Linköping University | Department of Management and Engineering
Master's Thesis, 30 credits | Mechanical Engineering
Spring 2025

Integrating Vision, Language Models, & Robotic Control for Personalized Task Execution in Virtual Environments

Author:
Oscar Chigozie Ikechukwu

Supervisor: Mehdi Tarkian
Examiner: Johan Persson

Full-time Report

13 May 2025

Linköping University
SE-581 83 Linköping, Sweden
+46 13 28 10 00, www.liu.se

Master's thesis, 30 credits
Mechanical Engineering
Spring semester, 2025

Author:
Oscar Chigozie Ikechukwu
M.Sc. Student, Mechanical Engineering
Engineering Design and Product Development
Linköping University, Sweden

Supervisor:
Mehdi Tarkian
Senior Associate Professor
Dept. of Management and Engineering (IEI)
Division of Product Realisation (PROD)
Linköping University, Sweden

Examiner:
Johan Persson
Associate Professor, Head of Unit
Dept. of Management and Engineering (IEI)
Division of Product Realisation (PROD)
Linköping University, Sweden

Copyright Notice

© 2025 Oscar Chigozie Ikechukwu. All rights reserved.
No part of this thesis may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), without prior written permission from the author or Linköping University.

Abstract

{Place Holder}

Keywords: Omniverse Isaac Sim, Large Language Models, Computer Vision, Human Robot Collaboration

Declaration

I, Oscar Chigozie Ikechukwu, hereby declare that this thesis titled "Integrating Vision, Language Models, and Robotic Control for Personalized Task Execution in Virtual Environments", submitted to Linköping University in partial fulfilment of the requirements for the master's degree in mechanical engineering, is my original work.

In preparing this thesis, I utilized AI-assisted editing tools, specifically ChatGPT and Grammarly, to refine technical documentation, enhance structural coherence, and ensure linguistic accuracy. However, all core ideas, analyses, and conclusions presented herein are my own and reflect the original research conducted during this thesis project.

This research has not been submitted for any other degree or examination at any other institution. Any material derived from other sources has been appropriately acknowledged and cited.

Oscar Chigozie Ikechukwu
13 May 2025

Acknowledgements

I wish to express my deepest gratitude to my supervisor, *Mehdi Tarkian*, for his invaluable guidance, insightful feedback, encouragement and support throughout this research journey. His expertise in Design Automation have been instrumental in the successful completion of this thesis.

I am also profoundly grateful to my examiner, *Johan Persson*, for his critical evaluations and constructive suggestions, which have significantly enhanced the quality of this work.

Special thanks to my colleagues and friends at Linköping University, particularly *Sanjay Nambiar* and *Rahul Chiramel*, for their camaraderie, stimulating discussions, and continuous support. Your contributions have enriched my academic experience.

I also extend my heartfelt appreciation to my family for their steadfast support, patience, and constant motivation. Your belief in me has been my greatest source of strength throughout this endeavour.

Finally, I acknowledge the contributions of the research community, whose pioneering work has laid the foundation for this study.

Table of Contents

1. Introduction 1
  - 1.1. Problem Context 1
  - 1.2. Motivation 2
  - 1.3. Background 4
    - 1.3.1. Evolution of Human-Robot Interaction 4
    - 1.3.2. Vision-Based Perception: 5
    - 1.3.3. Language Understanding with LLMs 5
    - 1.3.4. Robotic Control and Simulation 6
    - 1.3.5. Personalization and Ethical Implications 6
  - 1.4. Aims & Goals 7
  - 1.5. Research Questions 7
  - 1.6. Limitations & Delimitations 8
    - 1.6.1. Limitations (external constraints) 8
    - 1.6.2. Delimitations (strategic choices) 8
  - 1.7. Expected Outcomes 9
    - 1.7.1. Planned deliverables 9
  - 1.8. Key Topics 10
  - 1.9. Thesis Outline 10

2. Literature Review 12
  - 2.1. Theoretical Framework 12
  - 2.2. Evolution of Cobots: from Scripted to Adaptive 12
  - 2.3. Vision Systems: Object Detection to User-Centric Context 13
    - 2.3.1. Seeing the world through cameras 14
    - 2.3.2. Biometrics as the first step to personalization 14
  - 2.4. Language Models: Translating Intent into Actionable Code 15
    - 2.4.1. NLP in robotics: “syntax to semantics” 15
    - 2.4.2. Limits of LLMs in embodied robotics 15
    - 2.4.3. Conversational memory: “chatbots to cobots” 17
  - 2.5. Robot Task Planning & Execution in Virtual Environments 17
    - 2.5.1. Robot control in virtual environments 17
    - 2.5.2. Modular architectures for control 17
    - 2.5.3. Task planning & execution frameworks 18
    - 2.5.4. Database-driven personalization in HRI 18
  - 2.6. Integration & challenges 18

3. Methodology 19
  - 3.1. Techniques & Tools 19

3.1.1. The Method; RAD using LLMs 19
3.1.2. Vision tools: 19
3.1.3. LLM tools: 19
3.1.4. Simulation tools: 19
3.1.5. Database tools: 19
3.2. System Architecture & Framework 19
3.2.1. High-level overview 19
3.2.2. Authentication module 19
3.2.3. Vision & perception module 19
3.2.4. Input processing module 19
3.2.5. Language understanding module 19
3.2.6. Task planning & Robot control module 19
3.2.7. Simulation module: 19
3.3. Data Collection, Pre-processing & Flow 19
3.4. System Integration & Implementation Steps 19
3.4.1. Central Task Manager / GUI 19
3.4.2. Implementation Steps 21
3.4.3. Notable Implementation Features 28
4. Results & Findings 29
4.1. Results 29
4.2. Findings 32
5. Discussion & Conclusion 33
5.1. Discussion 33
5.2. Conclusion & Future work 33

List of Figures & Tables

Table of Figures

Figure 1-1. A robot misinterpreting the command to "tidy up" by stuffing cables into a microwave. (AI generated image) 1

Figure 1-2. Proposed System Architecture - A unified framework integrating vision, LLMs, & robot control 2

Figure 1-3. Mirokai robot, CES 2025: A hospital-assistive robot leveraging vision & language understanding 3

Figure 2-1. Robot capabilities enabled by AI. 13

Figure 3-1. RAD Process Phases 19

Figure 3-3. High-level diagram of the system architecture. 19

Figure 3-4. Object detection data, showing object properties from the scene .19

Figure 3-5. Database tables showing integrated dataflow 19

Figure 3-6. Code snippet - showing the TaskManagerGUI as an entry point. 20

Figure 3-7: Operational workflow – Approach I 22

Figure 3-8. Excerpt from face_auth.py showing facial recognition approach 23

Figure 3-9. Excerpt showing how the user voice is captured for registration 24

Figure 3-10. Camera vision script updates the properties of detected objects in the database. 24

Figure 3-11. Threads concurrently run for command capture from the orchestrator script 25

Figure 3-12. Code snippet showing voice command capture 25

Figure 3-13. Code snippet showing gesture cue capture 25

Figure 3-14. Excerpt from synchronizer.py combining concurrent voice and gesture inputs 26

Figure 3-15. Code snippet showing LLM prompt, utilizing objects and sequences from database 26

Figure 3-16. Instruction_operation_sequence insertion logic 27

Figure 3-17. Illustrates final logging of success/failure or time metrics in simulation_results. 27

Table of Tables

Table 3-1. Inbound and outbound data flow 19

Table 3-2. Database schema 19

Abbreviations

AI:	Artificial Intelligence
BERT:	Bidirectional Encoder Representations from Transformers
Cobot:	Collaborative robot
CLIP:	Contrastive Language–Image Pretraining
GDPR:	General Data Protection Regulation
GPT:	Generative Pre-trained Transformer
HRI:	Human-Robot Interaction
LLM:	Large Language Model
NLP:	Natural Language Processing
RL:	Reinforcement Learning
ROS:	Robot Operating System
Sim2Real:	Simulation-to-Reality
SQLite:	Structured Query Language Lite.
UR:	Universal Robots (collaborative robot brand)
URDF:	Unified Robot Description Format
USD:	Universal Scene Description (Omniverse file format)
VLM:	Vision-Language Model
YOLO:	You Only Look Once (object detection framework)

Nomenclature

Affordance Recognition: Identifying how objects can be manipulated based on visual cues (e.g., a "handle" implies pulling).

Control Script: Code translating high-level task plans into low-level robot movements (e.g., joint trajectories).

Contrastive Language-Image Pretraining (CLIP): a vision-language model that aligns visual and textual representations, enabling robots to interpret commands like ‘pick up the red cup.’

Conversational Memory: An LLM’s ability to retain context from prior user interactions for adaptive dialogue.

Modular Architecture: A system design where components (vision, language, control) operate independently but integrate seamlessly.

Omniverse Simulation: A virtual environment (NVIDIA) for testing robotic systems with physics-based accuracy.

Personalization Layer: A database-driven component that tailors interactions using user-specific data (e.g., preferences, history).

Reality Gap: Discrepancies between idealized simulation and real-world robotic performance conditions.

Task Graph: A structured representation of tasks/subtasks generated from natural language commands.

Vision-Language Integration: Combining computer vision (object detection) and language models (LLMs) to interpret and execute tasks.

1. Introduction

This chapter introduces the motivation and problem context for the thesis. It highlights the background; key limitations in current robotic systems and presents a proposed framework that integrates vision, language, and control in simulation. The chapter further outlines the research aims, key questions, expected outcomes, and discusses the study's limitations and scope.

1.1. Problem Context

Imagine instructing a robot, *"Tidy the lab while I finish my coffee,"* only to find it attempting to clean by stuffing cables into a microwave. While humorous, this illustrates a serious limitation: most robotic systems excel in structured, predictable environments but struggle in dynamic, human-centred spaces.

Figure 1-1. A robot misinterpreting the command to "tidy up" by stuffing cables into a microwave. (AI generated image)

Collaborative robots are increasingly expected to operate in such unstructured settings, where humans communicate using casual, nuanced, and often ambiguous language. Yet, current systems rely on rigid commands, possess little to no contextual awareness, and often cannot recall prior interactions. This disconnect between natural human communication and robotic execution remains a major barrier to real-world adoption.

To bridge this gap, the seamless integration of computer vision, natural language processing, and advanced robotic control is essential. This thesis introduces a unified framework (Figure 1-2) that combines vision-based perception, large language models (LLMs), and adaptive robotic control to enable collaborative robots to interpret user commands, infer contextual intent, and simulate task execution within a virtual environment. The goal is to develop an AI-driven pipeline in which a cobot—such as ABB's YuMi or a Universal Robot arm—can process high level natural language commands, perform task decomposition and planning via LLM-based reasoning, and generate virtual demonstrations of task sequences prior to physical execution.

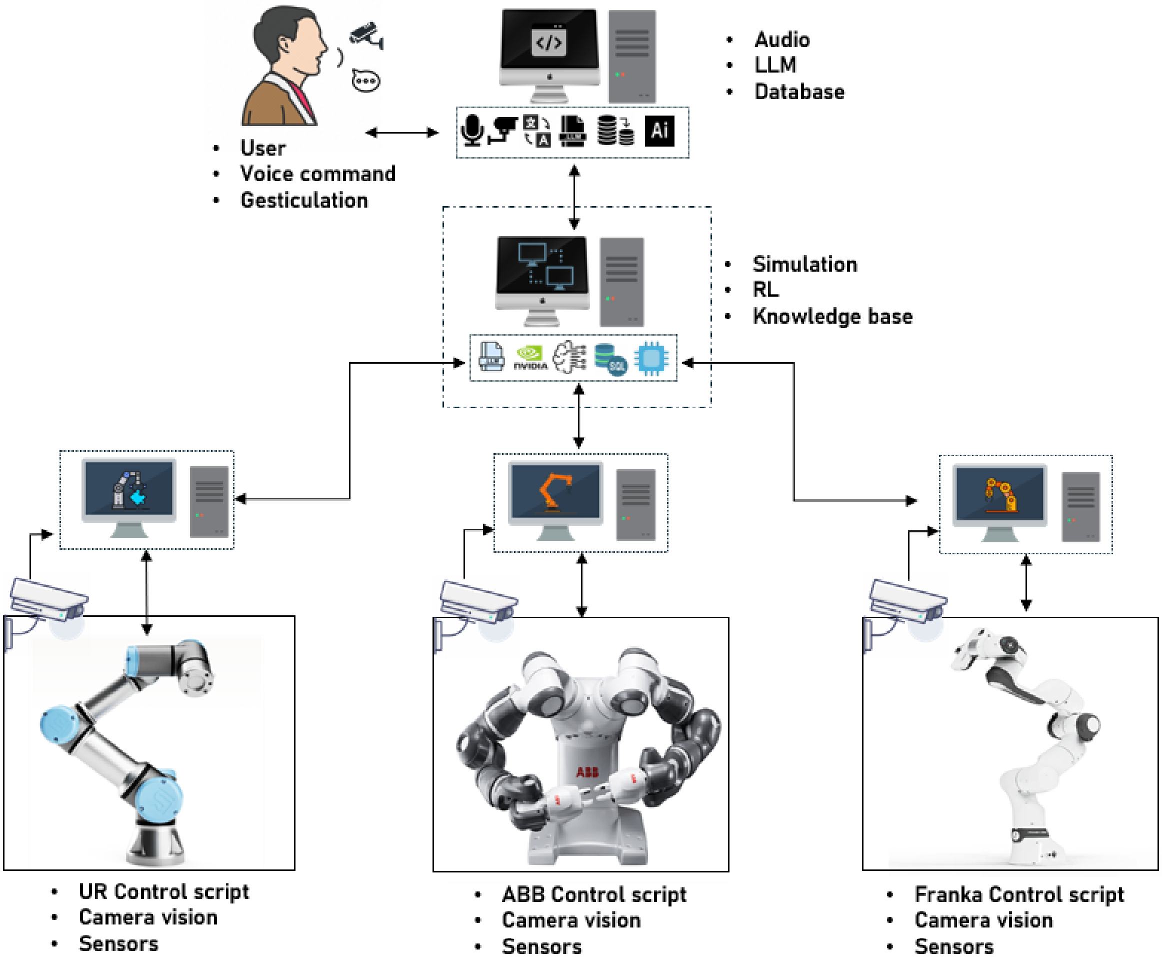


Figure 1-2. Proposed System Architecture - A unified framework integrating vision, LLMs, & robot control

1.2. Motivation

Now, consider a busy hospital ward during peak hours, where a cobot autonomously navigates hallways to deliver medications, greets nurses by name and adapts to their specific instructions. Such functionality is no longer science fiction. At CES 2025[a], prototypes of hospital-assistive robots (Figure 1-3) demonstrated significant advancements in computer vision, natural language understanding, and artificial intelligence, enabling them to learn from dynamic environments, generalize across tasks, and refine their behaviour through experience.

---

[a] *Consumer Electronics Show (CES) in Las Vegas [1]. Enchanted Tools, Mirokai Robots. (2025, January 5). CES Demonstration.*

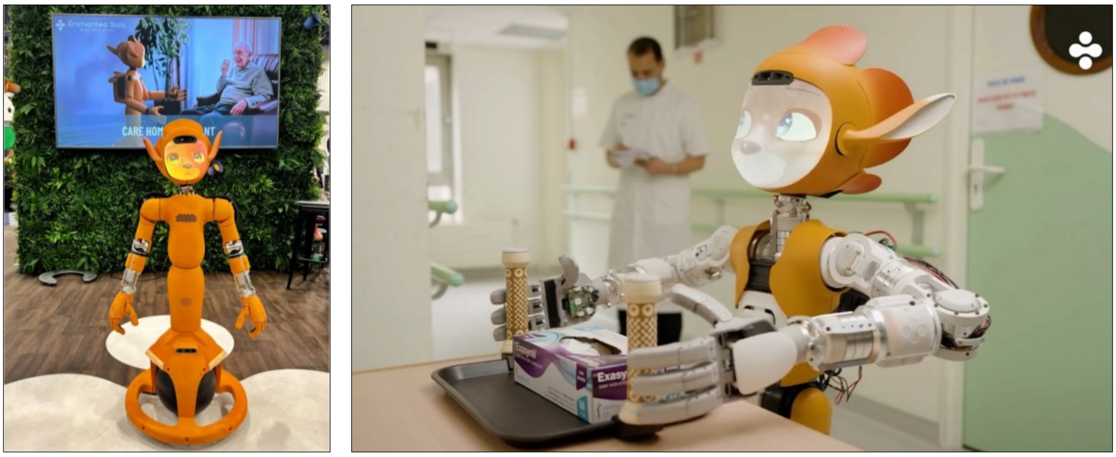


Figure 1-3. Mirokai robot[b], CES 2025: A hospital-assistive robot leveraging vision & language understanding

The convergence of computer vision, natural language processing (NLP), and robotics has no doubt led to systems capable of interpreting instructions such as "*Pick up the red cup*" and autonomously executing them in physical or simulated environments [4]. As Fei-Fei Li, co-director of the Stanford Human-Centred AI Institute and creator of ImageNet, aptly puts it: "*AI is the science of making machines smart; robotics is the art of making them useful*". Her latest venture, World Labs, aims to integrate spatial intelligence into AI systems, further narrowing the gap between perception and action in robotics.

