recovery Rate (RR), especially under noisy or masked audio conditions. It is particularly effective at recovering visually salient words, such as object names, thereby increasing task completion rates in instruction-following scenarios [14], [18].

Typical architectures combine a speech encoder, such as wav2vec 2.0, with a visual encoder, often based on CLIP-ViT, feeding both into a Transformer-based decoder that attends jointly to audio and visual features. Unlike uni-

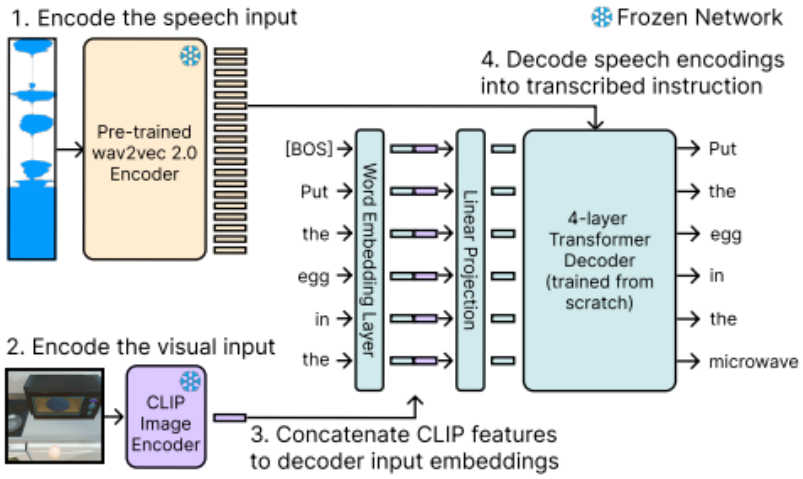

Figure 9: Multimodal ASR model architecture. (from [14])

modal ASR, which relies only on speech input, multimodal ASR grounds transcription in environmental context, reducing ambiguity and improving generalization to unseen speakers and settings [14], [18].

Applications include benchmarks like ALFRED, where agents must interpret spoken instructions to navigate and manipulate objects, as well as real-world human-robot communication. Compared with unimodal baselines, multimodal ASR achieves superior masked word recovery (up to 30% more) and mitigates performance degradation in noisy conditions [14], [18]. Challenges remain regarding generalization to real-world environments, as many studies rely on synthetic data and simplified noise models.

6.3 Neuro-Symbolic Artificial Intelligence (NSAI)

Neuro-Symbolic Artificial Intelligence (NSAI) is an emerging paradigm that combines the strengths of neural networks with symbolic reasoning. Neural networks excel in pattern recognition and data-driven learning but often lack explainability and explicit reasoning capabilities, operating as "black boxes". In contrast, symbolic AI is inherently interpretable and excels in logical reasoning but struggles with perception from raw data and requires extensive manual knowledge engineering. NSAI aims to bridge these gaps by integrating the robust statistical learning of neural networks with the structured knowledge and logic of symbolic AI, enabling systems to reason, make de-

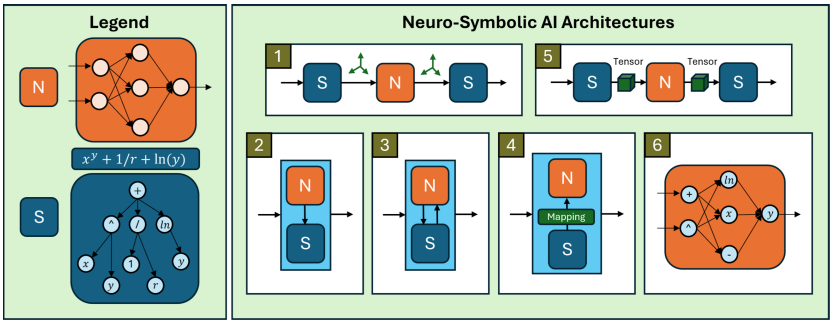

Figure 10: Neuro-Symbolic AI types (from [8])

cisions, and generalize knowledge more effectively from large datasets. This hybrid approach enhances interpretability, robustness, and trustworthiness, while also facilitating learning from less data [8].
NSAI systems can be categorized into various architectures based on the connection between their neural and symbolic components, as shown in Figure 10. These include:

- **Symbolic Wrapper (Type 1)**: Symbolic systems guide both input and output, leveraging neural networks internally for data-driven learning (e.g., DeepProbLog, Neuro-Symbolic Concept Learner (NSCL), hybrid neuro-symbolic robotics).

- **Symbolic [Neuro] (Type 2)**: Symbolic systems dominate, with neural modules used internally for perception and pattern recognition (e.g., in autonomous communication or scene interpretation).

- **Bidirectional Interaction (Type 3)**: Neural and symbolic components tightly interact, with neural models adjusting outputs based on symbolic constraints and symbolic modules evolving reasoning based on neural feedback (e.g., AlphaGo, AlphaZero, in robotics).

- **Sequential Connection (Type 4)**: Neural networks connect sequentially to symbolic systems via an explicit mapping layer, converting continuous neural outputs into structured symbolic knowledge (e.g., Neural Symbolic Machines (NSM)).

- **Embedded Symbolic Information (Type 5)**: Symbolic information is directly embedded into tensor representations, enhancing neural

network reasoning in a differentiable manner (e.g., Logic Tensor Networks (LTNs)).

- **Deeply Integrated (Type 6)**: Symbolic structures are directly processed by neural networks, blending symbolic reasoning and neural computation seamlessly (e.g., Neural Theorem Provers (NTP), Graph Neural Networks (GNNs)).

