ក្រសួងអប់រំ យុវជន និងកីឡា

វិទ្យាស្ថានបច្ចេកវិទ្យាកម្ពុជា

ដេប៉ាតឺម៉ង់ ទេពកោសល្យគណិតវិទ្យាអនុវត្ត និងស្ថិតិ

គម្រោងសញ្ញាបត្រវិស្វករ

ប្រធានបទ : ការបង្រៀនគំរូគំហើញនិងភាសាសម្រាប់រ៉ូបូតស្វ័យចល័ត

និស្សិត : កែវ វ័ន្តមុនីរ័ត្ន

ឯកទេស : វិទ្យាសាស្ត្រទិន្នន័យ

គ្រូទទួលបន្ទុក : លោក ឃាន វិសាល

ឆ្នាំសិក្សា : ២០២៤-២០២៥

MINISTERE DE L'EDUCATION, DE LA JEUNESSE ET DES SPORTS

INSTITUT DE TECHNOLOGIE DU CAMBODGE

DEPARTMENT DE MATHEMATIQUES APPLIQUEES ET STATISTIQUES

MEMOIRE DE FIN D'ETUDES

Titre : Fine-tuning le Modèle de Vision et de Langage pour Robot Mobile Autonome

Etudiant : KEO Vonmonyroth

Spécialité : Science des Données

Tuteur de stage : M. KHEAN Vesal

Année scolaire : 2024-2025

ក្រសួងអប់រំ យុវជន និងកីឡា
វិទ្យាស្ថានបច្ចេកវិទ្យាកម្ពុជា

ដេប៉ាតឺម៉ង់ ទេពកោសល្យគណិតវិទ្យាអនុវត្ត និងស្ថិតិ

គម្រោងសញ្ញាបត្រវិស្វកម្ម

របស់និស្សិត: កែវ ឌ័នមុនីរ័ត្ន

កាលបរិច្ឆេទការពារនិក្ខេបបទ: ថ្ងៃទី ០៩ ខែ កក្កដា ឆ្នាំ ២០២៥

អនុញ្ញាតឲ្យការពារគម្រោង

នាយកវិទ្យាស្ថាន: _____

ថ្ងៃទី ខែ ឆ្នាំ ២០២៥

ប្រធានបទ: ការបង្រៀនគំរូគំហើញនិងភាសាសម្រាប់រ៉ូបូតស្វ័យចល័ត

សហគ្រាស: ហ្វេកម៉ារី អេអាយ ឯ.ក

ប្រធានដេប៉ាតឺម៉ង់: **បណ្ឌិត លិន មន្តលសិរី** _____

សាស្ត្រាចារ្យដឹកនាំគម្រោង: **លោក ឃាន ឥសាល** _____

អ្នកទទួលខុសត្រូវក្នុងសហគ្រាស: **លោក ជឹម ថេងឌី** _____

រាជធានីភ្នំពេញ

MINISTERE DE L'EDUCATION, DE LA JEUNESSE ET DES SPORTS

INSTITUT DE TECHNOLOGIE DU CAMBODGE

DEPARTMENT DE MATHEMATIQUES APPLIQUEES ET STATISTIQUES

MEMOIRE DE FIN D'ETUDES

DE M. KEO Vonmonyroth

Date de soutenance : le 09 Juillet 2025

« Autorise la soutenance du mémoire »

Directeur de l'Institut:_____

Phnom Penh, le 2025

Titre : Fine-tuning le Modèle de Vision et de Langage pour Robot Mobile Autonome

Etablissement du stage : **FACTORYAI CO., LTD**

Chef du département : **Dr. LIN Mongkolsery** _____

Tuteur de stage : **M. KHEAN Vesal** _____

Responsable de l'établissement : **M. CHIM Thenggy** _____

PHNOM PENH

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to all those who have contributed to the successful completion of this research work.

First and foremost, I am deeply grateful to my academic advisor **Mr. KHEAN Vesal** and co-advisor , **Mrs. TANN Chantara**, for their invaluable guidance, unwavering support, and continuous encouragement throughout the entire internship. Their expertise and insightful feedback have been instrumental in shaping the direction and quality of this work.

Secondly, I would like to express my gratitude to my company Supervisors **Dr. KHUN Kimang** and **Mr. CHIM Thenggy**, for their trust, professional insights, and for providing a supportive and enriching work environment that allowed me to gain hands-on experience and grow professionally.