However, truly integrated, context-aware robotic systems remain rare in industrial practice. According to the International Federation of Robotics (IFR), as of 2023, collaborative robots—those designed to work alongside humans—account for only about 10% of industrial robot installations [5]. Supporting this, Eurofound (July 2024) reports that same <10% of robots installed in industrial settings are designed for human interaction, reinforcing the notion that human-interactive systems are still the exception rather than the norm [6]. Industry analysts further observe that "today, robots are still overwhelmingly programmed with traditional methods" (i.e. teach pendants or explicit code). This reveals a persistent gap between the interactive capabilities demonstrated in advanced research prototypes and the reality on factory floors, where natural language interfaces and adaptive behaviours remain largely absent.

On the research front, pioneers like Dr. Cynthia Breazeal, director of the Personal Robots Group at the MIT Media Lab, emphasize that *"the future of robotics hinges on seamless interaction with humans, both in terms of language understanding and personalized adaptation"* [7]. Her work on socially intelligent robots such as Kismet and Leonardo demonstrates the potential for robots to engage in meaningful, emotionally aware interactions through speech, facial cues, and adaptive behaviour [8], [9]. Despite this progress, most current frameworks lack personalization; robots often cannot recall individual preferences, user identities, or spatial configurations specific to a given task environment.

Moreover, recent studies suggest that approximately 73% of industrial robots remain incapable of processing unstructured, natural language commands [10]. This underscores the growing need

---

[b] *Source: Enchanted Tools [2], [3]*

for adaptive, user-centric robotic systems capable of learning from interactions, remembering user-specific preferences, and executing high-level instructions autonomously.

This thesis responds to that need by exploring a database-driven framework that integrates vision-based perception, LLMs, and robotic control within a virtual simulation environment. The system will be designed to interpret generalized spoken or visual instructions, personalize responses by recognizing user identity (via face or voice authentication) recall task preferences, and adjust its behaviour accordingly. By doing so, it seeks to realize the vision of a collaborative robot that is context-aware, user-centric, and capable of safely demonstrating natural language commands in simulation before real-world deployment.

1.3. Background

Developing intelligent robotic systems requires a multidisciplinary approach. This section outlines core technologies and concepts that underpin the proposed framework, including human-robot interaction, computer vision, language models, simulation platforms, and personalization, each contributing to enabling adaptive, user-aware robotic behaviour.

1.3.1. Evolution of Human-Robot Interaction

Industrial robots have traditionally been confined to structured environments, executing pre-programmed, repetitive tasks with high precision. In manufacturing, robotic arms have transformed assembly lines by automating operations such as welding, assembly, and material handling. Historically, these robots operated in isolation from human workers due to safety concerns and were limited to rigid motion paths, lacking any understanding of context or intent.

Today, we have transitioned into an era where robots must function in human spaces, requiring intuitive communication skills and dynamic adaptation. The past decade has seen a paradigm shift in robotics, moving from rigid automation to cobots capable of working alongside humans. This transformation has been driven by advances in three key areas: Computer vision, which enables robots to perceive their surroundings, Large Language Models (LLMs) for interpreting user commands, and Adaptive control techniques, which empower robots to adjust parameters such as grip force, trajectory precision, and timing based on user input.

Despite these advancements, and the growing capabilities of AI-powered robotics, real-world deployment remains limited. An IFR survey has reported that a meagre less than 10% of industrial robots feature advanced human interaction capabilities [5], [10]. WEF[c] has also highlighted AI-driven robotics as a central pillar for future manufacturing, forecasting a 40% increase in collaborative robotics adoption over the next two years. Yet, current systems struggle with unstructured user commands, mostly requiring rigid, structured inputs.

---

[c] *World Economic Forum's annual summit held in Davos-Klosters, Switzerland, in January, 2025 themed "Collaboration for the Intelligent Age" - [11]*

1.3.2. Vision-Based Perception:

For robots to interact effectively with humans then, they must be able to perceive their environment accurately. Computer vision enables robots to detect/recognize objects, track movement or dynamic changes in the environment, and infer spatial relationships.

- Object detection & tracking: Algorithms such as YOLO (You Only Look Once) can identify and locate objects in real time, enabling robots to, for instance differentiate between tools, furniture, and people [12].
- Facial & gesture recognition: Even more advanced vision models allow robots to identify users, recognize gestures or facial expressions, improving interaction and responsiveness.

However, while modern vision models can detect objects with human-level accuracy, they lack contextual understanding, which is a critical requirement for real-world robotics. For example, a robot may recognize a red cup, but does it understand that this specific red cup belongs to Alice and should be handled gently, since it's fragile? Existing vision pipelines treat all objects generically, failing to integrate personalized context.

This thesis tries to address this limitation by integrating vision models with user-specific database memory, enabling a foundational framework for robots to learn and adapt to individual users over time.

1.3.3. Language Understanding with LLMs

Advancements in natural language processing (NLP) have enabled systems that understand and respond to human instructions, utilizing Large Language Models (LLMs) such as OpenAI's GPT, Google's BERT[d], and Meta's LLaMA.

These models can:

- Parse unstructured human commands (e.g., *"Sort the tools like last time."*)
- Translate intent into structured task plans (e.g., *"Pick the wrench and place it on the tray."*)
- Adapt responses based on context and prior interactions

Notably, GPT-4.5 from OpenAI [14] excels at recognizing patterns, drawing connections, and generating creative insights without relying on traditional reasoning. However, LLMs are not inherently grounded in physical reality. Bridging the high-level instructions and the low-level actuator commands required for robotic control is non-trivial, often likened to "*teaching a toddler calculus*" [15]. So, while LLMs excel at understanding human commands, they struggle with real-world task execution.

By exploring the integration of LLM-driven planning with real-time visual perception, this research can enable robots to generate actionable, context-aware task plans, bridging the gap between language comprehension and robotic control.

---

[d] *from the revolutionary paper by Google that increased the State-of-the-art performance for various NLP tasks and set the stepping stone for many other revolutionary architectures [13].*

1.3.4. Robotic Control and Simulation

Achieving safe and reliable robot deployment requires realistic, physics-accurate simulation environments. Modern robotic frameworks rely on:

- The Robot Operating System (ROS): A standardized communication framework for robot control and motion execution [16].
- Simulation platforms like NVIDIA Omniverse: Providing photorealistic virtual grounds for testing and validation, reducing the risks and costs of real-world deployment.

Robotic frameworks like ROS have standardized how robots receive commands and execute motions, while Simulation platforms like NVIDIA Omniverse Isaac Sim provides a high-fidelity physics-based photorealistic simulation environment for testing robotic systems before real-world execution [17]. The synergy between these two elements can be likened to a "*dress rehearsal*," ensuring that each motion plan is viable before it goes live.

Much like a flight simulator for pilots, a virtual environment serves as a risk-free testbed for robots, where errors result in iteration, not injury or equipment damage. Developers can rapidly prototype, debug, and optimize AI-driven task execution pipelines in dynamic scenarios, such as warehouse organization or delicate assembly operations, without downtime or material costs.

Reinforcement Learning (RL) [18] and sim-to-real transfer [19] further enhance the robustness of robot control schemes. Nonetheless, the "*Reality Gap*" remains a challenge we must carefully account for—simulations do not always perfectly reflect real-world physics.

At GTC[e] 2023, NVIDIA unveiled Omniverse's "AI Gym" for training robots—an important step toward addressing the Sim2Real (simulation-to-reality) gap. By leveraging photorealistic physics, GPU-accelerated simulation, and multi-agent AI tooling, platforms like Isaac Sim enable iterative refinement of robotic behaviours. Thus, robotic systems can evolve from *"Error: Cup not found"* to personalized confirmations like *"Coffee's ready: two sugars, as usual."*

1.3.5. Personalization and Ethical Implications

With greater personalization also come ethical and societal considerations. Robots that store user data, such as facial recognition profiles and interaction histories, must be subject to robust privacy safeguards. Google's 2025 robotics privacy framework [22] explicitly recommends minimizing collection of personally identifiable information (PII), and clear transparency about sensor capabilities (cameras, microphones, etc.). As highlighted by a panel at the 2025 iREX (International Robot Exhibition) in Tokyo, "*A robot that recognizes your face shouldn't become a gossip*." This quote underscores the importance of secure data handling and respectful human-robot rapport. This thesis embeds user data in secure databases and would advise best practices for compliance, ensuring that personalization does not undermine trust or privacy in any way.

The upcoming iRex 2025[f], themed "*Sustainable Societies Through Robotics*," further highlights the growing impact of robotics in industry and society with a strong emphasis on human-robot

---

[e] *GPU Technology Conference 2023 hosted by NVIDIA; an annual global AI conference for developers, engineers, researchers, inventors, and IT professionals, gather to discuss the latest innovations in AI, deep learning, graphics, & high-performance computing [20], [21].*

[f] *International Robot Exhibition hosted once every two years in Tokyo, Japan since 1973, and highly regarded as one of the largest robot exhibitions in the world.*

collaboration. As experts advocate for secure data practices and ethical AI, the event reinforces a critical consensus: facial recognition-enabled robots must integrate privacy-by-design principles to safeguard user data and prevent misuse.

Existing research often focuses on isolated components—vision-only models, NLP-driven robotics, or motion planning in controlled environments. However, true AI-powered robotics requires a seamless integration of these elements.

Challenges include:

1. Multi-modal synchronization – merging vision inputs, language processing, and control commands into a cohesive system.
2. Personalization & user adaptation – ensuring robots can learn user preferences over time and recall past interactions.

This thesis tries to address these challenges by proposing an approach, unifying vision, language, and control into a single pipeline, while incorporating a database-driven personalization layer, turning robots from one-size-fits-all tools into attentive collaborators that remember your name, preferences, and past interactions.

1.4. Aims & Goals

The overarching aim of this thesis is to develop an intelligent, user-centric framework that integrates computer vision, large language models (LLMs), and robotic control for personalized, context-aware robotic task execution in virtual environments, thereby enhancing human-robot collaboration.

Specific Goals:

- Develop a vision pipeline for real-time object detection and user identification to enhance task execution accuracy.
- Implement an LLM-powered interface that translates natural language commands into structured, executable task plans.
- Enable the validation of the framework in NVIDIA Omniverse, simulating real-world robot tasks using an industrial robot (ABB YuMi or Universal Robot arms).
- Design a database-driven personalization module, enabling robots to store and recall user preferences and executed tasks for improved interaction quality.

1.5. Research Questions

This thesis aims to answer the following key research questions:

RQ1: How can computer vision and large language models be synergistically integrated to improve task planning and execution accuracy in robotic systems?

RQ2: How can virtual simulation environments (e.g., NVIDIA Omniverse) be leveraged to validate and refine AI-driven robotic control?

RQ3: To what extent does the personalized interaction, using stored user-specific data, enhance task execution efficiency and user satisfaction compared to a generic, non-personalized robotic systems?

1.6. Limitations & Delimitations

While this thesis aims to advance AI-driven robotics, certain limitations must be acknowledged. These limitations highlight areas where additional optimization, validation, or ethical considerations are necessary.

1.6.1. Limitations (external constraints)

1. Interpretability of LLM-based decision making – Large Language Models operate as black-box systems, meaning their decision-making processes are often opaque and difficult to debug. This can lead to unpredictable task plans, necessitating manual validation before execution to ensure safety and efficiency.
2. Computational constraints – Running LLMs, vision models, and real-time control systems requires significant computational resources. High inference times may impact real-time responsiveness, necessitating optimization strategies such as model pruning and distributed processing.
3. Multi-modal integration challenges – Combining computer vision, NLP, and robotic control into a single framework presents inherent synchronization challenges. Ensuring that vision-based perception, LLM-based reasoning, and robotic execution align in real time requires efficient data flow management and low-latency communication protocols.
4. Dataset bias in facial recognition – Facial recognition models, trained on specific datasets, may exhibit biases in user identification, impacting accuracy across different demographics. Addressing this requires dataset diversification and bias-mitigation techniques to ensure fairness in user recognition.
5. Ethical and privacy considerations – Personalization requires storing user interaction data, raising concerns over data privacy and security. To comply with GDPR and ethical AI principles, this research employs local data storage, access control mechanisms, and anonymization techniques to protect user information.

1.6.2. Delimitations (strategic choices)

To ensure focus and feasibility, this study intentionally limits its scope in the following ways:

1. Focus on pick-and-place tasks – The research prioritizes object manipulation and task planning, rather than highly dexterous or complex assembly tasks requiring fine motor skills.
2. Simulation-first approach – Instead of real-world deployment, this thesis validates robotic behaviour in NVIDIA Omniverse, leveraging virtual testing before physical trials.
3. Pre-trained LLMs instead of custom training – Rather than training new language models, the study leverages existing pre-trained models (e.g., Mistral, LLaMA, DeepSeek, or similar

open-source models), applying prompt engineering and fine-tuning techniques for robotic task execution.

4. Controlled user study – The evaluation phase includes a small sample size (3-10 users) to test personalization features. A larger-scale validation study is planned as future work.
5. Privacy-centric design – While the system personalizes robotic interactions, it avoids storing sensitive biometric data long-term, ensuring compliance with ethical AI guidelines.

Despite these constraints, this thesis provides a scalable approach to explore context-aware human-robot interaction, offering a foundation for future research or real-world implementation.

1.7. Expected Outcomes

This research is expected to deliver:

1. A functional prototype, demonstrating an AI-driven robotic system capable of interpreting and simulating user commands in NVIDIA Omniverse.
2. A scalable framework for integrating computer vision, LLMs, and robotic control, applicable to industrial automation, healthcare, and assistive robotics.
3. A database-driven personalization layer, enabling robots to recall past interactions, user preferences, and task histories for enhanced long-term adaptation.
4. Quantitative and/or qualitative evaluation metrics, assessing how personalization impacts task execution success rates, user satisfaction, and adaptability.

1.7.1. Planned deliverables

End-to-End System in simulation

- An integrated functional pipeline integrating computer vision and NLP libraries, LLMs, and robot control scripts, capable of interpreting natural language commands for adaptive task execution, validated through simulated robotic actions in Omniverse Isac Sim.

Scalable vision-LLM integration

- A unified vision-language modules combining computer vision and language parsing to translate commands into robot task plans/execution.
- GitHub repository with code, and detailed documentation for replicating the integration.

Personalization Module

- An SQLite-database-driven GUI implementation for the personalization module, enabling users to view and manage their profiles, interaction histories, and task preferences.
- A recognized user's command e.g. "Re-do yesterday's cubes' pick and drop task" triggers retrieval of prior session data.
- Demonstrated improvement in command interpretation (e.g., recalling user-specific terminology) through user-specific context.

Empirical Validation

- Comparative analysis of personalized vs. generic command interpretation. The comparative analysis will involve toggling the personalization layer on/off while using identical tasks, enabling a direct assessment of the personalization module's impact on task execution efficiency and user satisfaction.
- The personalized system's performance will be evaluated through a user study involving at least five participants, each issuing 10 commands. Metrics such as task planning success rate, frequency of clarification requests, and user satisfaction scores will be compared between personalized and generic systems.

1.8. Key Topics

The key research topics or search terms include:

Vision-language integration in robotics: Vision-language models in robotics bridge object detection (e.g., YOLO) and command parsing (e.g., GPT-4), enabling robots to interpret commands like 'pick up the red cup' both visually and linguistically.

LLMs for robotic command interpretation (e.g., GPT, BERT): Large Language Models (LLMs) like GPT-4 and BERT are used to parse and interpret natural language commands, translating high-level instructions (e.g., *"organize my desk"*) into structured task plans. These models enable robots to manage ambiguous or context-dependent commands, making human-robot interaction more intuitive and efficient.