6.4 Large Language Model (LLM)

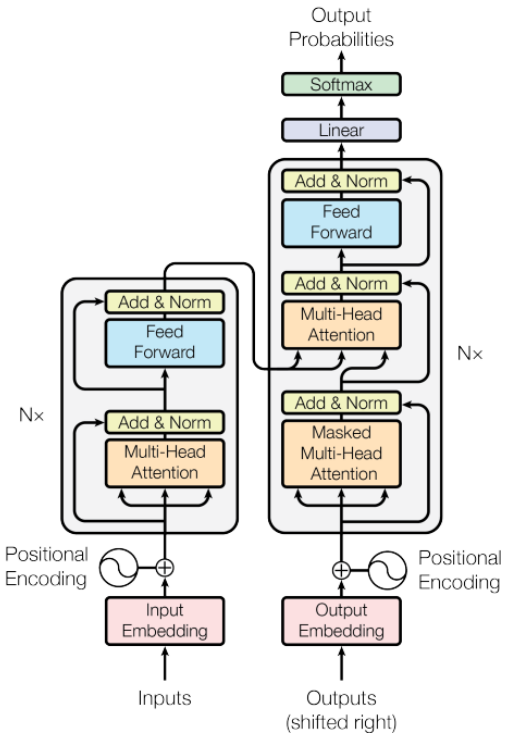

Figure 11: The Transformer-model architecture (from [26])

Large Language Models (LLMs), such as BERT and GPT-5, are pretrained neural networks that serve as powerful foundation models for language understanding and generation. They build on the Transformer architecture, shown in Figure 11, a sequence model that relies entirely on attention mechanisms and avoids recurrence or convolutions. The Transformer employs

an encoder-decoder structure with stacked layers of multi-head self-attention and feed-forward networks, enabling the model to relate different positions within a sequence and capture diverse representation subspaces. Positional encodings are added to input embeddings to provide information about token order [26]. LLMs represent words as vectors and acquire robust language and world knowledge through large-scale self-supervised learning, such as predicting masked words or the next word in a sequence. These models excel at tasks including machine translation, question answering, text classification, and text generation, leveraging vast amounts of text data to learn meanings and factual knowledge. Despite their capabilities, LLMs have limitations in deep understanding, reasoning, and completeness of knowledge, and raise societal concerns regarding bias and centralized model control [27].

The reason why it is preferred over traditional Natural Language Processing (NLP) models is due to the impressive ability to grasp broad contexts with little to no training. Mostly outperforms various Natural Language Processing (NLP) techniques without having to fine-tune with execution cost. Among the algorithms capable of handling a range of Natural Language Processing tasks, Large Language Model (LLM) has not only frequently appeared in numerous past studies but has also showcased immense potential for generating plans, transforming an initial world state into a desired state [4].

6.5 Hierarchical Reasoning Model (HRM)

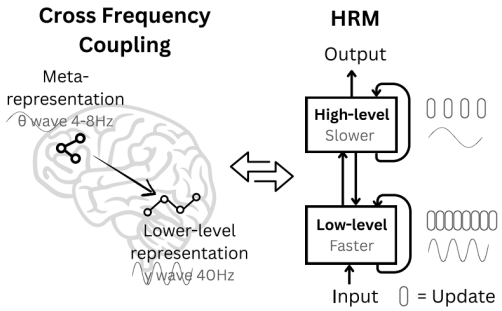

Figure 12: Hierarchical Reasoning Model Architecture Inspired by the Brain (from [28])

The Hierarchical Reasoning Model (HRM) is a brain-inspired neural ar-

chitecture designed to overcome the limitations of current Large Language Models (LLMs) in reasoning tasks. Whereas LLMs rely heavily on Chain-of-Thought (CoT) prompting, which breaks problems into text-based intermediate steps that are brittle, slow, and data-hungry, HRM performs reasoning entirely in its hidden state space. The model employs two recurrent modules operating at different timescales: a high-level module that conducts slow, abstract planning, and a low-level module that executes fast, detailed computations, as shown in Figure 12. This division enables HRM to achieve substantial computational depth in a single forward pass, avoiding the premature convergence and training instability typical of recurrent networks. To further improve efficiency, HRM introduces a one-step gradient approximation, bypassing the heavy memory requirements of Backpropagation Through Time (BPTT) while maintaining stable learning dynamics [28].

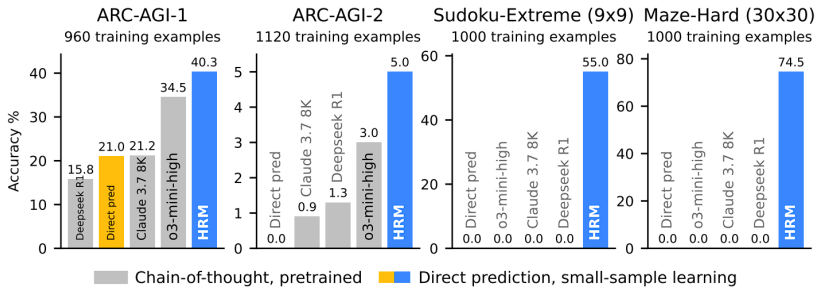

Figure 13: Hierarchical Reasoning Model Performances on Complex Benchmarks (from [28])

With only 27 million parameters and training on just 1,000 examples per task, HRM attains near-perfect performance on complex benchmarks such as Sudoku-Extreme and Maze-Hard, where state-of-the-art CoT-based LLMs fail entirely. It also surpasses much larger models on the Abstraction and Reasoning Corpus (ARC), a benchmark widely regarded as a measure of general reasoning ability, as shown in Figure 13. Notably, HRM achieves this without pre-training or access to CoT data, demonstrating strong sample efficiency and scalability. By integrating principles of hierarchical processing, temporal separation, and recurrent refinement—mirroring neural mechanisms in the brain—HRM represents a promising step toward universal computation and general-purpose reasoning systems that go beyond the shallow reasoning capacity of today's LLMs [28].