Thirdly, My heartfelt appreciation goes to the members of the thesis committee and reviewers for taking the time to evaluate my work and provide constructive feedback. Their comments have helped me improve the quality and rigor of this research.

Furthermore, I would also like to express my gratitude to the Department of Applied Mathematics and Statistics (AMS) at the Institute of Technology of Cambodia, particularly to **Dr. LIN Mongkolsery**, and lecturers in AMS who have taught and guided me throughout the program. I am also thankful to department secretary, **Ms. CHHEANG Sreypich**, for her administrative and academic support. I would like to extend my deepest gratitude to the rector of ITC, **Dr. PO Kimtho**, for leading an institution that fosters academic excellence and research.

Lastly, I would like to thank my friends, familly and colleagues who stood by me and supported me throughout this journey. Their kindness and encouragement have meant a lot to me, and I will always cherish the memories we made together.

In conclusion, the completion of this work would not have been possible without the collective efforts and support of all those mentioned above. I am deeply grateful for their contributions, and I acknowledge their significant impact on this work.

សេចក្តីសង្ខេប

ការអភិវឌ្ឍន៍ថ្មីៗក្នុងគំរូគំហើញនិងភាសា (vision language foundation models) បានបង្ហាញពីសមត្ថភាពរបស់វាក្នុងការយល់ដឹងពីទិន្នន័យច្រើនប្រភេទ និងដោះស្រាយការកិច្ចផ្សំគ្នារវាងគំហើញ និងភាសាដ៏ស្មុគស្មាញ រួមទាំងការគ្រប់គ្រងរូបូត (robotics manipulation)។ គម្រោងនេះផ្តោតសំខាន់ទៅលើការប្រើប្រាស់គំរូគំហើញនិងភាសា (VLMs) ដែលមានស្រាប់ ជាមួយនឹងការបង្រៀនលើទិន្នន័យរូបូត ដើម្បីធ្វើការគ្រប់គ្រងរូបូតស្វ័យចល័ត តាមរយៈការការបញ្ចូលរូបភាព និងសេចក្តីណែនាំជាអក្សរ។

គោលបំណងនៃគម្រោងនេះគឺធ្វើការបង្រៀនគំរូគំហើញនិងភាសា (VLMs) ឱ្យអាចបកប្រែសំណើរជារូបភាពបិងសេចក្តីណែនាំជាអក្សរទៅជាពាក្យបញ្ជារូបូត។ ដោយប្រើប្រាសការអនុវត្តបច្ចេកទេសបង្រៀនលើគំរូ VLM ដែលបានបណ្តុះបណ្តាលលើព័ត៌មានទូទៅរួចទៅហើយ គំរូនោះនឹងរៀនពីរលំនាំនៃរូបភាពដែលផ្សំជាមួយសេចក្តីណែនាំជាអក្សរនិងពាក្យបញ្ជារូបូតដែលជាពាក្យបញ្ជារដែលអាចអោយរូបូត ធ្វើការផ្លាស់ទី និងការបត់បែន។

ដើម្បីសម្រេចគោលដៅនៃគម្រោងនេះ ទិន្នន័យសម្រាប់រូបូតដែលមានរូបភាព និងសេចក្តីណែនាំជាអក្សរដើម្បីគាំទ្រការងារនេះ។ ទិន្នន័យនេះគ្របដណ្តប់លើស្ថានភាពជាច្រើនរបស់រូបភាពដែលថតចេញពីកាម៉ារូបូតនៅក្នុងបរិស្ថានសិប្បនិម្មិតរួមមានស្ថានភាពដែលវត្ថុគោលដៅអាចមើលឃើញ (កណ្តាល, ឆ្វេង, ស្តាំ) ឬមើលមិនឃើញគិតចេញពីទិដ្ឋភាពរបស់កាម៉េរារូបូត។ គ្រប់ទិន្នន័យត្រូវបានផ្គូផ្គងជាមួយពាក្យបញ្ជាដែលមានទម្រង់ជា JSON តំណាងឱ្យសកម្មភាពរូបូតដែលសាកសម។ គំរូដែលបានបង្រៀននឹងដើរតួនាទីជាម៉ូឌុលគោលការណ៍ (policy module) ដែលណែនាំឱ្យរូបូតអាចស្វែងរក និងធ្វើការផ្លាស់ទីដោយស្វ័យប្រវត្តិទៅកាន់វត្ថុដែលអ្នកប្រើប្រាស់ចង់អោយវារក។ នៅក្នុងករណីរបស់យើងគឺមានវត្ថុដូចជាធុងសំរាម ហិបដាក់សម្ភារៈពេទ្យ និងបាល់។