Simulation platforms for robotic validation (Omniverse, Gazebo): Simulation platforms like NVIDIA Omniverse and Gazebo provide virtual environments for testing and validating robotic systems. These platforms allow researchers to refine control policies, assess task execution, and reduce the reality gap (differences between simulated and real-world performance) before deploying robots in physical environments.

Human-robot interaction (HRI) personalization techniques: Personalization techniques in HRI involve tailoring robotic interactions to individual users based on stored profiles, preferences, and interaction histories. For example, a robot might remember a user's preferred coffee order or frequently used objects, enabling more efficient and context-aware task execution.

Task planning and control for collaborative robots: Task planning and control for cobots involve generating executable task plans from high-level commands and translating them into low-level control signals. This process ensures that robots like ABB YuMi or Universal Robots can safely and efficiently perform tasks in dynamic, human-centric environments.

1.9. Thesis Outline

This thesis is structured into five chapters. Chapter 1 introduces the motivation, problem context, and research objectives. Chapter 2 reviews relevant literature on robotics, natural language processing, and human-robot interaction. Chapter 3 presents the system architecture and core components of the proposed framework. Chapter 4 details the implementation process within a virtual simulation environment. Chapter 5 discusses the experimental results, evaluates the system’s performance, and concludes the thesis with key insights and directions for future work.

2. Literature Review

This chapter reviews existing research on human-robot interaction (HRI), computer vision, large language models (LLMs), database-driven task execution, and simulation frameworks like NVIDIA Omniverse Isaac Sim. It contextualizes the research problem, identifies gaps, and establishes a theoretical foundation for integrating vision, language processing, and robotic control into a unified framework. The review covers personalized robotics, vision-based perception, LLM advancements, database-driven execution, and simulation frameworks for validating robotic control before deployment.

2.1. Theoretical Framework

In recent years, the field of robotics has undergone a transformative evolution, driven by rapid advancements in artificial intelligence (AI), particularly in the domains of computer vision and natural language processing (NLP). These technologies have enabled the development of robotic systems that can perceive their environment with remarkable accuracy, understand and respond to human language, and execute complex tasks autonomously. These capabilities are essential for applications such as industrial automation and healthcare, as well as domestic assistance and education, where robots must interact naturally and efficiently with humans. However, the true potential of robotics lies not only in the individual strengths of these technologies but in their seamless integration. Combining computer vision, large language models (LLMs), and advanced robot control techniques, modern robotics can create intelligent systems that understand user intents, adapt to individual preferences, and perform tasks in a personalized and context-aware manner.

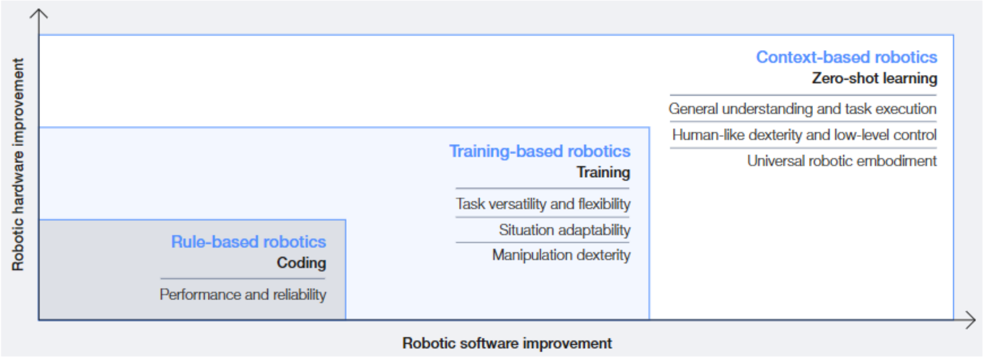
Through this theory, we aim to identify gaps in current research - such as limitations in real-time integration, scalability of personalization, and fidelity of simulation environments - and establish a theoretical foundation for the proposed thesis. By synthesizing foundational and up to recent research findings from various domains, this chapter builds a case for the proposed research, highlighting the necessity of integrating vision, language processing, and robot control in a modular framework for personalised task execution to enhance human-robot collaboration.

2.2. Evolution of Cobots: from Scripted to Adaptive

The quest to create robots capable of natural human interaction has progressed through *three* distinct eras [23]. Early industrial robots operated in strictly structured environments following rigid pre-programmed routines of commands, with minimal human intervention [24]. While the second-generation systems incorporated basic sensor feedback for adaptive grasping [25], they still followed preprogrammed trajectories with limited adaptability. These robotic systems were traditionally designed for efficiency in manufacturing lines, such as robotic arms used in car assembly plants.

The current cognitive robotics paradigm, marked by the emergence of collaborative robots (cobots), such as ABB's YuMi and Universal Robots' UR series, marked a significant shift in robotics. Unlike traditional industrial robots, cobots are designed to safely interact with humans, featuring real-time sensors, adaptive controls, and AI-driven decision-making. Research in HRI emphasizes intuitive interactions, enabling robots to understand and respond to user intentions using speech, gestures, and vision-based cues [26].

The AI revolution is ushering in a new era in robotics. Embodied AI integrates intelligent algorithms into physical systems, enabling robots to perceive and interact with their environments through dynamic movements. Three primary robotic systems have emerged—rule-based, training-based, and context-based (Figure 2-1). These agents use sensors (e.g., cameras, microphones) to observe the world and actuators like advanced grippers to perform actions. Applied to industrial operations, these agents enable more sophisticated automation overcoming traditional challenges such as those associated with handling unstructured environments or manipulating unstable objects [27].



Robotic hardware improvement
Context-based robotics
Zero-shot learning
General understanding and task execution
Human-like dexterity and low-level control
Universal robotic embodiment
Training-based robotics
Training
Task versatility and flexibility
Situation adaptability
Manipulation dexterity
Rule-based robotics
Coding
Performance and reliability
Robotic software improvement

Figure 2-1. Robot capabilities enabled by AI. - Boston Consulting Group

Furthermore, recent studies highlight the need for personalization in HRI, where robots not only follow generalized instructions but also understand contextual nuances and adapt to individual users' preferences, roles, and past interactions [28].

The evolution from rigid industrial automation where robots performed repetitive tasks in isolated settings, to knowledge-driven personalised robotics, where cobots work alongside humans in dynamic environments have proven crucial in domains such as assistive healthcare [29], collaborative manufacturing, and household robotics.

2.3. Vision Systems: Object Detection to User-Centric Context

> *"A camera that sees a 'red box' is useful; one that recognizes 'John's fragile red box' is revolutionary."*

Modern robotic vision systems excel at detecting objects but remain oblivious to the *who* and *why* behind tasks. This section critiques the field's fixation on generic perception and argues for a paradigm shift toward user-centric context

2.3.1. Seeing the world through cameras

Modern robotic vision systems have evolved from static object detection pipelines to dynamic scene interpreters. These advancements in real-time object detection (YOLO11) [30], [31], [32] and pose estimation (OpenCV, NVIDIA DeepStream) have transformed how robots interpret their environments. Algorithms like You Only Look Once (YOLO) have set benchmarks by processing images at remarkable speeds - *YOLOv6-3.0 achieves 142 FPS inference speeds on NVIDIA Jetson platforms while maintaining 52.3% mAP accuracy* [32]. Such real-time performance enables cobots like ABB YuMi to process 360° RGB-D streams at 30Hz, detecting objects within ±3mm positional accuracy, for identification and location of objects in applications ranging from warehouse logistics to surgical robotics [31], [33]. However, raw detection metrics only tell half the story.

Pose estimation further enhances a robot's interaction with its surroundings by determining the orientation and position of objects. Techniques utilizing OpenCV facilitate real-time pose estimation, allowing robots to infer not just *what* objects exist, but *how* they relate spatially - a prerequisite for executing tasks requiring precision and adaptability.

Despite these technological strides, many vision pipelines perceive objects as generic entities, devoid of user-specific meaning.

> *"A 'red box' detected by YOLO lacks context - is it fragile? Who owns it? How was it handled last time?".*

Amazon's Astro robot exemplifies this limitation: while it can recognize faces, it doesn't adjust its tasks based on the user's expertise or history [34].

Object detection [35] affordance recognition [36], [37], [38], and facial recognition such as in [39] are all crucial for contextual understanding: the first helps locate items, the second infers how to interact with them, and the third enables personalized user interactions. While these capabilities have advanced individually, integrating them into a cohesive robotic vision pipeline, especially in dynamic and occluded environments, remains an ongoing frontier in research.

By integrating *scene data* with *database-driven user identity*, a system can enrich object metadata with user history. For instance, detecting '*John's red box*' triggers a pre-stored preference for low-force grip handling, avoiding the one-size-fits-all approach of existing frameworks.

2.3.2. Biometrics as the first step to personalization

Biometric authentication -face recognition and voice processing- has matured into a reliable tool for user identification, from security checkpoints to collaborative robotics. Yet, some current implementations treat biometrics as mere "keys" to access systems, failing to leverage them for *collaborative* or *adaptive* applications.

By employing face and voice recognition technologies [40], robots can identify users and access individualized profiles, paving the way for customized responses. For example, Sensory Inc. has developed embedded face and voice biometrics that provide secure and convenient user verification, enhancing the personalization of interactions [40].

By linking biometric authentication to an SQLite database of user profiles, robots can recall past interactions and preferences. For instance, recognizing 'Rahul' via face authentication retrieves his stored preference for slower pick-and-place task execution, while voice commands like 'Handle carefully' update the robot's torque limits in real time. This fusion of biometrics, scene data, and historical context enables truly personalized execution.

In summary, advancing from basic object detection to user-centric context requires integrating sophisticated scene understanding with biometric-driven personalization. While existing tools like OpenCV or ROS's vision pipelines excel at *what* and *where*, a database-driven architecture could answer *for whom* - a leap from robotic perception to collaborative understanding.

2.4. Language Models: Translating Intent into Actionable Code

> *"What if Siri could not only set a timer but also teach a robot to assemble your IKEA furniture?"*

2.4.1. NLP in robotics: "syntax to semantics"

The advent of transformer-based large language models (LLMs) like GPT-4, BERT, and LLaMA for natural language processing (NLP), reasoning and language generation has redefined human-robot interaction. These models act as neural compilers, translating unstructured linguistic inputs into structured task representations through three computational phases: *syntactic parsing* (identifying verbs/nouns), *contextual grounding* (e.g., mapping "left panel" to CAD-defined part IDs), and *code generation* (producing ROS-compatible motion [41]. Recent work demonstrates their surprising adaptability: *ChatGPT*, designed for text generation, now generates robot control code[42], while domain-specific models like *PaLM-E* integrate vision and language for embodied reasoning [43]. Yet, as the *2023 LLM HRI Review* notes, these systems often operate in isolation - parsing language without grounding it in real-time sensor data or user history.

2.4.2. Limits of LLMs in embodied robotics

Despite their prowess, the application of LLMs in embodied robotic systems face three unique challenges:

Ambiguity Resolution in Instructions:

LLMs often struggle to interpret ambiguous or underspecified human instructions in a robotics context. Different LLMs can produce divergent behaviours when given the same ambiguous command, highlighting inconsistency in understanding intent [44].

For example, an instruction like *"put the block in the bowl you think it best fits"* leaves too much subjectivity—one model might choose a different bowl than another. This issue is often described as a problem of *abstraction matching*, meaning the robot's system must align an ambiguous natural-language request with a precise, unambiguous action sequence [45].

In practice, current LLM-driven robots still find it challenging to reliably disambiguate instructions without additional contextual clues or clarifications, underscoring the need for better grounding of language in robotic perception and knowledge. Proper handling of ambiguity is crucial; otherwise, the robot's actions may not match the user's intent due to misinterpretation.

Temporal Context and Sequential Reasoning:

LLMs often struggle with real-time processing requirements and the integration of multimodal sensory inputs, such as visual and tactile data, which are essential for accurate task execution in robotics.

> *While chatbots retain conversational memory (e.g., ChatGPT's chat history), robotic systems must also track state changes across interactions.*

Robotic tasks typically unfold over a sequence of steps and time, which poses problems for LLM-based controllers in maintaining temporal context. One limitation is that LLMs generate actions in an auto-regressive (step-by-step) manner—errors can compound over time. A mistake or misunderstanding in an early step becomes part of the context for subsequent steps, potentially derailing the entire plan.

Moreover, studies have found that LLM-generated plans sometimes execute actions out of order or with improper sequencing, reflecting difficulty in reasoning about event order. For instance, a robot might skip a necessary preparatory action or perform an irrelevant step because the model failed to connect it to the earlier context. This indicates that while LLMs can break down tasks into steps, they do not always capture the correct temporal dependencies. Ensuring that a robot understands what to do *first*, *next*, and *last* (and how to adjust if something changes mid-task) remains an open challenge. Researchers are addressing this by incorporating feedback loops and temporal logic checks, but LLM-based systems still require improvements in context retention and temporal reasoning to handle long-horizon, dynamic tasks reliably.

Actionability and Grounding of Language:

Another significant challenge for applying LLMs in embodied robots is translating language output into feasible, real-world actions.

> *LLMs "hallucinate" - a phenomenon where LLMs generate plausible-sounding but incorrect or unfeasible actions.*

This issue arises because LLMs, trained predominantly on text data, may lack grounding in the physical realities of robotic capabilities and environmental constraints. For example, a plan might instruct the robot to navigate to a non-existent location or pick up an object that isn't present.

Such instructions reveal a lack of grounding; the model does not fully grasp the robot's physical context or limitations. Even when the described actions are theoretically possible, they might not be *directly executable* without further translation. For instance, an LLM might output a high-level goal (*'tidy the room'*) or a vague action (*'fix the disorganized shelf'*) that the robot cannot act on without decomposition into its low-level motor commands. Researchers have noted that LLM-based planners need interfaces (such as skill libraries or code-generation layers) to convert abstract language into concrete robot behaviours, but this translation is brittle and often requires careful prompt design or model fine-tuning.

In summary, ensuring *actionability* means the robot's interpretation of an LLM's output aligns with what it can do in its environment. Overcoming this gap involves better world-model grounding for LLMs and stricter checks on the feasibility of suggested actions, so the robot doesn't attempt steps outside its capabilities or environment state.

2.4.3. Conversational memory: “chatbots to cobots”

In the realm of chatbots, conversational memory enables systems to maintain context over interactions, providing coherent and contextually appropriate responses. Translating this capability to collaborative robots (cobots) involves more than just retaining dialogue history; it requires the integration of past interactions, user preferences, and environmental context into the robot’s task planning and execution processes. This deep integration allows cobots to anticipate user needs, personalize task execution, and improve efficiency over time. For instance, a cobot equipped with conversational memory can recall a user’s preferred methods for assembling furniture, thereby streamlining the process in future tasks. Implementing such systems demands robust data management and real-time processing frameworks to handle the complex interplay of language, memory, and action in dynamic settings.

In summary, standalone LLMs remain *language-bound* – their brilliance in text generation often falters when confronted with the *physics-aware* demands of robotic systems. This gap is bridged through hybrid architectures that fuse LLM outputs with real-time visual data and user-specific context, enabling robots to adapt to dynamic environments and user-specific preferences, moving beyond generic responses to tailored solutions.

2.5. Robot Task Planning & Execution in Virtual Environments

2.5.1. Robot control in virtual environments

Virtual environments have become indispensable tools for designing, testing, and validating robot control strategies prior to real-world [46]. These platforms offer photorealistic and physics-accurate simulations, reducing the necessity for physical prototyping and accelerating iteration cycles in robotics research. For instance, NVIDIA Omniverse Isaac Sim provides a scalable and extensible simulation environment that supports various sensors and robot models, enabling comprehensive testing of AI-driven robotics solutions.

Challenges from use of Simulation tools

Simulation is the process of using a computer to approximate how a dynamic system (here a robot) evolves in time, and as has been poignantly observed, “Simulation is doomed to succeed”, [47], [48] provides an overview of the concept and role of physics-based simulation in designing intelligent robots.

[49] identified 10 high-level challenges associated with the use of simulation and outlined ideas on how to tackle these challenges to unlock the full potential of simulation-based testing. These challenges include the gap between simulation and reality, a lack of reproducibility, and considerable resource costs associated with using simulators.

2.5.2. Modular architectures for control

Modern robot control architectures increasingly favour modular frameworks that separate perception, planning, and actuation. Within such architectures, the control component is typically responsible for generating collision-free trajectories and closed-loop feedback corrections based on sensor inputs. Open-source software like the Robot Operating System (ROS) and its MoveIt framework exemplify this modular principle, offering a suite of libraries for motion planning, inverse kinematics, and real-time control. Recent advancements have extended these concepts into GPU-accelerated simulation environments, such as NVIDIA Isaac Sim, to support faster-than-real-time iterations of planning and control strategies.