6.6 Vision Language Model (VLM)

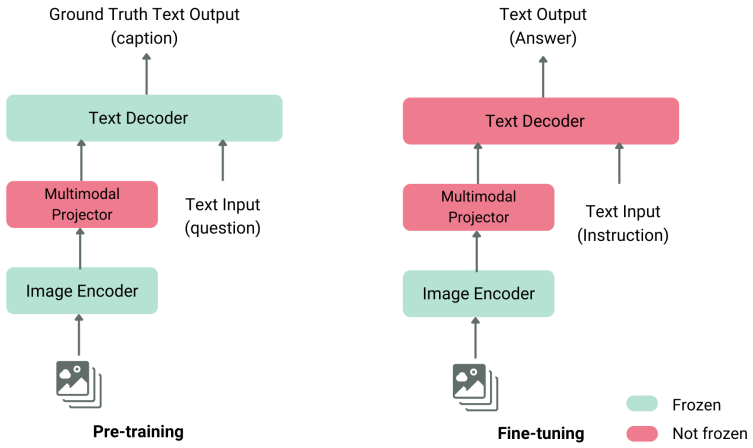

Figure 14: Structure of a Typical Vision Language Model (from [29])

Vision Language Models (VLMs) are multimodal generative AI models that combine image and text understanding to perform a wide range of tasks, including visual question answering, image captioning, document analysis, and instruction-based image recognition. Many VLMs can also reason about spatial relationships in images, producing outputs like bounding boxes or segmentation masks. Architecturally, they usually consist of an image encoder, a projection layer that aligns image and text embeddings, and a text decoder for generating output, as shown in Figure 14. Popular open-source VLMs include LLaVA, KOSMOS-2, Qwen-VL, CogVLM, and Fuyu-8B, each varying in model size, supported features such as chat or grounding, and image resolution [29].

VLMs are powerful in zero-shot generalization, allowing them to handle unseen images and prompts without additional training. Furthermore, fine-tuning allows adaptation to specialized applications, such as educational tools, interactive assistants, or automated document evaluation. Their limitations include susceptibility to hallucinations, biases from training data, and high computational requirements for end-to-end training or large models [29].

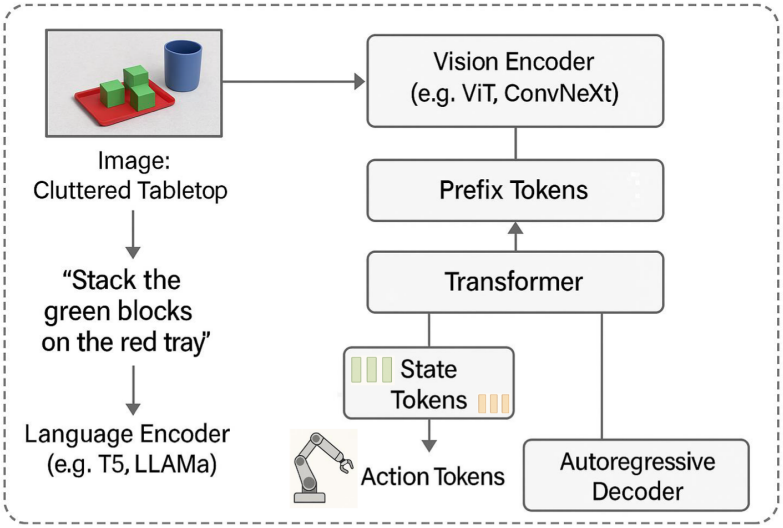

Figure 15: End-to-end Tokenization and Representation Process in VLA Models (from [30])

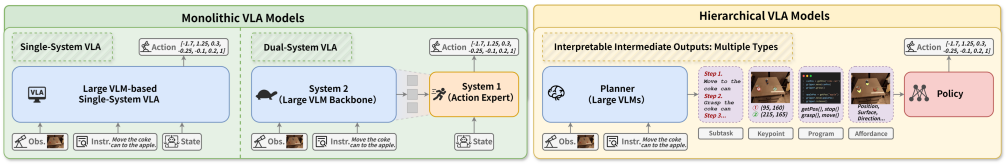

Figure 16: Comparison between Monolithic and Hierarchical models (from [31])

6.7 Vision-Language-Action (VLA) Models

Vision-Language-Action (VLA) models are multimodal AI systems that unify visual perception, natural language understanding, and embodied action into a single framework. Originally developed to overcome the fragmentation of traditional robotic pipelines, VLAs integrate pretrained Vision-Language Model (VLM) backbones with action policies, enabling robots to interpret high-level human instructions, reason about spatial relationships, and execute manipulation tasks in dynamic, real-world environments [30]. For example, a VLA can follow commands such as "place the red mug next to the laptop onto the top shelf" by grounding visual perception, spatial reasoning,

and motor control in a shared computational space. Architecturally, VLAs often rely on transformer-based models that fuse visual, linguistic, and state inputs into a common embedding [30], [31].

A core innovation of VLAs lies in treating robot control as a sequence generation problem. Continuous actions, such as joint angles or end-effector poses, are discretized into tokens, allowing the model to autoregressively predict action tokens in the same way language models generate text. Alongside these, prefix tokens encode context from images and text, while state tokens represent the robot's internal configuration, as shown in Figure 15. This unified tokenization framework enables reasoning and execution to occur in the same latent space, with a de-tokenizer translating action tokens back into executable motor commands [30], [31].

Two main architectural paradigms have emerged in robotic manipulation, as show in Figure 16. Monolithic models directly map multimodal inputs to low-level actions, sometimes combining a "slower" VLM for reasoning with a "faster" policy module for reactive control. In contrast, hierarchical models decouple high-level planning from low-level execution, with a planner producing intermediate representations (e.g., subtasks or keypoints) for downstream policies [31]. These designs balance efficiency, interpretability, and modularity, depending on task complexity.

VLAs represent a shift from handcrafted pipelines toward end-to-end, language-driven policy generation. They leverage the semantic grounding of VLM pretraining while extending it to action, offering broad potential in robotics, embodied assistants, and human-robot interaction. However, they face limitations such as high computational requirements, difficulty in precisely grounding abstract instructions, and restricted generalization beyond training distributions [30], [31].

6.8 Planning Domain Definition Language (PDDL)

Planning Domain Definition Language (PDDL) is a standardized language for modeling automated planning problems in artificial intelligence. It separates a domain (the rules of the world: types, predicates, and actions) from a problem (a specific initial state and goal). PDDL allows users to specify objects, predicates, actions, and goals in a structured way, enabling automated planners to generate sequences of actions that achieve desired outcomes. However, PDDL is deliberately neutral: it encodes the physics of a domain without embedding heuristics or solving strategies, meaning many

planners only support subsets of the language [32].