គំរូ VLMs ចំនួនពីរគឺ Qwen-2-VL-7B-Instruct និង Llava-V1.6-Mistral-7B ត្រូវបានជ្រើសរើសសម្រាប់ការសាកល្បង។ យើងធ្វើការបង្រៀនគំរូទាំងនេះដោយប្រើ LoRA ហើយវាស់វែងប្រសិទ្ធភាពរបស់វាតាមរយៈ Riemannian Distance សម្រាប់កំហុសការទស្សន៍ទាយនៃចម្ងាយ និងការបង្វិល។

លទ្ធផលពីការធ្វើពិសោធន៍បង្ហាញថា វិធីសាស្ត្រនេះអាចឱ្យគំរូគំហើញនិងភាសា (VLM) យល់ដឹងពីសំណើរជារូបភាពនិងសេចក្តីណែនាំជាអក្សរហើយបង្កើតពាក្យបញ្ជារូបូតបានត្រឹមត្រូវ។ ការងារនេះបានបង្ហាញពីសក្តានុពលនៃគំរូ VLMs ដែលបានបង្រៀនសម្រាប់រូបូតក្នុងជីវិតពិតប្រាកដ និងរួមចំណែកដល់ការអភិវឌ្ឍន៍នៃការយល់ដឹង និងការគ្រប់គ្រងប្រព័ន្ធស្វ័យប្រវត្តិ។

RESUME

Les progrès récents des modèles de base du langage visuel ont démontré leur capacité à comprendre des données multimodales et à résoudre des tâches complexes liées au langage visuel, notamment la manipulation robotique. Ce projet vise à exploiter les modèles de langage visuel (VLMs) existants en les ajustant simplement aux données robotiques afin de manipuler un robot autonome grâce à une compréhension multimodale des entrées d'images et des instructions textuelles.

L'objectif de ce projet est également d'ajuster les modèles de langage visuel afin qu'ils puissent traduire les instructions image/texte en commandes robotiques. En appliquant des techniques d'ajustement à un VLM pré-entraîné, le modèle apprend à associer des paires image/texte à des commandes robotiques mobiles exploitables telles que le mouvement et la rotation.

Pour atteindre les objectifs de ce projet, un ensemble de données personnalisé d'images et d'instructions textuelles spécifiques au spécifiques aux robots a été créé pour soutenir ce travail. Cet ensemble de données couvre divers scénarios d'images et de robots dans lesquels l'objet cible est visible (au centre, à gauche, à droite) ou non visible depuis la caméra du robot. Chaque échantillon de données est associé à une commande JSON structurée représentant une action appropriée du robot. Le modèle affiné servira de module de politique qui guidera le robot pour qu'il localise et se dirige de manière autonome vers l'objet spécifié dans son environnement, dans notre cas une poubelle, une trousse médicale et une balle.

Les résultats de l'expérience montrent que cette approche permet au VLM de comprendre les invites image-texte et de générer des commandes robotisées précises . Ce travail met en évidence le potentiel des VLM affinés dans les applications robotiques du monde réel et contribue à l'avancement de la perception et du contrôle multimodaux dans les systèmes autonomes.

ABSTRACT

Recent advancement in vision-language models (VLMs) has shown their ability to understand and precess multimodel data and resolve complicated vision language tasks, including robotics manipulation. This project aim to utilize of existing vision-language models (VLMs) by performing simple fine-tuning on robotics data to manipulate autonomous robot through multimodal understanding of image inputs and text instructions.

The goal of this project is to fine-tune vision-language models to be able to translate image/text instructions into robot commands. By applying fine-tuning techniques to a pre-trained VLM, the model learns to recognize patterns pattern of image/text pairs with actionable mobile robot commands such as movement and rotation.

To achieve the goals of this project, a custom dataset of images and text instruction for robot specific was created to support this work, the dataset covering various image and robot scenarios in which the target object is visible (center, left, right) or not visible from the robot's camera view. Each data sample is paired with a structured JSON command representing an appropriate robot action. The fine-tuned model will act as a policy module that guides the robot to autonomously locate and navigate toward the specified object in its environment, in our case it trash bin, medical kit, and ball.