2.5.3. Task planning & execution frameworks

A fundamental aspect of robotics research involves ensuring that robots can break down high-level goals into discrete, executable actions - colloquially referred to as task planning and execution. Classical AI planners like STRIPS [50] laid the groundwork for symbolic planning; more recent efforts integrate classical symbolic methods with modern AI techniques, particularly Large Language Models (LLMs) to enhance task planning by translating natural language instructions into symbolic representations, thus improving the generalization to new tasks, addressing challenges in task planning and execution in unstructured environments [51].

{PLACE HOLDER}

Review of approaches for action planning and execution in simulation environments.

generalization to new tasks and reducing the search space for planning (Tang et al., 2025). This integration allows robots to perform complex actions in unstructured environments, demonstrating high success rates in various task complexities (Silva et al., 2024).

A recent literature [52] proposes a TAsk Planing Agent (TaPA) in embodied tasks for grounded planning with physical scene constraint, where the agent generates executable plans according to existing objects in the scene by aligning LLMs with the visual perception models.

2.5.4. Database-driven personalization in HRI

{PLACE HOLDER}

Review of database methods for storing data for robot tasks.

2.6. Integration & challenges

The integration of computer vision (CV), large language models (LLMs), and adaptive control enable systems like the ABB YuMi cobot to operate in unstructured environments while maintaining industrial-grade precision - a capability fundamentally dependent on closing the loop between environmental perception, cognitive processing, and physical actuation.

While [6] merged vision and language for robotic manipulation, their system treated all users as strangers.

{PLACE HOLDER}

More about:

- retrieval-augmented generation (RAG) techniques show promise in enhancing LLMs' abilities to understand and generate structured queries from unstructured text inputs.

3. Methodology

This chapter presents the methodology for the proposed framework. It details the research design, system architecture, data collection, implementation, evaluation, and ethical considerations, ensuring rigor and reproducibility. Each section details key components, including user authentication, input processing, task planning, robotic execution, integration workflow, and validation, supported by relevant literature.

3.1. Techniques & Tools

3.2. System Architecture & Framework

3.3. Data Collection, Pre-processing & Flow

3.4. System Integration & Implementation Steps

Having established the core modules—Authentication, Vision & Perception, Input Processing, Language Understanding, Task Planning & Control, and Simulation—in Section 3.1, this subsection details the end-to-end workflow of the proposed system, emphasizing how these components interact in practice. A central Task manager/GUI serves as the main orchestrator (via a lightweight Python-based GUI, *tkinter*), coordinating the flow from user authentication through task simulation and final feedback.

These integration steps also reference the corresponding Python scripts (e.g., *face_auth.py*, *voice_auth.py*, *command_processor.py*, *camera_vision.py*, *gesture_processor.py*, *orchestrator.py*, *synchronizer.py*) to illustrate exactly how data and control signals flow within the system.

3.4.1. Central Task Manager / GUI

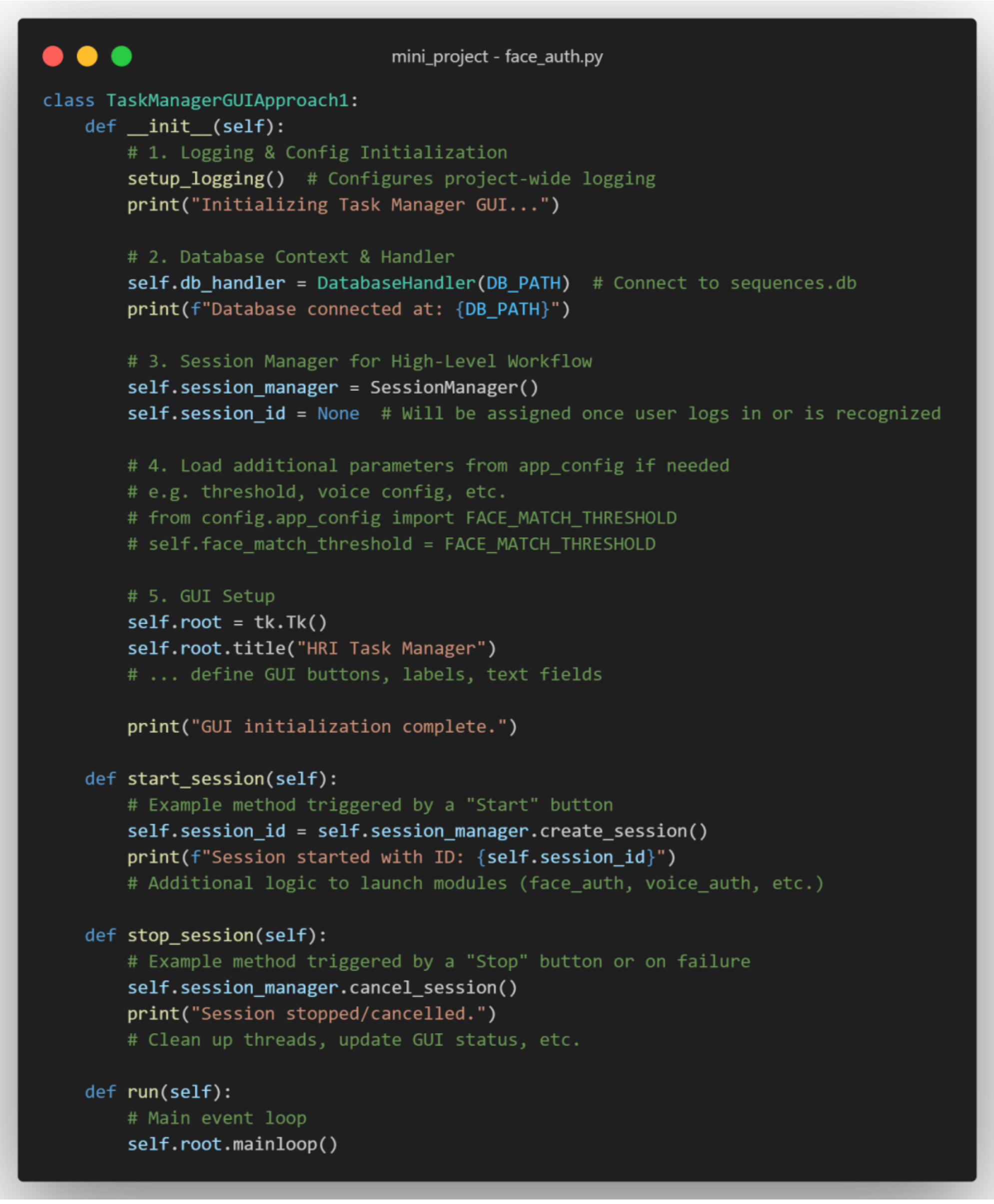
A lightweight GUI controller, Task Manager (see *task_manager.py* and related GUI files) functions as the central orchestrator, initializes the system, sets up the database context, and loads essential configurations, coordinating logic across all components:

1. Purpose
    - A central orchestrator (see *orchestrator.py*) manages session states, module activation, and inter-module data flow. It also ensures fault tolerance by handling re-authentication, re-planning, or fallback options when errors occur (e.g., unrecognized voice, ambiguous user command).

- It routes user profile data from the Authentication module to the Input Processing module, ensures multi-modal inputs (voice, gesture) are unified, and finally delivers structured task plans to the Task Planning and Simulation modules.

2. Implementation

- <u>Session Management</u>: Each new user session is identified by a unique session ID, generated after successful face or voice authentication. The orchestrator stores this ID and relevant user context in the database (tables like *users*, *interaction_memory*).
- <u>Multi-Module Coordination</u>: The orchestrator initiates processes in the Authentication (face/voice) pipeline, triggers Vision & Perception updates, collects instructions from Input Processing, and finally pushes validated commands to the Task Planning module for Omniverse simulation.



```python
class TaskManagerGUIApproach1:
    def __init__(self):
        # 1. Logging & Config Initialization
        setup_logging()  # Configures project-wide logging
        print("Initializing Task Manager GUI...")

        # 2. Database Context & Handler
        self.db_handler = DatabaseHandler(DB_PATH)  # Connect to sequences.db
        print(f"Database connected at: {DB_PATH}")

        # 3. Session Manager for High-Level Workflow
        self.session_manager = SessionManager()
        self.session_id = None  # Will be assigned once user logs in or is recognized

        # 4. Load additional parameters from app_config if needed
        # e.g. threshold, voice config, etc.
        # from config.app_config import FACE_MATCH_THRESHOLD
        # self.face_match_threshold = FACE_MATCH_THRESHOLD

        # 5. GUI Setup
        self.root = tk.Tk()
        self.root.title("HRI Task Manager")
        # ... define GUI buttons, labels, text fields

        print("GUI initialization complete.")

    def start_session(self):
        # Example method triggered by a "Start" button
        self.session_id = self.session_manager.create_session()
        print(f"Session started with ID: {self.session_id}")
        # Additional logic to launch modules (face_auth, voice_auth, etc.)

    def stop_session(self):
        # Example method triggered by a "Stop" button or on failure
        self.session_manager.cancel_session()
        print("Session stopped/cancelled.")
        # Clean up threads, update GUI status, etc.

    def run(self):
        # Main event loop
        self.root.mainloop()
```

Figure 3-5. Code snippet - showing the TaskManagerGUI as an entry point.

3. Error and Recovery

- Authentication Failures: When authentication or recognition fails (e.g. distance above threshold), the orchestrator prompts *face_auth.py* or *voice_auth.py* for re-capture or fallback procedures. If it consistently fails, the user can register anew.
- Ambiguous Input: If the LLM returns an incomplete plan (e.g., missing object_name), the orchestrator queries the user for clarification ("*Which object did you intend?*").
- Simulation Failures: If tasks fail in simulation, it requests re-planning from the LLM or re-checks the database for alternative references.

3.4.2. Implementation Steps

The system follows a structured, closed-loop sequence (see Figure 3-6). Each step is designed to minimize latency and maximize reliability, ensuring a smooth user experience:

Approach:

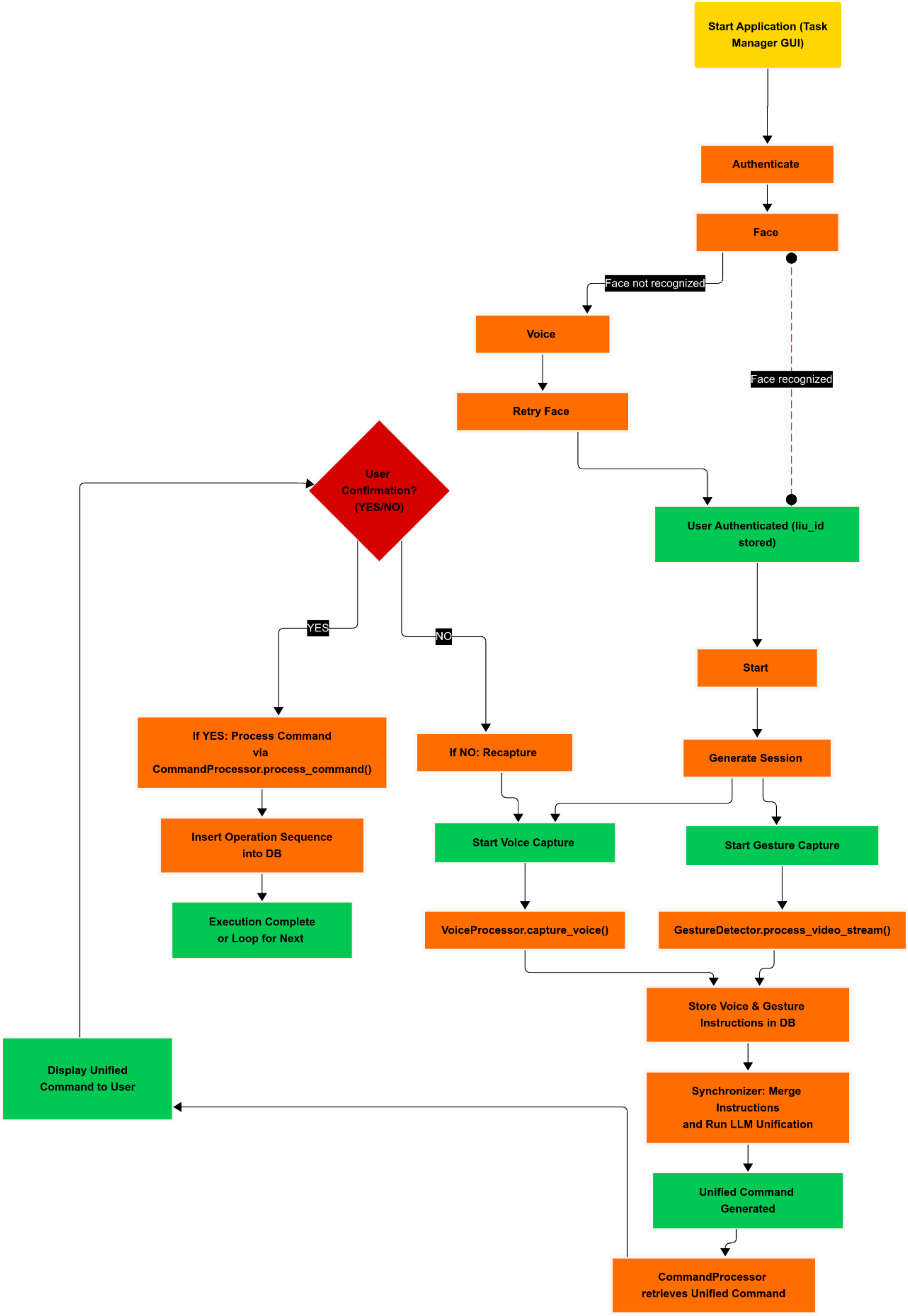

Start Application (Task Manager GUI)
Authenticate
Face
Face not recognized
Voice
Retry Face
Face recognized
User Authenticated (liu_id stored)
User Confirmation? (YES/NO)
YES
NO
Start
If YES: Process Command via CommandProcessor.process_command()
If NO: Recapture
Generate Session
Insert Operation Sequence into DB
Start Voice Capture
Start Gesture Capture
Execution Complete or Loop for Next
VoiceProcessor.capture_voice()
GestureDetector.process_video_stream()
Store Voice & Gesture Instructions in DB
Display Unified Command to User
Synchronizer: Merge Instructions and Run LLM Unification
Unified Command Generated
CommandProcessor retrieves Unified Command

Figure 3-6: Operational workflow – Approach I[5]

---

[5] *Approach II can be found in the appendix section of this report*

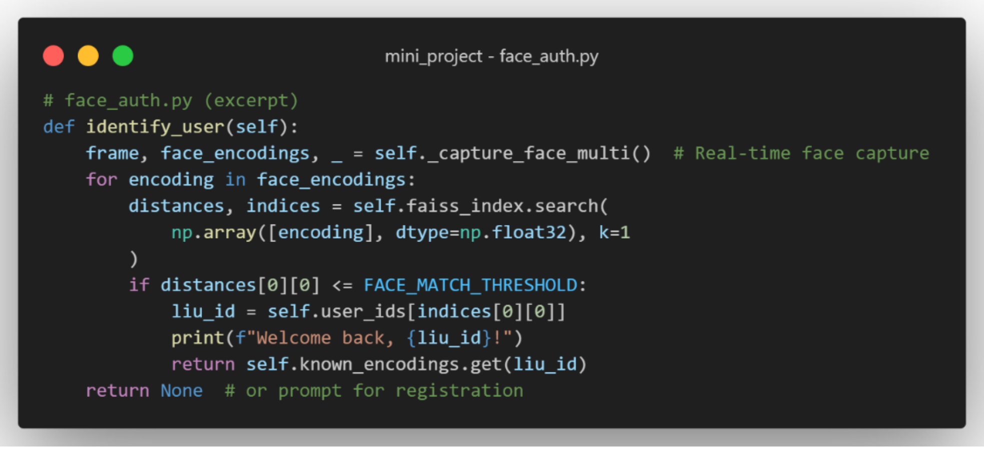
Step 0: Initialization & Database Context

On startup, the TaskManagerGUI class initializes a session via *session_manager.py*, setting up database connections and loading essential configurations (from *app_config.py*) for face matching thresholds, voice processing, and more.