A classic illustration is the Briefcase World. Here, moving the briefcase also moves everything inside it, while objects can be put in or taken out. Below, the domain defines actions and the problem specifies the goal of moving a dictionary and briefcase to the office while leaving a paycheck at home:

```
(define (domain briefcase-world)
  (:requirements :strips :typing :conditional-effects)
  (:types location physob)
  (:constants B - physob)
  (:predicates (at ?x - physob ?l - location) (in ?x ?y - physob))
  (:action mov-b
    :parameters (?m ?l - location)
    :precondition (and (at B ?m) (not (= ?m ?l)))
    :effect (and (at B ?l) (not (at B ?m))
                 (forall (?z)
                   (when (and (in ?z) (not (= ?z B)))
                     (and (at ?z ?l) (not (at ?z ?m))))))))
  (:action put-in
    :parameters (?x - physob ?l - location)
    :precondition (not (= ?x B))
    :effect (when (and (at ?x ?l) (at B ?l))
                  (in ?x)))
  (:action take-out
    :parameters (?x - physob)
    :precondition (not (= ?x B))
    :effect (not (in ?x)))

(define (problem get-paid)
  (:domain briefcase-world)
  (:init (place home) (place office)
         (object p) (object d) (object b)
         (at B home) (at P home) (at D home) (in P))
  (:goal (and (at B office) (at D office) (at P home))))
```

PDDL has evolved from its early versions, like PDDL 1.2, which used predicate logic with true/false properties and actions, to more expressive forms. PDDL 2.1 introduced durative actions and numeric fluents for modeling time and resources, while PDDL 2.2 added derived predicates and timed

literals. PDDL 3.0 incorporated soft constraints and preferences with costs, and PDDL+ enabled modeling of processes and uncontrollable events. Specialized variants such as PPDDL, NDDL, and MADDL were later developed for domain-specific planning needs [33].

6.9 ROSPlan

ROSPlan is a framework that provides a structured collection of tools for integrating AI planning into robotic systems built on the Robot Operating System (ROS). Originally developed to bridge the gap between symbolic task planning and real-world robotic execution, ROSPlan encapsulates the entire deliberation cycle: knowledge representation, problem generation, planning, parsing, and plan execution. This modular design enables robots to autonomously reason about their environment, generate feasible plans, and reliably execute them in dynamic contexts.

At its core, ROSPlan is composed of multiple dedicated nodes, each responsible for a specific stage in the planning pipeline, as shown in Figure 17. The Knowledge Base maintains both the PDDL domain model and the current problem instance, storing the state as a symbolic PDDL representation that can be queried or updated through ROS messages. Building on this, the Problem Interface generates new PDDL problem instances by combining domain details with the current world state, publishing them as strings or writing them to file.

Once a problem instance is generated, the Planner Interface invokes an external AI planner. Acting as a wrapper, it supplies the domain and problem files, calls the planner via a configurable command line, and reports whether a valid plan was found. The resulting plan is then processed by the Parsing Interface, which translates the raw planner output into structured ROS messages that can be executed within the system.

The execution stage is managed by Plan Dispatch, which subscribes to plan topics and coordinates the enactment of individual actions. Each action in the parsed plan is linked to the relevant ROS processes responsible for carrying it out, enabling seamless integration between high-level symbolic plans and low-level robot control.

Together, these nodes form a closed deliberation loop that unifies symbolic AI planning with ROS. ROSPlan thereby enables robots to progress from goals such as “navigate to room B and deliver the package” to actual coordinated actions, while maintaining modularity, transparency, and