Two open-source VLMs model, Qwen-2-VL-7B-Instruct and Llava-V1.6-Mistral-7B were selected for experment, we fine-tuned these with LoRA, and their performance was assessed using **Rieamnian Distance** for distance and rotation prediction errors.

The Experimental results demonstrate that the approach enables the VLM to understand image, text prompt related to image and generate accurate robot command . This work highlights the potential of fine-tuned VLMs in practical robotic applications and contributes to the progress of multimodal perception and control in autonomous systems.

ABBRVIATIONS AND SYMBOL

AI Artificial Intelligence

CNN Convolutional Neural Network

VLM Vision-Language Model

ROS Robot Operating System

LoRA Low Rank Adaptation

PEFT Parameter-Efficient Fine-Tunin

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1. PRESENTATION OF ORGANIZATION

1.1.1. Company Presentation

Established in 2023 and officially recognized under the Law of the Kingdom of Cambodia, Factory AI represents a national leap into the future of industrial innovation. Positioned at the forefront of Cambodia's AI transformation, Factory AI operates as a next-generation AI Factory framework inspired by global leaders like NVIDIA focused on centralizing and accelerating the development, deployment, and integration of artificial intelligence solutions across industries.

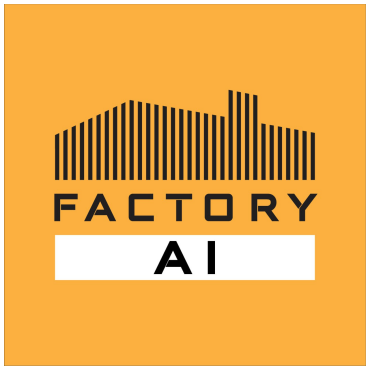


Figure 1.1 FactoryAI logo

1.1.2. Service

FactoryAI is a systematic, scalable model for producing AI-driven solutions. It brings together the four pillars of AI success people, data, platform, and process to deliver intelligent systems that optimize business functions and industrial operations. Much like how pharmaceutical factory delivers medicine to patients, an AI factory generates tailored AI solutions to solve real-world problems ranging from handheld assistants to predictive analytics in weather forecasting and industrial automation.

1.1.3. Mission & vision

To build a Cambodian-led GPT (Generative Pretrained Transformer) platform for Industry 4.0, laying the groundwork for AI-driven manufacturing, smart robotics, and intelligent

infrastructure across Southeast Asia and beyond. And to establish a Factory AI 4.0 ecosystem capable of producing:

1.1.4. Address & Contact

Head office address: THE FARM, FACTORY 3, RD. 39D, RING ROAD 2, DANGKOR, PHNOM PENH, KINGDOM OF CAMBODIA.

Contacts:

Facebook: https://www.facebook.com/factoryai.dev/

Telegram: https://t.me/factoryai

LinkedIn: https://www.linkedin.com/showcase/factoryaidev/

Figure 1.2 FactoryAI address

1.2. PROJECT PRESENTATION

1.2.1. Overview

Recent progress in vision-language foundation models (VLM) has presented their exhilarating ability in modeling and aligning the representation of images and words, and the unlimited potential to resolve a wide range of downstream tasks with multi-modality data, for instance, visual question-answering, image captioning, human-agent interactions. These successes, undeniably, encourage people to imagine a generalist robot equipped with such a vision-language comprehension ability to interact naturally with humans and perform complex manipulation tasks.

Fine-tuning VLM for an Autonomous Mobile Robot project aims to apply a technique called fine-tuning to existing Vision-Language Models (VLMs) so they can translate visual input and natural language instructions (image/text) into robot control commands. The fine-tuned model will serve as a manipulation policy that enables the robot to perform tasks such as navigating toward a target object.

By fine-tuning a pre-trained VLM on a custom dataset of annotated robot-camera images and corresponding action commands, the model learns to interpret image/text input and map it

to low-level behaviors such as movement and rotation. The dataset includes scenarios where the target object appears in different positions within the camera frame (left, right, center) or is not visible at all.

As a result of this fine-tuning process, the model effectively learned to act as a manipulation policy for navigation. When the object appears on the left side of the camera frame, the fine-tuned VLM outputs a command instructing the robot to turn left; when it is on the right, the model outputs a command to turn right. If the object is directly in front, the VLM generates a command to move forward toward it. In scenarios where the object is not visible, the model instructs the robot to rotate in place to search for the target object.