- The *task_manager.py* starts the user interface, handles start/stop sessions, user interface interactions, and ensures each module runs in the correct order.
- The SQLite database (*sequences.db*) through the database handler (*db_handler.py*) is ready to read/write user data, face/voice embeddings, and recognized object details (e.g., in *camera_vision*, *voice_instructions*).
- If the user cancels a session or if a module fails (e.g., face authentication times out), the Task Manager gracefully closes threads and updates the GUI status.

Step 1: User Authentication → Database Synchronization

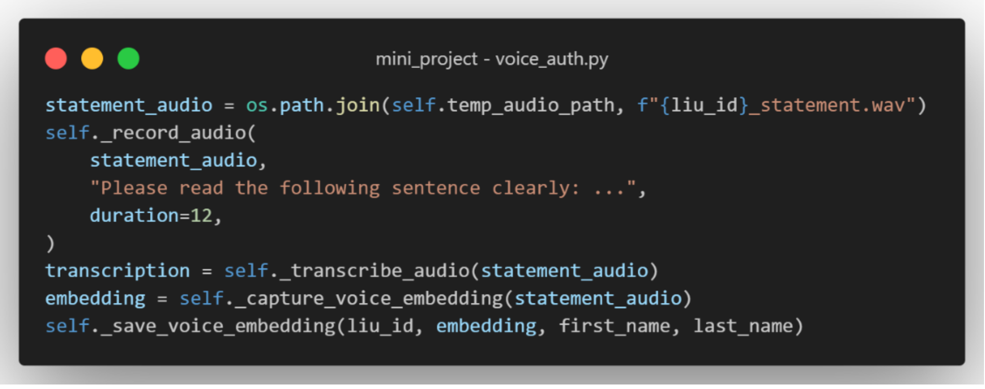
Upon session start or user arrival, the Authentication module verifies identity using face or voice, retrieving user details from the database. Figure 3-7 show how the system attempts to identify an incoming user. It queries the database for stored face encodings, compares them with a newly captured encoding, and sets an authenticated status if a match is found, while Figure 3-8 shows the user registration process from *voice_auth.py*, capturing a short voice sample and storing embeddings. This ensures an optional second authentication factor or fallback method if face recognition fails.



```python
# face_auth.py (excerpt)
def identify_user(self):
    frame, face_encodings, _ = self._capture_face_multi()  # Real-time face capture
    for encoding in face_encodings:
        distances, indices = self.faiss_index.search(
            np.array([encoding], dtype=np.float32), k=1
        )
        if distances[0][0] <= FACE_MATCH_THRESHOLD:
            liu_id = self.user_ids[indices[0][0]]
            print(f"Welcome back, {liu_id}!")
            return self.known_encodings.get(liu_id)
    return None  # or prompt for registration
```

Figure 3-7. Excerpt from face_auth.py showing facial recognition approach

- FaceAuth class (*face_auth.py*) or VoiceAuth class (*voice_auth.py*) captures user biometric data. Once validated (e.g., face distance < *threshold FACE_MATCH_THRESHOLD* = 0.6), the user's profile is loaded from the database (e.g., role, preferences, face encodings, voice embeddings).



mini_project - voice_auth.py

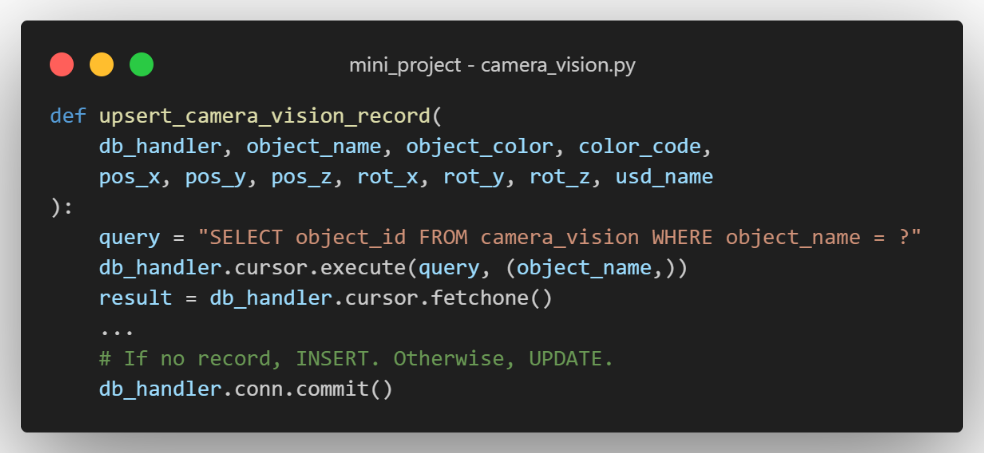
```
statement_audio = os.path.join(self.temp_audio_path, f"{liu_id}_statement.wav")
self._record_audio(
    statement_audio,
    "Please read the following sentence clearly: ...",
    duration=12,
)
transcription = self._transcribe_audio(statement_audio)
embedding = self._capture_voice_embedding(statement_audio)
self._save_voice_embedding(liu_id, embedding, first_name, last_name)
```

Figure 3-8. Excerpt showing how the user voice is captured for registration

- Outcome: A recognized user with a session ID. If new, face_auth.py can prompt for registration, storing face encodings or voice prints in the DB via DatabaseHandler(see *db_handler.py*).

Step 2: Vision & Perception → camera_vision

The system next gathers scene information so that user instructions (e.g., "pick the red slide") can be matched to actual objects. Additional scripts like *tray_and_holder_detection.py* can refine bounding box data or apply colour segmentation to track specialized items. The recognized objects (e.g., "holder," "slide_1") are essential for the LLM's final command interpretation.



mini_project - camera_vision.py

```
def upsert_camera_vision_record(
    db_handler, object_name, object_color, color_code,
    pos_x, pos_y, pos_z, rot_x, rot_y, rot_z, usd_name
):
    query = "SELECT object_id FROM camera_vision WHERE object_name = ?"
    db_handler.cursor.execute(query, (object_name,))
    result = db_handler.cursor.fetchone()
    ...
    # If no record, INSERT. Otherwise, UPDATE.
    db_handler.conn.commit()
```

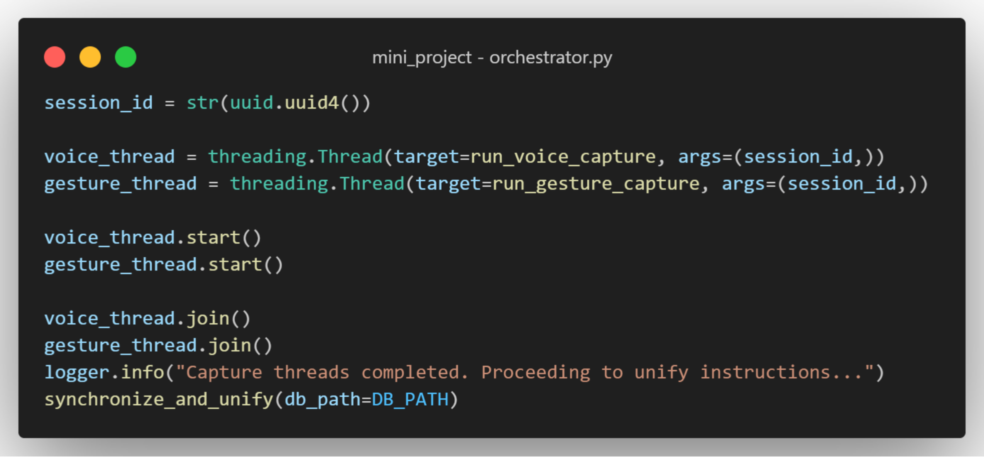
Figure 3-9. Camera vision script updates the properties of detected objects in the database.

- Vision (*camera_vision.py*) processes frames from a (real or simulated) camera. Object detection is done via colour-based segmentation, YOLO, or a hybrid approach. Identified objects (e.g., "red slide," "tray," "blue cube") are logged in camera_vision table with attributes like pos_x, pos_y, pos_z, rot_z, color_code.
- §Outcome: Real-time mapping of scene objects in the database, accessible for later command referencing by the LLM.

Step 3: Multi-Modal Input Processing

After authentication, real-time voice and gesture inputs are captured and queued for merging. The Whisper-based transcription and MediaPipe gesture recognition run in parallel, feeding text + gesture data to the orchestrator. Each modality logs recognized content in the database.

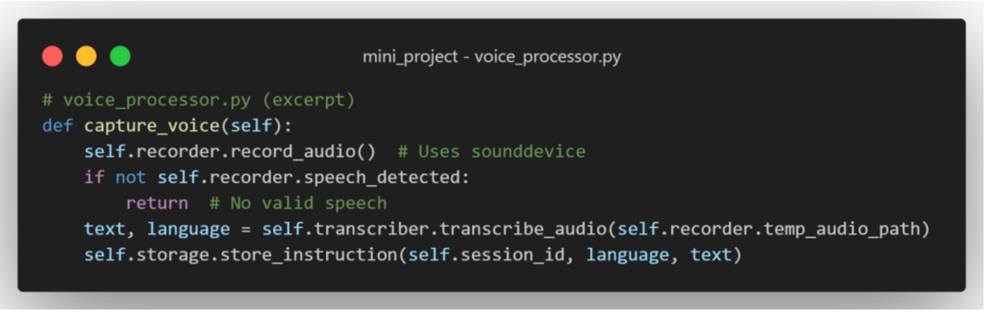
- Orchestrator (*orchestrator.py*) spawns capture threads for voice and gesture processing, merges these inputs, and, if needed, triggers re-authentication or re-planning sequences.



```python
session_id = str(uuid.uuid4())

voice_thread = threading.Thread(target=run_voice_capture, args=(session_id,))
gesture_thread = threading.Thread(target=run_gesture_capture, args=(session_id,))

voice_thread.start()
gesture_thread.start()

voice_thread.join()
gesture_thread.join()
logger.info("Capture threads completed. Proceeding to unify instructions...")
synchronize_and_unify(db_path=DB_PATH)
```

Figure 3-10. Threads concurrently run for command capture from the orchestrator script

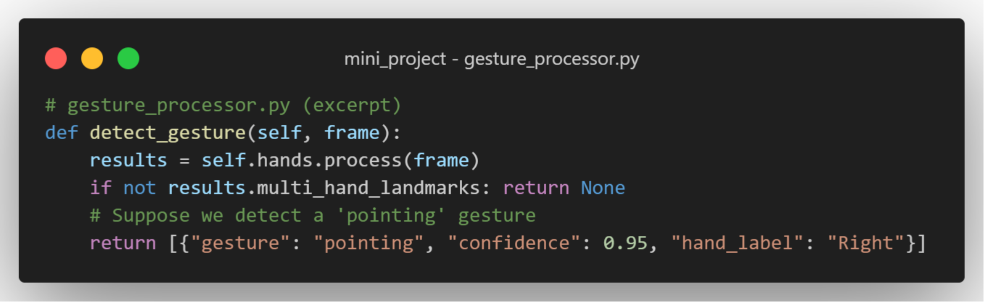
- Voice commands: Recorded with sounddevice and transcribed by Faster Whisper. record_audio() blocks until speech is detected or times out. Transcription yields a text command that is stored in voice_instructions table in DB.



```python
# voice_processor.py (excerpt)
def capture_voice(self):
    self.recorder.record_audio()  # Uses sounddevice
    if not self.recorder.speech_detected:
        return  # No valid speech
    text, language = self.transcriber.transcribe_audio(self.recorder.temp_audio_path)
    self.storage.store_instruction(self.session_id, language, text)
```

Figure 3-11. Code snippet showing voice command capture

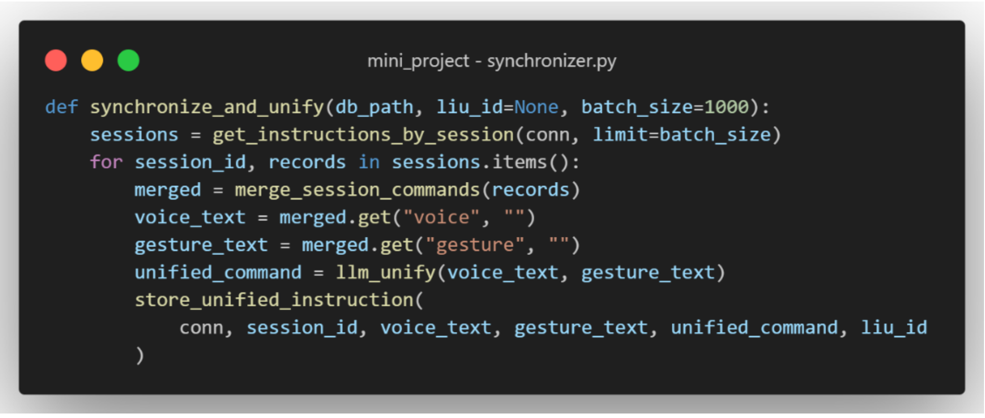
- Gesture cues: Detected in *gesture_processor.py* using MediaPipe Hands. Identified gestures (e.g., “pointing,” “thumbs_up”) are stored in gesture_instructions, along with a timestamp and confidence score.



```python
# gesture_processor.py (excerpt)
def detect_gesture(self, frame):
    results = self.hands.process(frame)
    if not results.multi_hand_landmarks: return None
    # Suppose we detect a 'pointing' gesture
    return [{"gesture": "pointing", "confidence": 0.95, "hand_label": "Right"}]
```

Figure 3-12. Code snippet showing gesture cue capture

- Synchronization: The orchestrator calls synchronizer.py to merge voice text and gesture data (e.g., “Pick up that block” + pointing direction). Then the unified command is appended to unified_instructions and passed to the LLM-based command processor (*command_processor.py*).

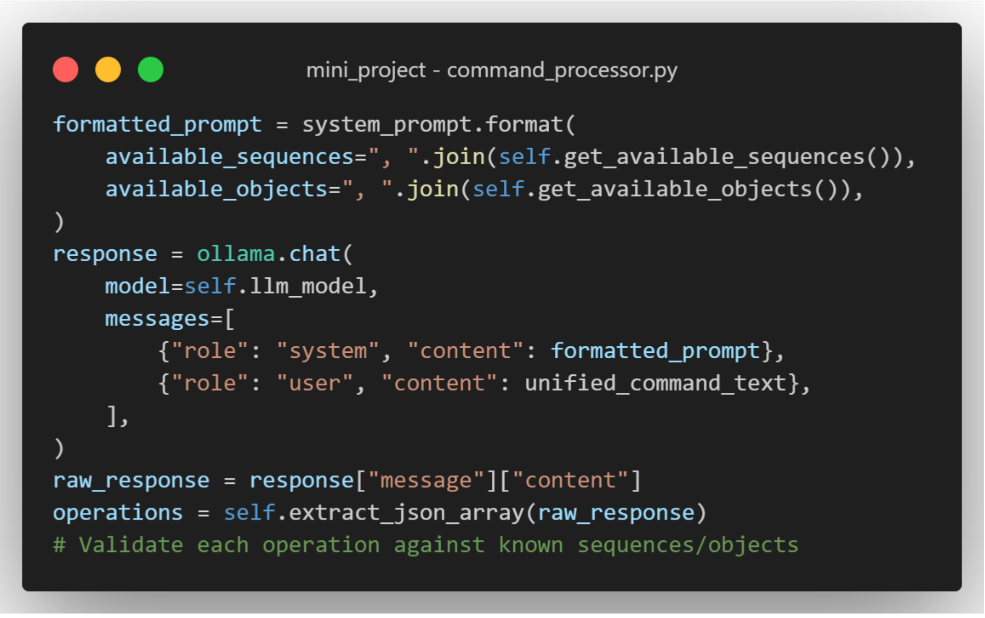


```
def synchronize_and_unify(db_path, liu_id=None, batch_size=1000):
    sessions = get_instructions_by_session(conn, limit=batch_size)
    for session_id, records in sessions.items():
        merged = merge_session_commands(records)
        voice_text = merged.get("voice", "")
        gesture_text = merged.get("gesture", "")
        unified_command = llm_unify(voice_text, gesture_text)
        store_unified_instruction(
            conn, session_id, voice_text, gesture_text, unified_command, liu_id
        )
```

Figure 3-13. Excerpt from synchronizer.py combining concurrent voice and gesture inputs

Step 4: Language Understanding → LLM Command Processing

- *CommandProcessor* class (*command_processor.py*) prompts a large language model (e.g., Mistral or Llama) using a specialized prompt template containing user context, recognized objects, and the unified command.



```
formatted_prompt = system_prompt.format(
    available_sequences=", ".join(self.get_available_sequences()),
    available_objects=", ".join(self.get_available_objects()),
)
response = ollama.chat(
    model=self.llm_model,
    messages=[
        {"role": "system", "content": formatted_prompt},
        {"role": "user", "content": unified_command_text},
    ],
)
raw_response = response["message"]["content"]
operations = self.extract_json_array(raw_response)
# Validate each operation against known sequences/objects
```

Figure 3-14. Code snippet showing LLM prompt, utilizing objects and sequences from database

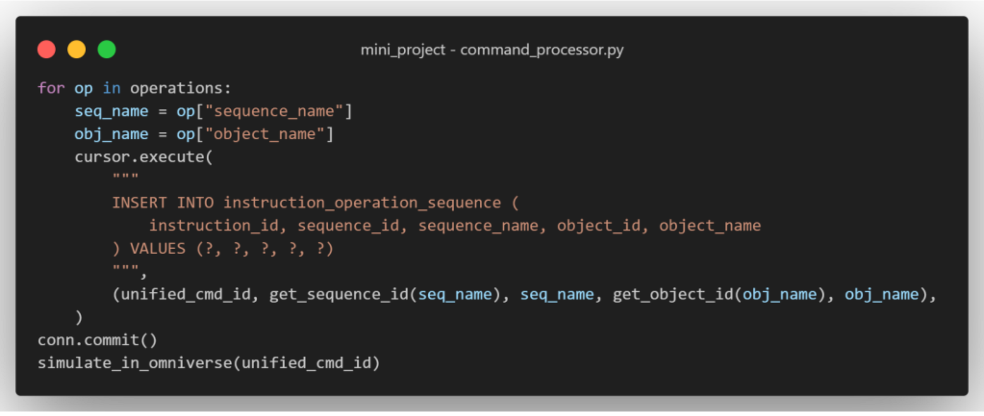
- Outcome: A structured JSON-based task sequence (e.g., [{"sequence_name": "pick","object_name":"slide_2"}, specifying each subtask (e.g., "*pick*," "*travel*," "*drop*"), tied to recognized objects from the DB.