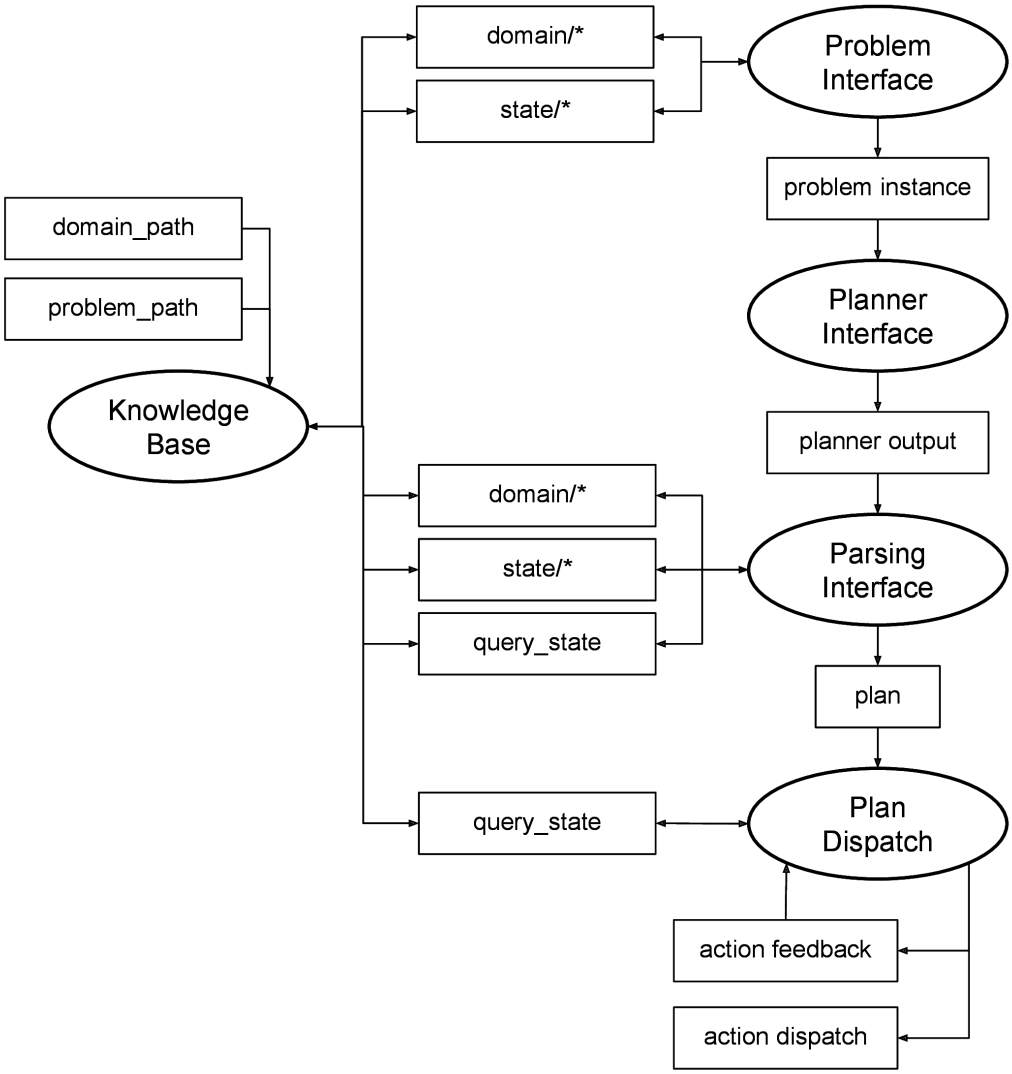

Figure 17: ROSPlan Architecture (from [34])

reusability across robotic domains. Its layered architecture provides a foundation for building robust, goal-driven autonomy in service robotics, multi-robot coordination, and other AI-enabled robotic applications [34].

6.10 MoveIt2

MoveIt2 is a motion planning framework for ROS2 that integrates kinematics, collision checking, planning, and execution into a unified pipeline. It enables robots to interpret high-level goals, such as "pick the object and place it in the bin", and compute collision-free, dynamically feasible trajectories to accomplish them.

At its core, MoveIt2 is structured around modular components, as shown in Figure 18. The Kinematics module translates between joint states and end-effector poses, while the Planning Scene Monitor maintains an up-to-date model of the environment. The Motion Planning system leverages planners such as OMPL or TrajOpt to generate valid paths, which are refined in the Trajectory Processing stage into smooth, executable trajectories.

The central move_group node provides a user-facing API for motion planning and execution, while advanced features like Hybrid Planning combine long-horizon strategies with local reactive control. The MoveIt Task Constructor further extends this capability by composing multi-step task sequences, making it suitable for applications such as bin picking and assembly.

MoveIt2 shifts motion planning from isolated libraries toward a flexible, extensible middleware. Its modular design promotes reusability and adaptability across robotic domains, though challenges remain in accurate modeling, computational cost, and system tuning [35], [36].

6.11 Artificial Intelligence in Image Processing

Artificial Intelligence (AI) has significantly transformed image processing, introducing cutting-edge methods and applications that streamline processes for enhanced speed and accuracy. Image processing itself involves understanding digital images as pixel-based visual information, along with various formats (e.g., JPEG, PNG), enhancement techniques (such as adjusting brightness or reducing noise), and filtering/restoration procedures. Within this framework, AI, especially through deep learning architectures like Convolutional Neural Networks (CNNs) and Generative Adversarial Networks

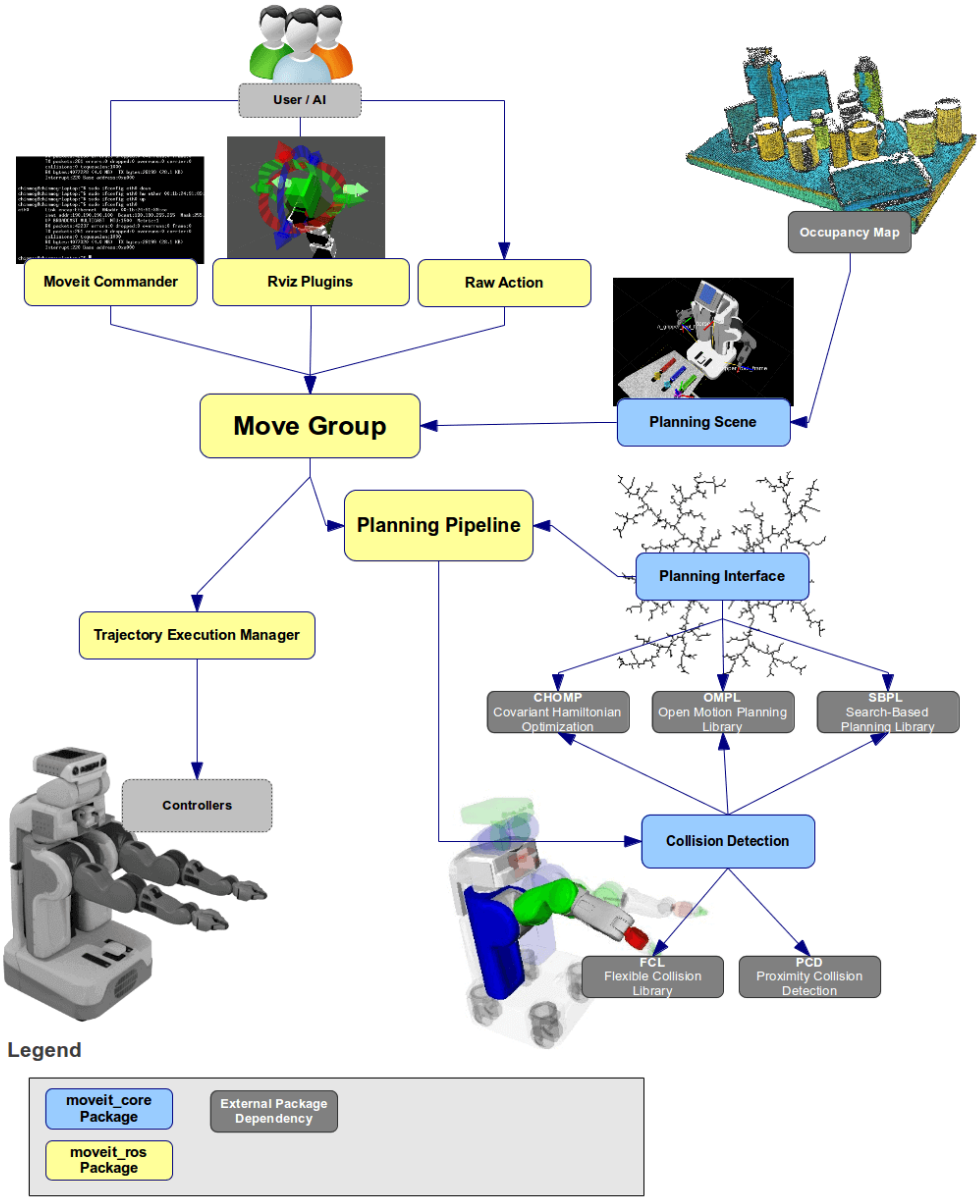

Figure 18: MoveIt2 motion planning pipeline.