These results demonstrate that the model can function as an effective policy for real-time navigation, showing the potential of vision-language-based learning in robotics.

1.2.2. Problem Statement

With recent progress of vision-language models (VLMs), most VLM pre-trained models are not directly applicable to real-time robotic manipulation due to domain mismatch since they are trained only for general task like Captioning and summarization, Image generation ,Image search and retrieval, Image segmentation, Object detection , Visual question answering (VQA) not robot specific task.

Another challenge is enabling a mobile robot with only a front-facing camera and simple movement and rotation commands to locate and approach a target object using a single multi-modal instruction (image/text). This requires the VLM to correctly perceive spatial cues, infer intent, and translate that understanding into actionable robot commands. Without task-specific fine-tuning, the model may fail to produce reliable or safe behavior in real-time environments.

1.2.3. Project Objective

The objective of this project is to fine-tune a vision-language model for robotic control based on multi-modal input. The specific goals are

- **Model Adaptation**: Fine-tune a pre-trained VLM to map visual observations and language instructions to actionable robot commands.
- **Dataset Development**: Construct a custom robot specific task dataset of image of robot-perspective, the dataset is then labeled with associate robot command and categorized by object visibility and position (left, right, center, not visible).

- **Autonomous Navigation**: Design a simple control policy that allows the robot to search, detect, and approach the target object such as soccer ball, medical kit and trash bin based on model output.
- **Evaluation**: Assess the model's accuracy in predicting appropriate actions and evaluate the overall system performance in simulated environment.

1.2.4. Planning

During the internship, it is important to identify the main objective that needs to be achieved. This objective serves as the guiding principle for the planning process. Once the objective is established, it is essential to determine the overall project requirements and understand the scope of work involved. his understanding forms the basis for creating a well-structured and actionable plan that includes specific tasks, key milestones, and a realistic timeline. By conducting thorough planning after clarifying the internship's main objective and project needs, the internship period can be utilized efficiently and effectively.