Step 5: Task Planning & Robotic Control

Once the structured tasks are validated, the orchestrator or the *command_processor.py* references the skill library in the DB (e.g., *sequence_library*) to map tasks to robot actions. If any object or skill is invalid, it queries the DB or requests user clarification.

- The orchestrator hands the validated JSON to the Task Planning module (part of *command_processor.py* or a dedicated script in *mini_project/workflow*). It matches "*pick*," "*travel*," "*drop*," "*screw*," etc., to parametric skill definitions stored in *sequence_library* or *operation_sequence*. Snippet below shows the insertion logic translating each subtask into database records so the simulator can read and simulate them.

- Integration with Simulation: Commands are forwarded to NVIDIA Omniverse (or a local Isaac Sim instance) via Python APIs or ROS bridges, simulating the full action path. If collisions or kinematic constraints arise, the system re-triggers partial re-planning or user clarification.



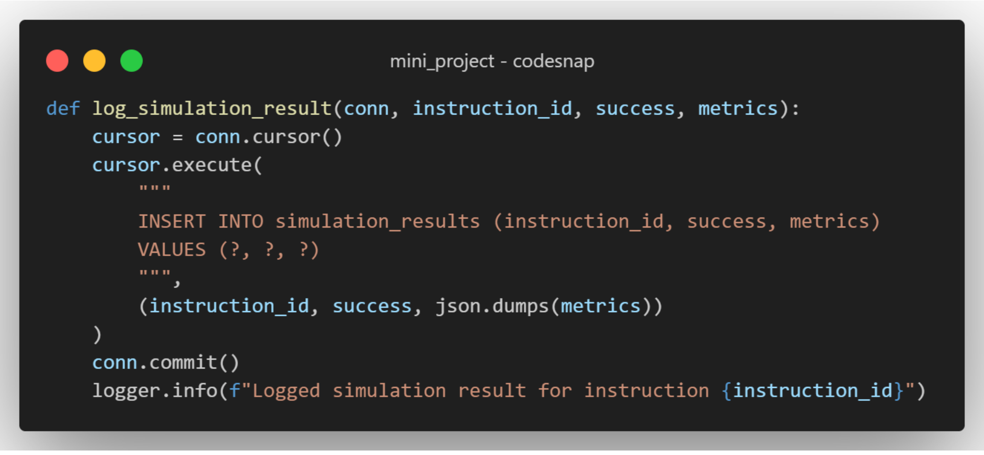
```
for op in operations:
    seq_name = op["sequence_name"]
    obj_name = op["object_name"]
    cursor.execute(
        """
        INSERT INTO instruction_operation_sequence (
            instruction_id, sequence_id, sequence_name, object_id, object_name
        ) VALUES (?, ?, ?, ?, ?)
        """,
        (unified_cmd_id, get_sequence_id(seq_name), seq_name, get_object_id(obj_name), obj_name),
    )
conn.commit()
simulate_in_omniverse(unified_cmd_id)
```

Figure 3-15. Instruction_operation_sequence insertion logic

Step 6: Simulation Feedback and Data Logging

The validated plan is dispatched to NVIDIA Omniverse. For instance, “pick RedCube” is executed by retrieving the cube’s location from camera_vision and instructing the robot to approach and grasp.

- The simulation checks feasibility: collision detection, approach angles, tool distances, etc. If successful, the final path is considered validated.
- The orchestrator logs success/failure in the DB (e.g., *unified_instructions* updated with processed=1, or new entry in simulation_results with metrics).
- User Feedback: The system displays or verbalizes “Task validated in simulation. Red cube moved to tray.” If a problem emerges, the user can revise the command, or the orchestrator can automatically re-plan. This snippet below show how the system tracks performance for each user command.



```
def log_simulation_result(conn, instruction_id, success, metrics):
    cursor = conn.cursor()
    cursor.execute(
        """
        INSERT INTO simulation_results (instruction_id, success, metrics)
        VALUES (?, ?, ?)
        """,
        (instruction_id, success, json.dumps(metrics))
    )
    conn.commit()
    logger.info(f"Logged simulation result for instruction {instruction_id}")
```

Figure 3-16. Illustrates final logging of success/failure or time metrics in simulation_results.

3.4.3. Notable Implementation Features

One of the core features of the system is its database-driven personalization. User preferences and historical interactions, such as specific references made by users to objects (e.g., "the red cube is 'the object I used last time'"), are stored in tables like users or unified_instructions. This enables the system to accurately interpret vague references, improving the precision of the language model's (LLM) responses. Additionally, updates from face or voice recognition, along with multiple encodings per user, are continuously fed into voice_instructions, gesture_instructions, and interaction_memory. This continuous stream of data allows the system to personalize interactions and enables historical analytics, thereby refining the user experience over time.

The system also incorporates centralized configuration in the form of *app_config.py*, which stores key constants like *FACE_MATCH_THRESHOLD*, *LLM_MODEL*, and *DB_PATH*. This approach ensures that any updates to parameters or changes to the environment are handled consistently, preventing disruptions to the core system logic. Centralizing configuration in this manner simplifies maintenance and adaptation of the system to different operational contexts.

Error handling and re-planning are also integral to the system's functionality. If a recognized object is no longer present in the scene—whether due to removal or being out of view—the system detects this by querying the camera_vision module. In response, an error is flagged, and the system prompts the user for clarification. Moreover, if the LLM's task plan references an invalid object or task, the orchestrator reverts to synchronizer.py to adjust the plan or request additional user input. This ensures that the system remains flexible and adaptive to changes in the environment, minimizing disruptions.

A user-centric approach lies at the heart of the system's design. By combining face recognition, voice embeddings, and a history of past interactions, the system tailors each session to the user's preferences and needs. This personalization improves the accuracy of interactions, reduces the need for repeated clarifications, and enhances overall user satisfaction by ensuring that the system "remembers" users' preferences and adapts accordingly.

Lastly, robust feedback loops are implemented to support continuous system improvement. The system logs both successes and failures, allowing for quick identification of misinterpretations or recurring issues. These logs provide valuable data that can be used to adjust the system in real-time, enabling iterative refinements that enhance the system's accuracy and responsiveness over time.

Together, these features form the foundation of a highly adaptable, user-focused system that can continuously improve and provide personalized, reliable interactions.

4. Results & Findings

In this chapter, the results of the integrated system are presented and discussed. The performance of each subsystem has been tested, with a focus on how well they work as part of the larger system. This chapter outlines the system's implementation and integration outcomes, technical capabilities, performance evaluation, and on key findings from the experiments conducted.

4.1. System Implementation and Integration

Overview

The system was successfully developed as a modular and extensible pipeline, integrating computer vision, a relational database, large language models (LLMs), user authentication, and robot control within both virtual (Omniverse Isaac Sim) and physical environments, through a web-based interface.

The architecture developed in this thesis is composed of the following key components:

Computer vision: Responsible for perceiving and interpreting the environment, including detecting objects, identifying user faces, recognizing gesture cues, and providing real-time scene data to the robotic system.

Database: This serves as a central repository for storing various system data, including object properties, task definitions, user profiles, interaction histories etc., which are then queried for task execution and personalization purposes.

LLM & Modalities: The use of large language models (LLMs) enables the system to interpret natural language and gesture commands, plan tasks based on the user's input and context, providing intelligent, context-aware interactions.

Authentication: This module ensures that user identity is verified through facial and/or voice recognition, providing a secure, role-based personalized access, that is based on the user's task preferences or previous commands.

Web Interface: The user interacts with the system via a web interface, where commands are input, and feedback is received, bridging the communication between the user and the robot. The GUI ensures that users can easily engage with the system, issue commands, monitor the database, and track task plan and progression.

Another core module worth mentioning is the *Robot Control* module. Though developed not as part of the thesis, but alongside it for the broader project goal. The module interfaces with both the digital twin in Omniverse Isaac Sim and the physical ABB Yumi cobot, retrieving task plans from the database and executing them in real or simulated environments.

Each of these components was developed independently and then integrated into a unified system. The following sections describe how these modules were brought together to form a fully functional pipeline.

Integration Outcomes

The integration of the individual modules into the system resulted in a seamless interaction between the robot, its environment, and the user. The workflow begins with user authentication, where upon initiation, the system ensures the user's identity through face and/or voice recognition. This initial step guarantees that the system recognizes the user and can provide personalized interactions based on their stored profile in the database.

Following authentication, the computer vision module, via a camera continuously captures real-time footage of the robot's workspace. The camera feeds are processed to detect and classify objects in the scene. Key object data, such as positions, types, and colours, are extracted and stored in the database, ensuring the system maintains an up-to-date representation of the scene. This allows for an accurate and dynamic interaction with the physical environment.

When the user issues a command or question, using their natural language, in voice and maybe gestures, the LLM interprets the request via well-defined system prompts, cross-referencing them with the stored user's profile, roles and previous interactions. In the case of a task/action request, the LLM processes the command, structures the required task, queries the database for relevant task parameters and object properties to generate a task plan, and intelligently formulates a response. Moreso, general non-action requests/questions, are handled by fine-tuned system prompts which ensures that interaction flows naturally, maintaining context.

Once the task plan is defined, the robot control module, initiates the task execution control script, queries the database for the task plan and object data, and proceeds to execute it in the virtual environment (Omniverse Isaac Sim), whilst the physical robot mirrors the actions in real time. The user request is accomplished using the real-time data retrieved from the database tables, either within a virtual simulation or in the real world, depending on the setup.

During task execution, the robot's actions can be continuously monitored though the database tables, with feedback provided to the user via the web interface if requested. This allows the user to track the status and progress of the task in real-time. The integration of a database ensures efficient data retrieval and management throughout the operation.

The system adapts based on the user's feedback (e.g. task confirmations) and any changes in the scene (e.g. object positions, orientations), offering further interaction opportunities. As tasks are completed, and new commands are issued, the system dynamically adjusts its responses to queries, utilizing interaction data logged in the database, ensuring that interactions remain contextually relevant.

The modular integration facilitated parallel development and testing of individual components, allowed for flexibility in adding new user data and features, through well-defined database schemas. This will ensure future scalability to meet new task requirements.

Key integration outcomes include:

1. The use of a centralized PostgreSQL database enabled efficient sharing of system data between modules, supporting real-time updates and concurrent queries.
2. The schema supports adding new task types, user roles, gesture definitions, object USD data, robot control features etc. without major codebase refactoring.
3. The system remained stable during extended testing, with error handling mechanisms at module boundaries to manage communication failures and data access issues.

Despite these successes, several integration challenges had to be addressed:

1. Coordinating task execution across simulation and real-world environments presented major challenges. Ensuring real-time synchronization between the virtual and physical robots required careful management of communication latency and error correction.
2. Rapid scene change, especially during object manipulation or occlusion events, sometimes caused mismatches between the vision output and the database state. This was addressed by implementing transaction management and state validation.
3. Integrating LLMs into the live interaction loop introduced challenges related to processing delay and command ambiguity. Prompt tuning, caching common responses and introducing fallback logic for unclear queries helped improve reliability.
4. The most significant challenge was hardware related. While the control logic was initially developed for a standard single-arm robot, the actual test robot was equipped with two arms, each with an extra joint. This mismatch limited reach and required adjustments to motion planning and execution strategies.

Despite the challenges, the integration phase validated the system's modularity and adaptability, providing a solid foundation for a scalable, personalized, and robust task execution.

Performance and Capabilities

Models:

NAME	ID	SIZE
llama3.3:latest	a6eb4748fd29	42 GB
deepseek-r1:32b	38056bbcbb2d	19 GB
phi4:latest	ac896e5b8b34	9.1 GB
mistral:latest	f974a74358d6	4.1 GB
gemma3:4b	a2af6cc3eb7f	3.3 GB
llama3.2:latest	a80c4f17acd5	2.0 GB
deepseek-r1:1.5b	a42b25d8c10a	1.1 GB
qwen2.5:0.5b	a8b0c5157701	397 MB

Voice processor

4.2. Findings

{Place Holder}

5. Discussion & Conclusion

5.1. Discussion

{Place Holder}

5.2. Conclusion & Future work

{Place Holder}

References

[1] 'CES - The Most Powerful Tech Event in the World'. Accessed: Mar. 05, 2025. [Online]. Available: https://www.ces.tech/

[2] 'Enchanted Tools / Changing the face of robotics', Enchanted Tools. Accessed: Mar. 10, 2025. [Online]. Available: https://enchanted.tools/

[3] C. Young, 'New-look Mirokai robot assists hospital patients, serves as concierge', Interesting Engineering. Accessed: Mar. 10, 2025. [Online]. Available: https://interestingengineering.com/ces-2025/new-look-mirokai-robot-assists-patients

[4] M. Shridhar, L. Manuelli, and D. Fox, 'CLIPort: What and Where Pathways for Robotic Manipulation', in *Proceedings of the 5th Conference on Robot Learning*, PMLR, Jan. 2022, pp. 894–906. Accessed: Mar. 05, 2025. [Online]. Available: https://proceedings.mlr.press/v164/shridhar22a.html

[5] T. R. R. Staff, 'IFR World Robotics report says 4M robots are operating in factories globally', The Robot Report. Accessed: Mar. 10, 2025. [Online]. Available: https://www.therobotreport.com/ifr-4-million-robots-operating-globally-world-robotics-report/

[6] S. Riso, D. Adascalitei, and European Foundation for the Improvement of Living and Working Conditions, Eds., *Human-robot interaction: what changes in the workplace?* Luxembourg: Publications Office, 2024. doi: 10.2806/67956.

[7] 'Human-Centered Robots Are Key to Creating a Society We All Want to Live In', NTT DATA. Accessed: Mar. 12, 2025. [Online]. Available: https://www.nttdata.com/global/en/insights/technology/academic-collaborations/mit-media-lab-cynthia-breazeal

[8] C. Breazeal, K. Dautenhahn, and T. Kanda, 'Social Robotics', in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., Cham: Springer International Publishing, 2016, pp. 1935–1972. doi: 10.1007/978-3-319-32552-1_72.