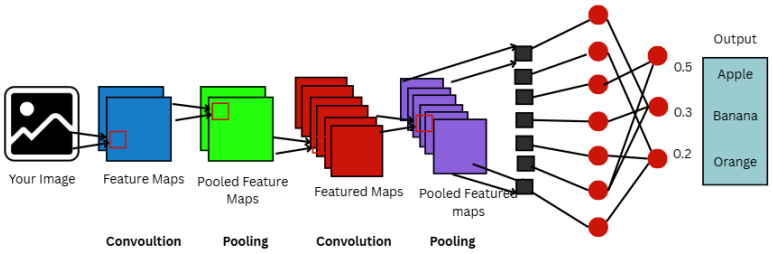

Figure 19: Convolutional Neural Network (CNN) Architecture for Image Classification

(GANs), enables systems to extract crucial features, recognize objects, segment images, and even generate realistic visual content. This capability allows robots to interpret visual information similarly to humans, finding practical uses in areas like autonomous vehicles for navigation and surveillance systems for security [37].

The capabilities of AI image processing are broad and impactful, including accuracy enhancements, efficiency improvements, and the capacity to manage large datasets. AI has demonstrated significant success in improving diagnostic accuracy and early disease detection in healthcare, optimizing industrial packing strategies for greater efficiency and reduced waste, and enabling real-time applications and deployment on devices with limited resources through crucial optimization techniques like model compression. Advanced functions such as object detection, image segmentation, and content-based image retrieval are also facilitated by these methods. Despite these advantages, the application of AI in image processing faces notable limitations and ethical challenges. These include critical concerns regarding data privacy, interpretability of AI decisions, and the potential for biases embedded in training data to produce prejudiced results. Therefore, ensuring responsible, transparent AI systems that respect cultural diversity and address potential impacts on employment are vital for the technology's continued development [37].

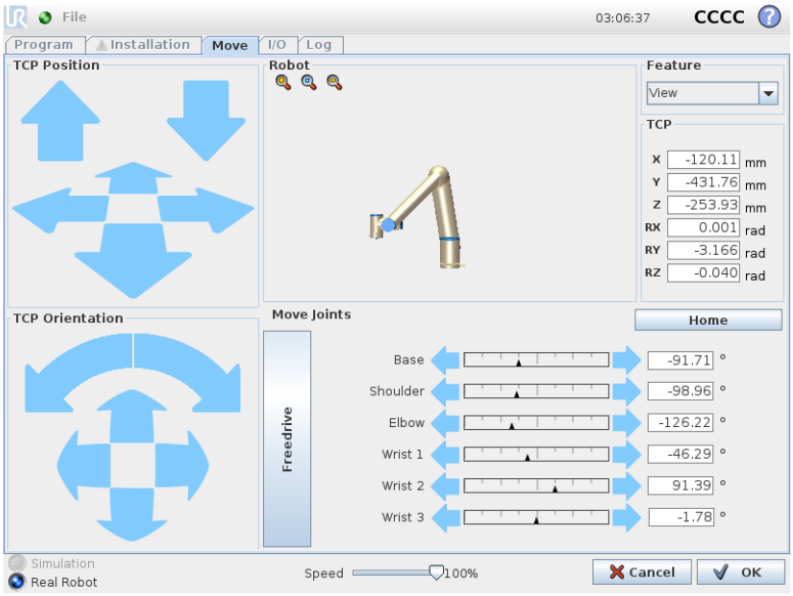

Figure 20: the Move Screen interface of a Universal Robots (UR) robotic arm's [38]

6.12 Universal Robots Arm (UR ARM)

The Universal Robots (UR) robotic arm is a widely adopted collaborative robotic manipulator (cobot) designed for flexible deployment in industrial, engineering, and research settings. Unlike traditional industrial robots, UR arms are specifically engineered to operate safely alongside humans without requiring extensive physical barriers, thanks to their integrated safety features, lightweight design, and force-limiting mechanisms. These properties make UR arms particularly suitable for human-robot collaboration in dynamic environments such as manufacturing, assembly, and tool-handling applications. From a technical perspective, UR arms employ high-precision servo motors and torque sensors at each joint, enabling six degrees of freedom for dexterous manipulation tasks. Their kinematic flexibility allows execution of both structured and unstructured manipulation tasks, such as grasping, tool usage, and assembly operations. Additionally, UR arms are equipped with programmable interfaces, including graphical user interfaces (GUI), teach pendants, and increasingly, APIs compatible with external AI systems [38]. This enables integration with higher-level control architectures,

such as neuro-symbolic AI (NSAI) or large language models (LLMs), for advanced planning, reasoning, and adaptive execution.

7 Project Outcome

By the deadline of this project , there will be a detailed report documenting how the proposed AI robot arm planning approach performs on the TASK and Motion Planning (TAMP) benchmarks, compared to other baseline solutions. This report will highlight the assistant's strength in combining voice-command understanding, image-processing, and robotic manipulation for engineering tool-handling task.

Accompanying the project is a robotic prototype built on a UR5e robotic arm equipped with a parallel gripper. The robot is connected to a workstation running for UR Arm Library, which serves as the central processing unit, integrating both perception and planning modules. Through this workstation, the robot interfaces with:

- **Camera**:A depth camera for real-time object recognition and workspace mapping.
- **Microphone**: A input with ASR(Automatic Speech Recognition) pipeline for natural voice-command input.
- **Robotics**: Motion controllers for safe execution of grasping and tool-passing tasks.