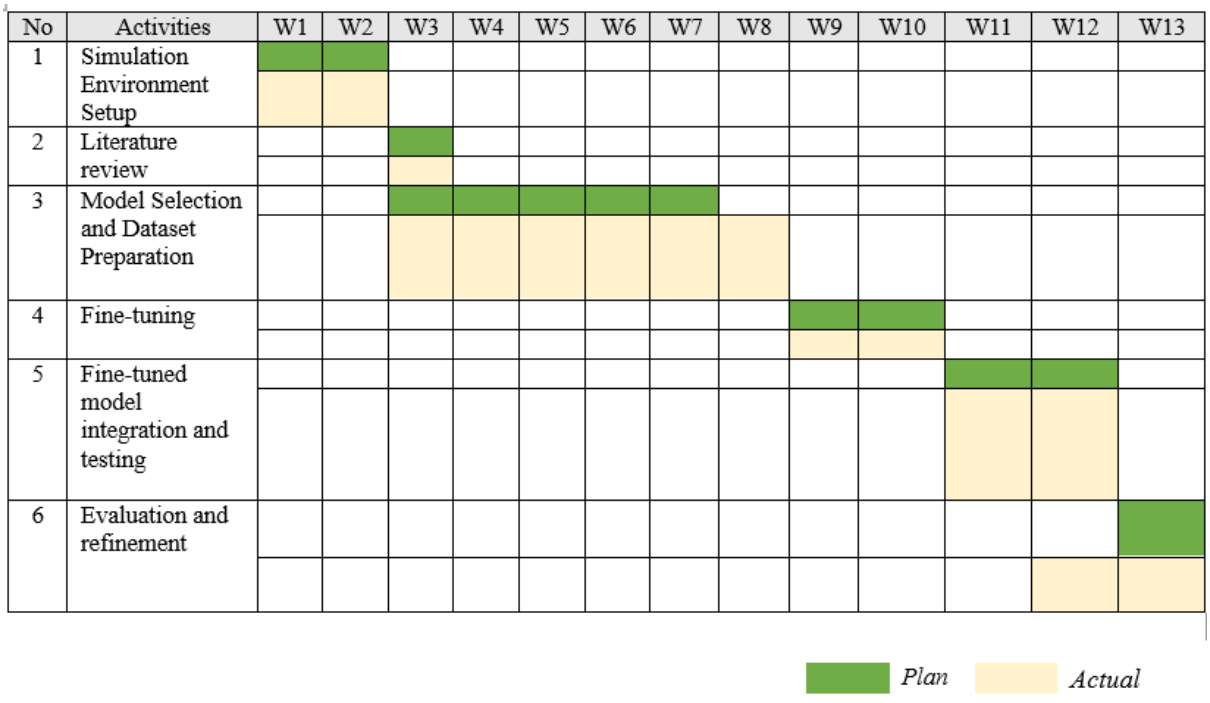


Figure 1.3 Internship Plan

Chapter 1

LITERATURE REVIEW

The purpose of this chapter is to critically analyze and synthesize the existing body of research on fine-tuning technique and the using VLM-based for robot control in order to provide a comprehensive understanding of the current state of knowledge, identity gaps for further investigation.

2.1. ViSION LANGUAGE MODEL

The rise of powerful Large Language Models (LLMs) like GPT and LLaMA in recent years are now able to solve such a large variety of tasks that their usage is becoming more and more popular. Such models that were mostly limited to text inputs are now extended to having visual inputs. Connecting vision to language will unlock several applications that will be key to the current AI-based technological revolution. VLMs extend these capabilities by adding visual understanding, enabling the development of AI agents that can perceive and reason about the world using both images and text. (Bordes et al., 2024)

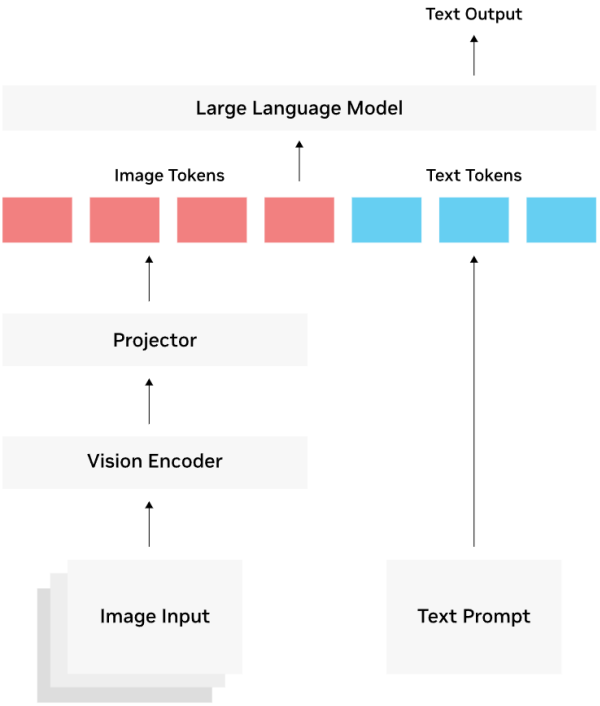


Figure 2.4 common three-part architecture for vision language models. Source: NVIDIA Glossary, 2024, https://www.nvidia.com/en-us/glossary/vision-language-models/

2.2. VLMS FOR ROBOTIC CONTROL

Recent progress in vision-language foundation models (VLM) has presented their exhilarating ability in modeling and aligning the representation of images and words, and the unlimited potential to resolve a wide range of downstream tasks with multi-modality data, for instance, visual questionanswering, image captioning, human-agent interactions. These successes, undeniably, encourage people to imagine a generalist robot equipped with such a vision-language comprehension ability to interact naturally with humans and perform complex manipulation tasks. (Li et al., 2024)

Brohan et al. (2023) introduced RT-2, a family of Vision-Language-Action (VLA) models that directly map visual and textual inputs to low-level robot actions, while leveraging knowledge acquired from web-scale pre-training. Traditional robotic control systems often rely on task-specific architectures or modular pipelines where highlevel planning is decoupled from low-level action execution. These approaches typically do not benefit from the rich semantic understanding and emergent reasoning capabilities demonstrated by large language models (LLMs) and VLMs trained on internet-scale data. However, collecting sufficient real-world robotic interaction data to match the scale of such pre-trained models is infeasible due to time, cost, and safety constraints. Thus, there is a growing interest in exploring how existing VLMs can be adapted to directly perform closed-loop robotic control. Brohan et al. (2023) address this challenge by proposing a unified framework in which both natural language responses and robotic actions are treated as sequences of tokens within the same model architecture. This enables the direct fine-tuning of state-of-the-art VLMs on robotic trajectory data, alongside Internet-scale visionlanguage tasks like visual question answering (VQA). The authors build upon the Robotics Transformer (RT-1) architecture but extend it by incorporating large pre-trained VLMs such as PaLI-X and PaLM-E . The key innovation lies in treating robot actions as text tokens similar to words in a sentence and integrating them into the training set. This allows the model to learn a joint distribution over language, vision, and actions. During training, the model is co-fine-tuned on both robotic manipulation data and diverse web-scale vision-language datasets. The robotic data consists of annotated trajectories collected from real-world kitchen environments, including tasks such as picking, placing, opening drawers, and moving objects. The output space is constrained during inference to ensure valid robot actions when executing commands, while allowing full natural language generation for standard VQA tasks. RT-2 demonstrates impressive generalization and reasoning capabilities, it is limited in its abil-