[9] 'Cynthia Breazeal', *Wikipedia*. Mar. 09, 2025. Accessed: Mar. 10, 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Cynthia_Breazeal&oldid=1279548619

[10] 'Digitalsheet_A4_World_Robotics_2023_low.pdf'. Accessed: Mar. 05, 2025. [Online]. Available: https://ifr.org/img/worldrobotics/Digitalsheet_A4_World_Robotics_2023_low.pdf

[11] 'World Economic Forum Annual Meeting 2025', World Economic Forum. Accessed: Mar. 05, 2025. [Online]. Available: https://www.weforum.org/meetings/world-economic-forum-annual-meeting-2025/

[12] G. Jocher, J. Qiu, and A. Chaurasia, *Ultralytics YOLO*. (Jan. 2023). Python. Accessed: Mar. 05, 2025. [Online]. Available: https://github.com/ultralytics/ultralytics

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding', May 24, 2019, *arXiv*: arXiv:1810.04805. doi: 10.48550/arXiv.1810.04805.

[14] 'Introducing GPT-4.5'. Accessed: Mar. 05, 2025. [Online]. Available: https://openai.com/index/introducing-gpt-4-5/

[15] S. Tellex, N. Gopalan, H. Kress-Gazit, and C. Matuszek, 'Robots that use language', *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 25–55, 2020.

[16] M. Quigley *et al.*, 'ROS: an open-source Robot Operating System'.

[17] 'NVIDIA Omniverse', NVIDIA Docs. Accessed: Mar. 05, 2025. [Online]. Available: https://docs.nvidia.com/omniverse/index.html

[18] S. Levine, C. Finn, T. Darrell, and P. Abbeel, 'End-to-End Training of Deep Visuomotor Policies', Apr. 19, 2016, *arXiv*: arXiv:1504.00702. doi: 10.48550/arXiv.1504.00702.

[19] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, 'Domain randomization for transferring deep neural networks from simulation to the real world', in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 23–30. doi: 10.1109/IROS.2017.8202133.

[20] 'GTC 2023 Keynote | GTC Digital Spring 2023 | NVIDIA On-Demand', NVIDIA. Accessed: Mar. 05, 2025. [Online]. Available: https://www.nvidia.com/en-us/on-demand/session/gtcspring23-s52226/

[21] 'NVIDIA GTC 2023 to Feature Latest Advances in AI Computing Systems, Generative AI, Industrial Metaverse, Robotics; Keynote by Jensen Huang; Talks by OpenAI, DeepMind Founders', NVIDIA Newsroom. Accessed: Mar. 05, 2025. [Online]. Available: https://nvidianews.nvidia.com/news/nvidia-gtc-2023-to-feature-latest-advances-in-ai-computing-systems-generative-ai-industrial-metaverse-robotics-keynote-by-jensen-huang-talks-by-openai-deepmind-founders

[22] 'Privacy in the age of robotics | IAPP'. Accessed: Mar. 11, 2025. [Online]. Available: https://iapp.org/news/a/privacy-in-the-age-of-robotics

[23] Y. Zhou and H. Li, 'A Scientometric Review of Soft Robotics: Intellectual Structures and Emerging Trends Analysis (2010–2021)', *Front. Robot. AI*, vol. 9, May 2022, doi: 10.3389/frobt.2022.868682.

[24] G. C. Devol, 'Programmed Article Transfer', US 2,988,237, 1961

[25] B. Siciliano and O. Khatib, Eds., *Springer Handbook of Robotics*. in Springer Handbooks. Cham: Springer International Publishing, 2016. doi: 10.1007/978-3-319-32552-1.

[26] M. A. Goodrich and A. C. Schultz, 'Human–Robot Interaction: A Survey', *HCI*, vol. 1, no. 3, pp. 203–275, Jan. 2008, doi: 10.1561/1100000005.

[27] 'WEF_Frontier_Technologies_in_Industrial_Operations_2025.pdf'. Accessed: Mar. 05, 2025. [Online]. Available: https://reports.weforum.org/docs/WEF_Frontier_Technologies_in_Industrial_Operations_2025.pdf

[28] K. Dautenhahn, 'Socially intelligent robots: dimensions of human–robot interaction', *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 362, no. 1480, pp. 679–704, Feb. 2007, doi: 10.1098/rstb.2006.2004.

[29] M. Stolarz, A. Mitrevski, M. Wasil, and P. G. Plöger, 'Learning-based personalisation of robot behaviour for robot-assisted therapy', *Front. Robot. AI*, vol. 11, Apr. 2024, doi: 10.3389/frobt.2024.1352152.

[30] G. Jocher and J. Qiu, *Ultralytics YOLO11*. (2024). [Online]. Available: https://github.com/ultralytics/ultralytics

[31] J. Redmon and A. Farhadi, 'YOLOv3: An Incremental Improvement', Apr. 08, 2018, *arXiv*: arXiv:1804.02767. doi: 10.48550/arXiv.1804.02767.

[32] Ultralytics, 'RT-DETR vs YOLOv6-3.0: Uma comparação pormenorizada de modelos'. Accessed: Feb. 22, 2025. [Online]. Available: https://docs.ultralytics.com/pt/compare/rtdetr-vs-yolov6

[33] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, 'End-to-End Object Detection with Transformers', May 28, 2020, *arXiv*: arXiv:2005.12872. doi: 10.48550/arXiv.2005.12872.

[34] 'The science behind visual ID', Amazon Science. Accessed: Feb. 22, 2025. [Online]. Available: https://www.amazon.science/blog/the-science-behind-visual-id

[35] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, 'You Only Look Once: Unified, Real-Time Object Detection', May 09, 2016, *arXiv*: arXiv:1506.02640. doi: 10.48550/arXiv.1506.02640.

[36] R. Xu, F.-J. Chu, C. Tang, W. Liu, and P. A. Vela, 'An Affordance Keypoint Detection Network for Robot Manipulation', *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2870–2877, Apr. 2021, doi: 10.1109/LRA.2021.3062560.

[37] N. Yamanobe *et al.*, 'A brief review of affordance in robotic manipulation research', *Advanced Robotics*, vol. 31, no. 19–20, pp. 1086–1101, Oct. 2017, doi: 10.1080/01691864.2017.1394912.

[38] F.-J. Chu, R. Xu, L. Seguin, and P. A. Vela, 'Toward Affordance Detection and Ranking on Novel Objects for Real-World Robotic Manipulation', *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 4070–4077, Oct. 2019, doi: 10.1109/LRA.2019.2930364.

[39] F. Schroff, D. Kalenichenko, and J. Philbin, 'FaceNet: A Unified Embedding for Face Recognition and Clustering', in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 815–823. doi: 10.1109/CVPR.2015.7298682.

[40] 'Sensory's TrulySecure Face & Voice Biometrics Software', Sensory. Accessed: Feb. 22, 2025. [Online]. Available: https://www.sensory.com/face-voice-biometrics/

[41] F. Zeng, W. Gan, Y. Wang, N. Liu, and P. S. Yu, 'Large Language Models for Robotics: A Survey', Nov. 13, 2023, *arXiv*: arXiv:2311.07226. doi: 10.48550/arXiv.2311.07226.

[42] J. Wang *et al.*, 'Large Language Models for Robotics: Opportunities, Challenges, and Perspectives', Jan. 09, 2024, *arXiv*: arXiv:2401.04334. doi: 10.48550/arXiv.2401.04334.

[43] D. Driess *et al.*, 'PaLM-E: An Embodied Multimodal Language Model', Mar. 06, 2023, *arXiv*: arXiv:2303.03378. doi: 10.48550/arXiv.2303.03378.

[44] Google DeepMind, 'Demonstrating Large Language Models on Robots', in *Robotics: Science and Systems XIX*, Robotics: Science and Systems Foundation, Jul. 2023. doi: 10.15607/RSS.2023.XIX.024.

[45] Z. Hu *et al.*, 'Deploying and Evaluating LLMs to Program Service Mobile Robots', *IEEE Robot. Autom. Lett.*, vol. 9, no. 3, pp. 2853–2860, Mar. 2024, doi: 10.1109/LRA.2024.3360020.

[46] H. Choi *et al.*, 'On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward', *Proceedings of the National Academy of Sciences*, vol. 118, no. 1, p. e1907856118, Jan. 2021, doi: 10.1073/pnas.1907856118.

[47] C. K. Liu and D. Negrut, 'The Role of Physics-Based Simulators in Robotics', *Annu. Rev. Control Robot. Auton. Syst.*, vol. 4, no. 1, pp. 35–58, May 2021, doi: 10.1146/annurev-control-072220-093055.

[48] L. Žlajpah, 'Simulation in robotics', *Mathematics and Computers in Simulation*, vol. 79, no. 4, pp. 879–897, Dec. 2008, doi: 10.1016/j.matcom.2008.02.017.

[49] A. Afzal, D. S. Katz, C. L. Goues, and C. S. Timperley, 'A Study on the Challenges of Using Robotics Simulators for Testing', Apr. 15, 2020, *arXiv*: arXiv:2004.07368. doi: 10.48550/arXiv.2004.07368.

[50] R. E. Fikes and N. J. Nilsson, 'Strips: A new approach to the application of theorem proving to problem solving', *Artificial Intelligence*, vol. 2, no. 3–4, pp. 189–208, Dec. 1971, doi: 10.1016/0004-3702(71)90010-5.

[51] J. Tang, Z. Ye, Y. Yan, Z. Zheng, T. Gao, and Y. Jin, 'Zero-shot Robotic Manipulation with Language-guided Instruction and Formal Task Planning', Jan. 25, 2025, *arXiv*: arXiv:2501.15214. doi: 10.48550/arXiv.2501.15214.

[52] Z. Wu, Z. Wang, X. Xu, J. Lu, and H. Yan, 'Embodied Task Planning with Large Language Models', Jul. 04, 2023, *arXiv*: arXiv:2307.01848. doi: 10.48550/arXiv.2307.01848.

[53] H. Liu, Y. Zhu, K. Kato, I. Kondo, T. Aoyama, and Y. Hasegawa, *LLM-Based Human-Robot Collaboration Framework for Manipulation Tasks*. 2023. doi: 10.48550/arXiv.2308.14972.

[54] L. Yang, S. Luo, X. Cheng, and L. Yu, 'Leveraging Large Language Models for Enhanced Digital Twin Modeling: Trends, Methods, and Challenges', Mar. 04, 2025, *arXiv*: arXiv:2503.02167. doi: 10.48550/arXiv.2503.02167.

[55] V. Kumar, S. Ranjani, M. Srinivas, and S. Bhat, 'Improving LLM-based Robot Control via Human-Robot Cooperation', *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 12, pp. 1–12, Aug. 2024, doi: 10.15680/IJIRCCE.2024.1208016.

[56] T. Nguyen, N. Gopalan, R. Patel, M. Corsaro, E. Pavlick, and S. Tellex, 'Robot Object Retrieval with Contextual Natural Language Queries', Jul. 2020. doi: 10.15607/RSS.2020.XVI.080.

[57] D. Martinez-Baselga, O. de Groot, L. Knoedler, J. Alonso-Mora, L. Riazuelo, and L. Montano, *Hey Robot! Personalizing Robot Navigation through Model Predictive Control with a Large Language Model*. 2024. doi: 10.48550/arXiv.2409.13393.

[58] H. Pan *et al.*, *Recent Advances in Robot Navigation via Large Language Models: A Review*. 2024. doi: 10.13140/RG.2.2.35284.41603.

[59] G. Lu, Z. Yan, J. Luo, and W. Li, 'Integrating Historical Learning and Multi-View Attention with Hierarchical Feature Fusion for Robotic Manipulation', *Biomimetics*, vol. 9, p. 712, Nov. 2024, doi: 10.3390/biomimetics9110712.

[60] H. Liu *et al.*, 'Enhancing the LLM-Based Robot Manipulation Through Human-Robot Collaboration', *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–8, Aug. 2024, doi: 10.1109/LRA.2024.3415931.

[61] W. Wang, X. Li, Y. Dong, J. Xie, and D. Guo, 'Natural Language Instruction Understanding for Robotic Manipulation: a Multisensory Perception Approach', May 2023, pp. 9800–9806. doi: 10.1109/ICRA48891.2023.10160906.

[62] I. Singh *et al.*, 'ProgPrompt: Generating Situated Robot Task Plans using Large Language Models', May 2023, pp. 11523–11530. doi: 10.1109/ICRA48891.2023.10161317.

[63] J. Wu *et al.*, 'TidyBot: personalized robot assistance with large language models', *Autonomous Robots*, vol. 47, pp. 1–16, Nov. 2023, doi: 10.1007/s10514-023-10139-z.

[64] H. Liu *et al.*, *Enhancing the LLM-Based Robot Manipulation Through Human-Robot Collaboration*. 2024. doi: 10.48550/arXiv.2406.14097.

Other very relevant works to cite (IGNORE THIS)

[53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64]

Ahn, M., Brohan, A., Brown, N., et al. (2022). Do as I can, not as I say: Grounding language in robotic affordances. arXiv preprint, [Online]. Available: https://doi.org/10.48550/arXiv.2204.01691

Brown, T., Mann, B., Ryder, N., et al. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems (NeurIPS), 33, 1877–1901. [Online]. Available: https://doi.org/10.48550/arXiv.2005.14165

Chaplot, D. S., Salakhutdinov, R., & Gupta, A. (2021). Neural Topological SLAM for Visual Navigation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

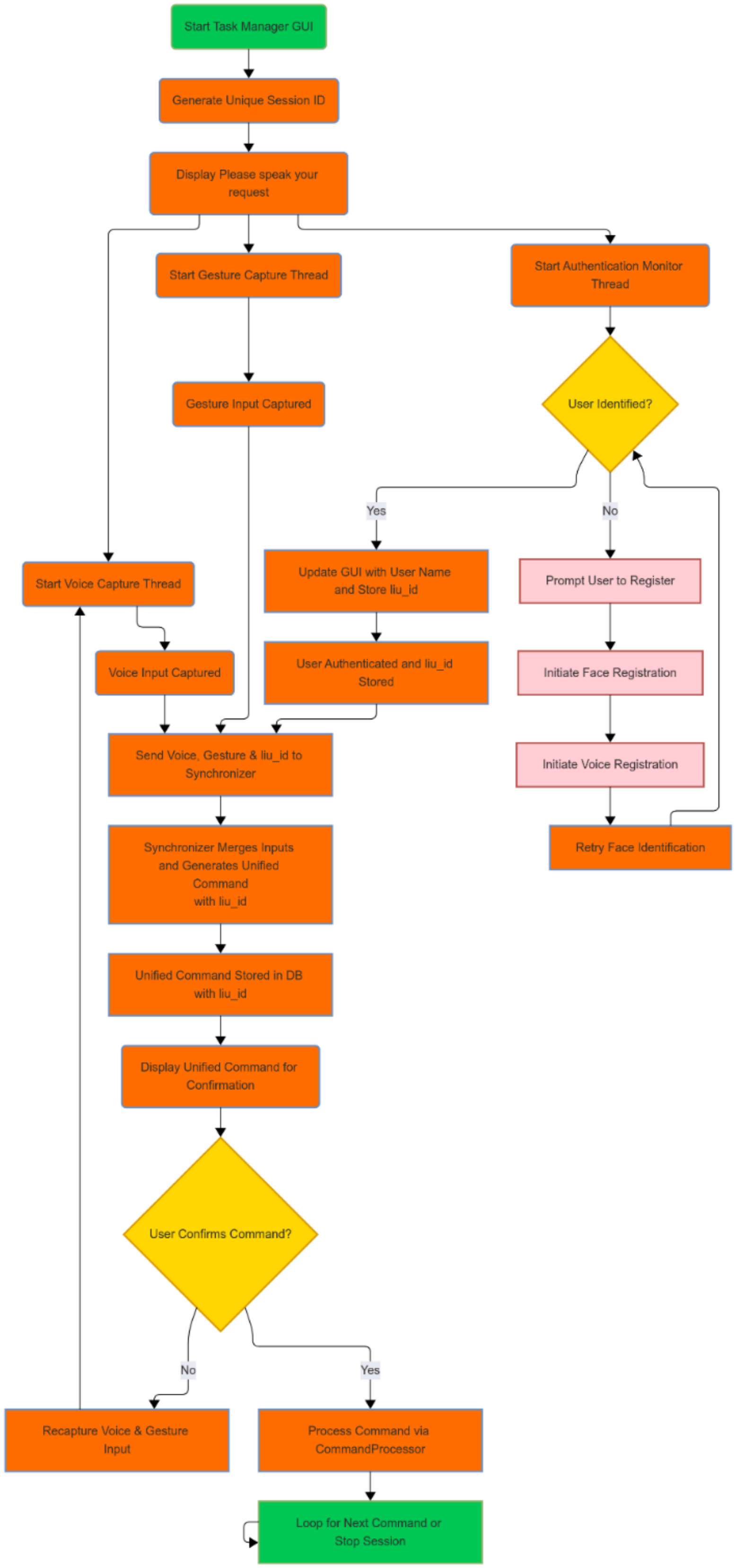
iREX Tokyo. (2025). Panel on Ethical Robotics in Healthcare [Conference session]. International Robot Exhibition.