With these specifications, the prototype is expected to demonstrate the following core capabilities:

7.1 Autonomous Planning:

The robot should be able to decompose natural voice instructions (e.g., "pass me the wrench, then place the screwdriver on the table") into a symbolic task plan and execute it.

7.1.1 Path and Motion Safety

The robot should plan trajectories that are collision-free, navigating cluttered tool-workspaces without endangering users or damaging objects.

7.1.2 Generalization:

The assistant should adapt to unseen scenarios, such as handling tools of unfamiliar shapes or working in rearranged environments, while maintaining performance targets of more than 70 % sequence correctness and 85 % collision-free trajectories in simulation when executing TAMP tasks [1]

7.1.3 Human Collaboration:

The robot must safely pass tools to human collaborators by respecting dynamic human safety zones, pausing or re-planning if a human unexpectedly enters its workspace.

Overall, at the surface level, the outcome of this project will be a robot arm assistant capable of autonomously planning and executing tool-handling tasks in engineering workshop scenarios. On a deeper level, this project demonstrates how combining neural and symbolic AI can produce robots that reason about complex tasks, adapt to novel environments, and safely interact with humans. In the long run, this prototype serves as a foundation for intelligent robotic assistants in manufacturing, laboratories, and educational workshops, showing how autonomous planning can be extended from research into real-world engineering applications.

8 Summary and Benefit to Real Industry

This project aims to build a prototype of a robotic planning assistant that integrates Large Language Models (LLMs) with symbolic reasoning for tool-handling in engineering environments. The LLM is responsible for translating natural language instructions into multi-step task plans, while Symbolic AI serves as a verification layer, checking for correctness and preventing unsafe or hallucinated actions. The prototype will serve as a proof of concept for how hybrid planning architectures can enable safe, interpretable, and adaptable robotic assistants in industrial settings.

As noted in recent studies [7], [9], [12], [15], LLMs have demonstrated strong capabilities in task decomposition, code generation, and multimodal integration, while symbolic systems contribute precision, interpretability, and logical consistency [12]. Combining these approaches enhances reliability and operator trust, allowing robots to adapt to unseen scenarios while remaining verifiable.

The expected outcome is a system that reduces downtime, improves safety, and supports human-robot collaboration by enabling robots to be re-tasked directly from natural language instructions without extensive re-training. If successful, this approach can extend beyond tool-handling to applications in **manufacturing, construction, healthcare, and hazardous environments**, offering a scalable framework for safe and efficient robotic deployment across industries.

9 Roles and Responsibilities

Project Task	**Task Description**	**Assigned to**
Planning Algorithm	Research architectures, select method, implement, and integrate neuro-symbolic planning algorithm	Kanisorn Sangchai
Image Processing	Research techniques, select best method, implement, and integrate image processing for robot system	Methasit Boonpun
Speech Processing	Research methods, choose best approach, implement, and integrate speech-to-command module	Withawin Kraipetchara
Interface with Robot Arm	Develop UR Arm library, integrate hardware, run simulation, and perform real implementation	Krittin Kitjaruwannakul
Project Paperwork	Prepare proposal, progress reports, draft final report, and final presentation	All Members

Table 1: An outline of major responsibilities assigned to each member within the project

The roles and duties of this project revolve around the four main features. Each feature is allocated to a team member as outlined in Table 1

References

[1] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, p. 265–293, May 2021. [Online]. Available: http://dx.doi.org/10.1146/annurev-control-091420-084139

[2] Z. Zhao, S. Cheng, Y. Ding, Z. Zhou, S. Zhang, D. Xu, and Y. Zhao, "A survey of optimization-based task and motion planning: From classical to learning approaches," *IEEE/ASME Transactions on Mechatronics*, vol. 30, no. 4, p. 2799–2825, Aug. 2025. [Online]. Available: http://dx.doi.org/10.1109/TMECH.2024.3452509

[3] H. Guo, F. Wu, Y. Qin, R. Li, K. Li, and K. Li, "Recent trends in task and motion planning for robotics: A survey," *ACM Computing Surveys*, vol. 55, no. 13s, p. 1–36, Jul. 2023. [Online]. Available: http://dx.doi.org/10.1145/3583136

[4] H. Wei, Z. Zhang, S. He, T. Xia, S. Pan, and F. Liu, "Plangenllms: A modern survey of llm planning capabilities," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2025, p. 19497–19521. [Online]. Available: http://dx.doi.org/10.18653/v1/2025.acl-long.958

[5] Z. Li, X. Wu, H. Du, H. Nghiem, and G. Shi, "Benchmark evaluations, applications, and challenges of large vision language models: A survey," Jan. 2025. [Online]. Available: http://dx.doi.org/10.32388/GXR68Q

[6] M. Tantakoun, C. Muise, and X. Zhu, "Llms as planning formalizers: A survey for leveraging large language models to construct automated planning models," in *Findings of the Association for Computational Linguistics: ACL 2025*. Association for Computational Linguistics, 2025, p. 25167–25188. [Online]. Available: http://dx.doi.org/10.18653/v1/2025.findings-acl.1291

[7] G. Tziafas and H. Kasaei, "Enhancing interpretability and interactivity in robot manipulation: A neurosymbolic approach," 2023. [Online]. Available: https://arxiv.org/abs/2210.00858

[8] R. Fitas, "Neuro-symbolic ai for advanced signal and image processing: A review of recent trends and future directions," *TechRxiv Preprint*, Apr. 2025. [Online]. Available: http://dx.doi.org/10.36227/techrxiv.174353019.93925639/v1

[9] N. K, H. Singh, V. Bindal, A. Tuli, V. Agrawal, R. Jain, P. Singla, and R. Paul, "Learning neuro-symbolic programs for language guided robot manipulation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2023, p. 7973–7980. [Online]. Available: http://dx.doi.org/10.1109/icra48891.2023.10160545

[10] K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. B. Tenenbaum, "Neural-symbolic vqa: Disentangling reasoning from vision and language understanding," in *Neural Information Processing Systems*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:52919654