ity to learn fundamentally new physical skills beyond those present in the robot demonstration data. Its physical behavior remains constrained to the distribution of skills seen during training, although these skills can be deployed in novel semantic contexts. Additionally, the computational cost of running such large models in realtime remains a bottleneck for deployment in high-frequency control settings.

2.3. MOBILE ROBOT NAVIGATION AND VISION-BASED CONTROL

Tan et al (2024), introduce HAM-Nav, a novel architecture that leverages pre-trained vision-language models (VLMs) to enable mobile robots to navigate using hand-drawn maps across diverse environments and drawing styles. Traditional map-based robot navigation systems typically rely on metrically accurate maps generated by sensors such as LiDAR or RGB-D cameras. These approaches are not well-suited for interpreting hand-drawn maps, which lack precise geometric fidelity but are easy for humans to create and share. Existing methods that attempt to use hand-drawn maps often require strict constraints, such as predefined symbols or simplified environments, limiting their applicability in real-world settings. To bridge this gap HAM-Nav was proposed by leveraging recent advances in VLMs, which have demonstrated strong performance in understanding both visual and textual modalities. The goal is to develop a system that can interpret abstract, imprecise hand-drawn maps and translate them into actionable navigation instructions for robots. The core innovation of HAM-Nav lies in its Selective Visual Prompting (SVP) module, which enables the model to focus on relevant visual elements in the hand-drawn map while ignoring irrelevant or misleading artifacts. The architecture integrates:

- A pre-trained VLM (BLIP-2) for multimodal reasoning.
- A spatial alignment mechanism to match the robot's real-time perception with the drawn map.
- A path planning interface that generates high-level commands based on the interpreted map.

Previous approaches require conversion of hand-drawn maps into metrically accurate representations, HAM-Nav operates directly on raw sketches, making it robust to variations in style and accuracy. HAM-Nav shows promising results, yet it still requires some level of semantic consistency between the drawn map and the real environment. Extremely abstract or poorly annotated maps may lead to navigation errors. Additionally, the current implementation does

not support dynamic obstacle avoidance or real-time replanning, focusing instead on high-level route interpretation.

2.4. VLM FINE-TUNING TECHNIQUES

Training VLMs has shown great effectiveness in cross-domain vision and language tasks. However, as the size of pre-trained models continues to grow, fine-tuning the entire parameter set of these models becomes impractical due to computational constraints. To address this challenge, Parameter-Efficient Fine-Tuning (PEFT) methods have been developed to address the high computational cost associated with fine-tuning large-scale models. These methods focus on training a subset of parameters, rather than the entire model, to adapt to downstream tasks.

Low Rank Adapters (LoRa) is recognized as a popular method for parameter fine-tuning. LoRA can be applied to both pure language models and vision-language models. Several variants of LoRA have been developed to enhance its functionality and efficiency. One such variant is QLoRA, which integrates LoRA with a quantized backbone and enables the back-propagation of gradients through a frozen, 4-bit quantized pre-trained language model into LoRA. Another variant is VeRA, which is designed to reduce the number of trainable parameters in comparison to LoRA, while maintaining equivalent performance levels. This is achieved by utilizing a single pair of low-rank matrices shared across all layers and learning small scaling vectors instead. Lastly, DoRA decomposes the 29 pre-trained weight into two components, magnitude and direction, for fine-tuning. DoRA has demonstrated the capability to generalize Low-rank adaptation methods from language models to Vision-Language benchmarks through empirical experiments. (Hu et al., 2022)