Liu, W., Paxton, C., Fox, D., & Gupta, A. (2023). Bridging Vision-Language Models with Embodied Robotics for Interactive Task Execution. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Radford, A., Kim, J. W., Hallacy, C., et al. (2021). Learning Transferable Visual Models from Natural Language Supervision. Proceedings of the 38th International Conference on Machine Learning (ICML). [Online]. Available: https://doi.org/10.48550/arXiv.2103.00020

Appendix

A - Operational workflow – Approach II


Start Task Manager GUI
Generate Unique Session ID
Display Please speak your request
Start Gesture Capture Thread
Start Authentication Monitor Thread
Gesture Input Captured
User Identified?
Yes
No
Update GUI with User Name and Store liu_id
Prompt User to Register
Start Voice Capture Thread
Voice Input Captured
User Authenticated and liu_id Stored
Initiate Face Registration
Send Voice, Gesture & liu_id to Synchronizer
Initiate Voice Registration
Synchronizer Merges Inputs and Generates Unified Command with liu_id
Retry Face Identification
Unified Command Stored in DB with liu_id
Display Unified Command for Confirmation
User Confirms Command?
No
Yes
Recapture Voice & Gesture Input
Process Command via CommandProcessor
Loop for Next Command or Stop Session

B - Research Organisation (IGNORE THIS)

Abstract

~~A summary of the thesis: problem statement, objectives, methodology, key results, and contributions (1+50–300 words)~~

~~Single paragraph, no citations.~~

Introduction

Literature Review
Review related works on:
Vision-based perception., Language models for robotics., Learning-based task planning, Omniverse Isaac Sim.
| *Critical analysis of vision-language robotics, personalization, Omniverse Isaac Sim.*
Identify research gaps.
| *Highlight gaps this work addresses.*

METHODOLOGY:
System Architecture
| *System architecture, tools (e.g., PyTorch, ROS, Omniverse), dataset/pipeline development.*
| *Include pseudocode/diagrams.*
Algorithms & Tools | *(LLMs, ROS, Omniverse Isaac Sim)*
Datasets | *Data collection methods.*
Implementation Details
| *Detailed workflow: Vision-Language integration, task planning, simulation pipeline.*
| *Subsections: Vision Perception, LLM Adaptation, Robotic Control, Omniverse Setup*.
Setup for robotic framework.
| *Simulation setup in Omniverse Isaac Sim.*
Testing methodologies.

Results & Findings

Results

Presentation of results. |
Figures, tables, graphs. | *Tables/figures with statistical validation.*

Findings

Performance evaluation. | *Simulation outcomes, task success rates, user study metrics (personalization efficacy).*

Discussion & Conclusion

Discussion

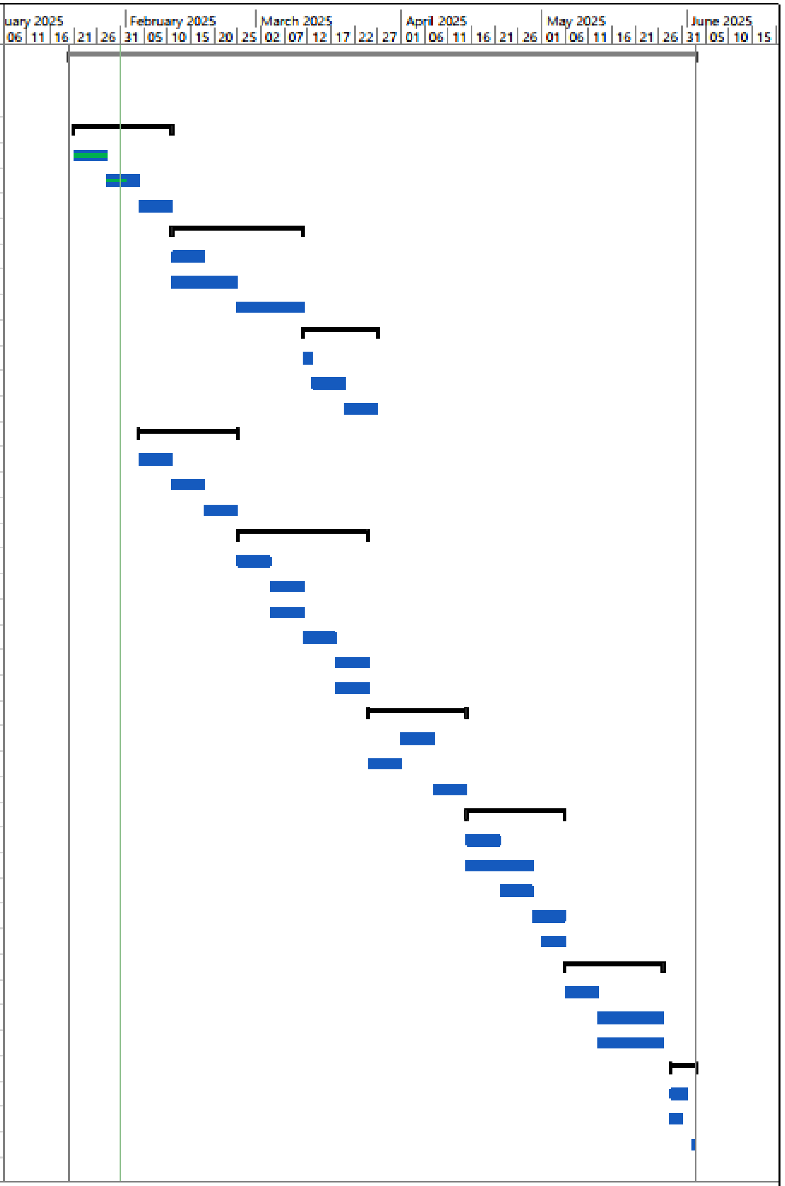
Interpretation of results. | *Interpret results, compare with literature, address limitations.*
Comparison with related work. | *Link to research questions.*
Challenges and limitations.

Conclusion & Future work

Summary of contributions. | *Recap contributions, and propose extensions (e.g., real-world deployment).*
Key takeaways. | *Avoid introducing new data.*
Future research directions.

Project plan

ID	ℹ	Task Mode	WBS	Task Name	Duration	Start	Finish	Deliverable
0			0	**Integrating Vision, Language Models, and Robotic Control for Personalized Task Execution in Virtual Environments.**	**96 days**	**20 Jan**	**02 Jun**	
1			1	**Project Planning & Setup**	**15 days**	**21 Jan**	**10 Feb**	**Approved planning report**
2	✓		1.1	Define scope & objectives	1 wk	21 Jan	27 Jan	Scope document
3	📅		1.2	Set up tools	1 wk	28 Jan	03 Feb	Tools ready
4			1.3	Report: Introduction	1 wk	04 Feb	10 Feb	Chapter 1 of report completed
5			2	**Literature Review & Background Study**	**20 days**	**11 Feb**	**10 Mar**	**Literature review notes, reference list**
6	📅		2.1	Study SOTA in Vision, LLM, Robot Control	1 wk	11 Feb	17 Feb	Summaries of relevant work
7	📅		2.2	Identify Gaps & re-formulate RQs	2 wks	11 Feb	24 Feb	Updated research questions
8	📅		2.3	Report: Literature review	2 wks	25 Feb	10 Mar	Chapter 2 of report completed
9			3	**Requirements & System Architecture**	**12 days**	**11 Mar**	**26 Mar**	**Architecture diagram**
10			3.1	Draft requirements & system architecture	2 days	11 Mar	12 Mar	Requirements checklist
11			3.2	Plan implementation strategy	1 wk	13 Mar	19 Mar	Process flow charts
12			3.3	Report: System design & implementation	1 wk	20 Mar	26 Mar	Chapter 3 of report completed
13			4	**Vision-Based Perception**	**15 days**	**04 Feb**	**24 Feb**	**Functional vision pipeline**
14	📅		4.1	Develop vision pipeline	1 wk	04 Feb	10 Feb	Preliminary vision code
15	📅		4.2	Integrate face/voice recognition	1 wk	11 Feb	17 Feb	User Identification module
16	📅		4.3	Link user IDs to profiles	1 wk	18 Feb	24 Feb	User profiles linked
17			5	**Language Understanding**	**20 days**	**25 Feb**	**24 Mar**	**LLM integrated with CV**
18	📅		5.1	Set-up & Integrate LLM	1 wk	25 Feb	03 Mar	LLM operational
19			5.2	Integrate audio & vision inputs with LLM	1 wk	04 Mar	10 Mar	Working pipeline to parse commands
20			5.3	Define Prompts & conversation logic	1 wk	04 Mar	10 Mar	LLM prompt structure
21	📅		5.4	Develop task representation	1 wk	11 Mar	17 Mar	Task representation ready
22			5.5	Test LLM Parsing, Basic Dialogue	1 wk	18 Mar	24 Mar	LLM test report
23	📅		5.6	Build conversational memory	1 wk	18 Mar	24 Mar	Conversational memory built
24			6	**Robotic Control**	**15 days**	**25 Mar**	**14 Apr**	**Decision engine & skill library**
25	📅		6.1	Define robot skill library	1 wk	01 Apr	07 Apr	Skill library ready
26	📅		6.2	Map decision engine (LLM to skills)	1 wk	25 Mar	31 Mar	Decision-making code
27	📅		6.3	Generate GUI control scripts	1 wk	08 Apr	14 Apr	Graphic User Interface
28			7	**Simulation in Omniverse**	**15 days**	**15 Apr**	**05 May**	**Omniverse simulations**
29	📅		7.1	Set up virtual robot & environment	1 wk	15 Apr	21 Apr	Omniverse environment ready
30			7.2	Integrate Vision, LLM, Control Modules	2 wks	15 Apr	28 Apr	Working integrated pipeline
31	📅		7.3	Simulate task execution	1 wk	22 Apr	28 Apr	Task simulation logs
32	📅		7.4	Debug & refine	1 wk	29 Apr	05 May	Validation report
33			7.5	Report: Methodology	3 days	01 May	05 May	Chapter 3 of report
34			8	**Testing & Validation**	**15 days**	**06 May**	**26 May**	**Test results**
35	📅		8.1	Test, analyse logs, & identify issues	1 wk	06 May	12 May	Robot testing complete
36	📅		8.2	Validate performance	2 wks	13 May	26 May	Performance validation report
37			8.3	Report: Results	2 wks	13 May	26 May	Chapter 4 of report complete
38			9	**Documentation & Reporting**	**3.5 days**	**28 May**	**02 Jun**	**Thesis draft, presentations**
39	📅		9.1	Prepare presentations/posters	0.5 wks	28 May	31 May	Presentations, posters, videos
40			9.2	Report: Discussion & Conclusion	0.5 wks	28 May	30 May	Chapter 5 & 6 of report complete
41	📅		9.3	Final submission	1 day	02 Jun	02 Jun	Final report, reflection document
42								

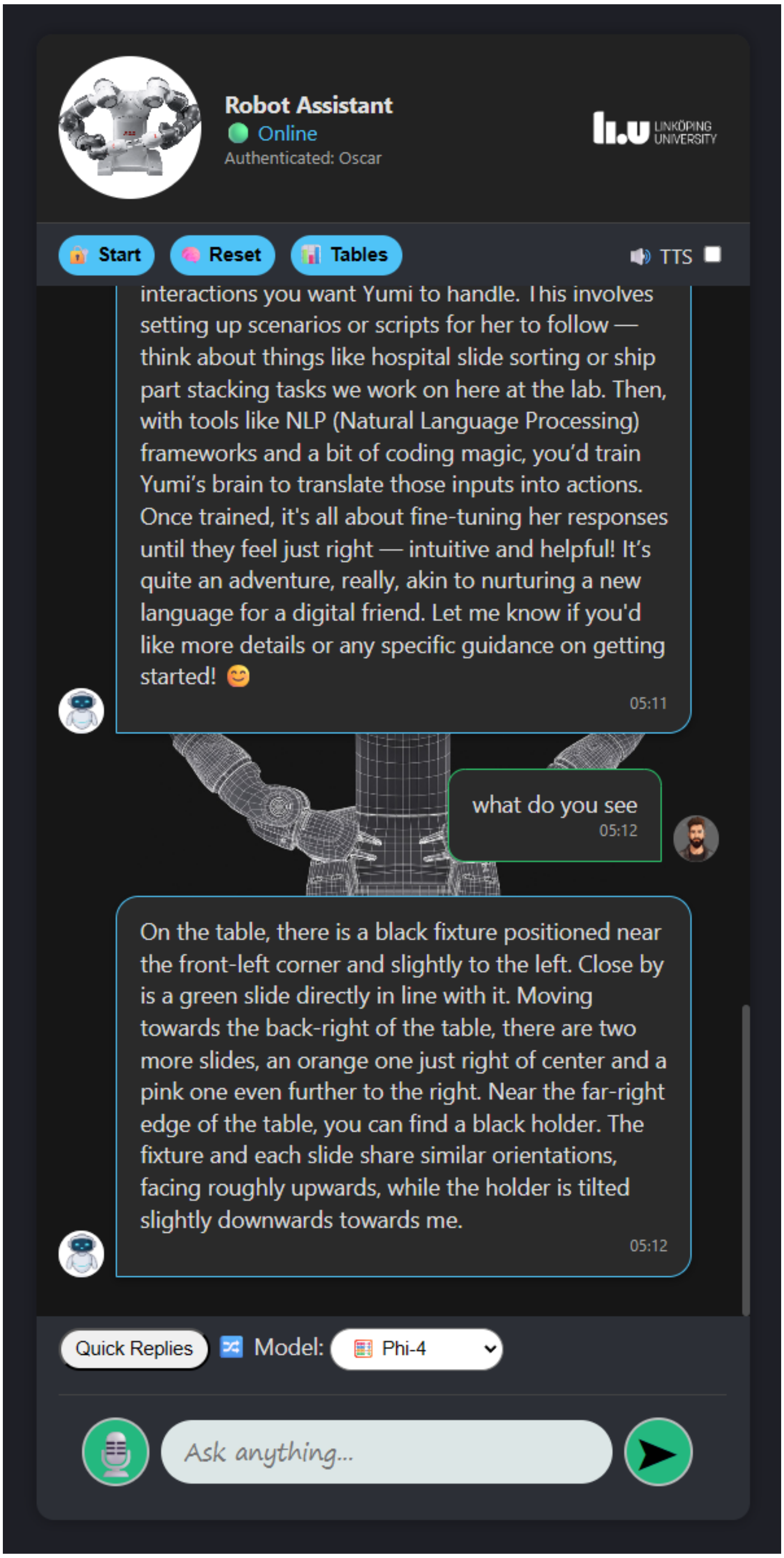


Project: Integrating Vision, Language Models, and Robotic Control for Personalized Task Execution in Virtual Environments.

Task	▬	Project Summary	┗━┛	Manual Task	▬	Start-only	[	Deadline	⬇
Split	┈┈┈	Inactive Task	▭	Duration-only	▬	Finish-only	]	Progress	━
Milestone	◆	Inactive Milestone	◇	Manual Summary Rollup	━	External Tasks	▬	Manual Progress	━
Summary	┗━┛	Inactive Summary	┗━┛	Manual Summary	┗━┛	External Milestone	◇		

Master's Thesis Project Plan - VT2025

Date: 31 Jan



interactions you want Yumi to handle. This involves setting up scenarios or scripts for her to follow — think about things like hospital slide sorting or ship part stacking tasks we work on here at the lab. Then, with tools like NLP (Natural Language Processing) frameworks and a bit of coding magic, you'd train Yumi's brain to translate those inputs into actions. Once trained, it's all about fine-tuning her responses until they feel just right — intuitive and helpful! It's quite an adventure, really, akin to nurturing a new language for a digital friend. Let me know if you'd like more details or any specific guidance on getting started! 😊
05:11

what do you see
05:12

On the table, there is a black fixture positioned near the front-left corner and slightly to the left. Close by is a green slide directly in line with it. Moving towards the back-right of the table, there are two more slides, an orange one just right of center and a pink one even further to the right. Near the far-right edge of the table, you can find a black holder. The fixture and each slide share similar orientations, facing roughly upwards, while the holder is tilted slightly downwards towards me.
05:12