[11] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," 2021. [Online]. Available: https://arxiv.org/abs/2109.12098

[12] Y. Chen, Y. Hao, Y. Zhang, and C. Fan, "Code-as-symbolic-planner: Foundation model-based robot planning via symbolic code generation," 2025. [Online]. Available: https://arxiv.org/abs/2503.01700

[13] A. A. M. E.-H. El-Azazy, R. A.-e. El-kammar, A. M. Fawzy, and H. M. Abd Elkader, "A survey on advancements in voice control systems enhancing human-computer interaction through speech recognition and ai," *Engineering Research Journal (Shoubra)*, vol. 0, no. 0, p. 0–0, Jan. 2025. [Online]. Available: http://dx.doi.org/10.21608/erjsh.2025.334371.1373

[14] A. Chang, X. Zhu, A. Monga, S. Ahn, T. Srinivasan, and J. Thomason, "Multimodal speech recognition for language-guided embodied agents," in *Interspeech 2023*, 2023, pp. 1608–1612.

[15] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics*

and Automation (ICRA). IEEE, May 2023. [Online]. Available: http://dx.doi.org/10.1109/icra48891.2023.10160591

[16] Y. Liu, Z. Liang, Z. Chen, T. Chen, M. Hu, W. Dong, C. Xu, Z. Han, Y. Qin, and Y. Mu, "Hycodepolicy: Hybrid language controllers for multimodal monitoring and decision in embodied agents," 2025. [Online]. Available: https://arxiv.org/abs/2508.02629

[17] Y. Meng, F. Chen, Y. Chen, and C. Fan, "Audere: Automated strategy decision and realization in robot planning and control via llms," 2025. [Online]. Available: https://arxiv.org/abs/2504.03015

[18] J. Hu, Z. Li, P. Wang, H. Ai, L. Zhang, and H. Zhao, "Vhasr: A multimodal speech recognition system with vision hotwords," 2024. [Online]. Available: https://arxiv.org/abs/2410.00822

[19] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, A. Wahid, V. Sindhwani, and J. Lee, "Transporter networks: Rearranging the visual world for robotic manipulation," 2022. [Online]. Available: https://arxiv.org/abs/2010.14406

[20] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: from natural language instructions to feasible plans," *Autonomous Robots*, vol. 47, no. 8, p. 1345–1365, Nov. 2023. [Online]. Available: http://dx.doi.org/10.1007/s10514-023-10131-7

[21] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," 2023. [Online]. Available: https://arxiv.org/abs/2304.02643

[22] A. Munir, F. Z. Qureshi, M. H. Khan, and M. Ali, "Tlac: Two-stage lmm augmented clip for zero-shot classification," 2025. [Online]. Available: https://arxiv.org/abs/2503.12206

[23] H. Wang, Y. Zhang, Y. Wang, and J. Li, "6-dof grasp detection in clutter with enhanced receptive field and graspable balance sampling," 2024. [Online]. Available: https://arxiv.org/abs/2407.01209

[24] J. V. Hurtado and A. Valada, "Semantic scene segmentation for robotics," *arXiv preprint*, 2024. [Online]. Available: https://arxiv.org/abs/2401.07589

[25] U. Robots, "Universal robots ros2 driver," https://github.com/UniversalRobots/Universal_Robots_ROS2_Driver, 2023, accessed: 2025-09-12.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[27] C. D. Manning, "Human language understanding & reasoning," *Daedalus*, vol. 151, no. 2, pp. 127–138, 05 2022. [Online]. Available: https://doi.org/10.1162/daed_a_01905

[28] G. Wang, J. Li, Y. Sun, X. Chen, C. Liu, Y. Wu, M. Lu, S. Song, and Y. A. Yadkori, "Hierarchical reasoning model," 2025. [Online]. Available: https://arxiv.org/abs/2506.21734

[29] Merve and E. Beeching, "Vision language models explained," 2024, accessed: 2025-08-23. [Online]. Available: https://huggingface.co/blog/vlms

[30] R. Sapkota, Y. Cao, K. I. Roumeliotis, and M. Karkee, "Vision-language-action models: Concepts, progress, applications and challenges," 2025. [Online]. Available: https://arxiv.org/abs/2505.04769

[31] R. Shao, W. Li, L. Zhang, R. Zhang, Z. Liu, R. Chen, and L. Nie, "Large vlm-based vision-language-action models for robotic manipulation: A survey," 2025. [Online]. Available: https://arxiv.org/abs/2508.13073

[32] D. McDermott, M. Ghallab, A. E. Howe, C. A. Knoblock, A. Ram, M. M. Veloso, D. S. Weld, and D. E. Wilkins, "Pddl-the planning domain definition language," 1998. [Online]. Available: https://api.semanticscholar.org/CorpusID:59656859

[33] A. Green, "What is planning domain definition language (pddl)?" accessed: 2025-08-23. [Online]. Available: https://planning.wiki/guide/whatis/pddl

[34] K. P. Group, "Rosplan documentation," 2025, accessed: 2025-09-15. [Online]. Available: https://kcl-planning.github.io/ROSPlan/documentation/

[35] P. Robotics, "Moveit2 concepts documentation," 2025, accessed: 2025-09-15. [Online]. Available: https://moveit.picknik.ai/main/doc/concepts/concepts.html

[36] E. Robotics, "Moveit2 development guide (ros2)," 2025, accessed: 2025-09-15. [Online]. Available: https://docs.elephantrobotics.com/docs/mycobot_280_ar_en/3-FunctionsAndApplications/6.developmentGuide/ROS/12.2-ROS2/12.2.5-Moveit2/#introduction-to-moveit2

[37] S. Boopathi, B. K. Pandey, and D. Pandey, *Advances in Artificial Intelligence for Image Processing: Techniques, Applications, and Optimization*. IGI Global, Jun. 2023, p. 73–95. [Online]. Available: http://dx.doi.org/10.4018/978-1-6684-8618-4.ch006

[38] Universal Robots A/S, *User manual – UR10e e-Series – SW 5.13*, Universal Robots, 2025, available at: https://www.universal-robots.com/download/.