Prompt-based methods, another method for efficient fine-tuning is linked with prompting. Zhou et al. (2022) introduce Context Optimization (CoOp), a technique designed to adapt large pre-trained vision language models, such as CLIP, for downstream image recognition tasks, eliminating the need for manual prompt engineering. CoOp optimizes the context words of the prompt using learnable vectors during the training process. The method provides two implementations: unified context and class-specific context. Experimental results from 11 datasets indicate that CoOp outperforms hand-crafted prompts and linear probe models in few-shot learning. Additionally, it exhibits superior domain generalization capabilities compared to zero-shot models that utilize manual prompts. Then Jia et al. (2022) present Visual Prompt Tun-

ing (VPT), for adapting large-scale Transformer models in vision. Contrary to the conventional approach of full fine-tuning, which updates all backbone parameters, VPT introduces a minimal amount (less than 1% of model parameters) of trainable parameters in the input space. This is achieved while keeping the model backbone frozen, and in many instances. VPT demonstrates comparable or even superior accuracy to full fine-tuning.

Adapter-based methods. Adapters refer to new modules added between layers of a pre-trained network. Specifically, in the vision-language model domain, CLIP-Adapter fine-tunes with feature adapters on either the visual or language branch. It adopts an additional bottleneck layer to learn new features and performs residual-style feature blending with the original pre-trained features. In addition, VL-adapter evaluates various adapter-based methodologies, within a unified multi-task framework across a diverse range of image-text and video-text benchmark tasks. The study further delves into the concept of weight-sharing between tasks as a strategy to augment the efficiency and performance of these adapters. Empirical results indicate that the application of the weight-sharing technique in conjunction with adapters can effectively rival the performance of full fine-tuning, while necessitating updates to only a minimal fraction of the total parameters (4.18% for image-text tasksand 3.39% for video-text tasks). Subsequently, LLaMA-Adapter V2 proposes a parameter-efficient visual instruction model that enhances large language models' multi-modal reasoning capabilities without requiring extensive parameters or multi-modal training data. It proposes unlocking more learnable parameters (e.g., norm,bias, and scale) and an early fusion method to incorporate visual tokens into LLM layers. Compared to other full-fine-tuning approaches like MiniGPT-4 and LLaVA, LLaMA-Adapter V2 involves much fewer additional parameters. (Bordes et al., 2024)

Mapping-based methods. Injecting trainable modules into pretrained models through adapters or LoRA requires some knowledge of the network's architecture to decide where to insert or adapt parameters. In the context of VLMs, (Mañas et al. 2023) and (Merullo et al. 2022) propose a simpler approach which only requires training a mapping between 30 pretrained unimodal modules (i.e., vision encoders and LLMs), while keeping them completely frozen and free of adapter layers. In addition, this method requires fewer trainable parameters and leads to increased data-efficiency.

2.5. SUMMARY

In this chapter, we reviewed several papers about Vision-Language Models and Robotics. Through this review we got a lot of ideals for implementation the fine-tuning VLMs for autonomous mobile robot.

Table 2.1. Summary of Papers on Vision-Language Models and Robotics

Authors	**Titles**	**Description**
Brohan et al., 2023	Vision-Language-Action Models Transfer Web Knowledge to Robotic Control	RT-2 is a vision-language-action model that robot control by training on Internet-scale data.
Tan et al., 2024	Mobile Robot Navigation Using Hand-Drawn Maps	HAM-Nav is a VLM-based system enables robots to navigate using hand-drawn maps.
Bordes et al., 2024	An Introduction to Vision-Language Modeling	This paper introduces VLMs, explaining their workings, training, evaluation, challenges, fine-tuning method.

Chapter 3

TECHNOLOGY AND FRAMEWORK

In this chapter, we present all the technologies and frameworks that we will use to implementation of fine-tuning VML for autonomous.

3.1. GAZEBO

Gazebo is a 3D dynamic simulator with the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. While similar to game engines, Gazebo offers physics simulation at a much higher degree of fidelity, a suite of sensors, and interfaces for both users and programs.

Figure 3.5 Gazebo Logo

3.2. ROBOT OPERATING SYSTEM

The robot operating system (ROS) is a flexible and powerful framework designed for robotics software development. ROS is not an operating system in the traditional sense, it's a middleware that operates on top of a conventional operating system such as Linux. It provides a set of libraries and tools that help developers create complex and robust robot applications. ROS consists of a collection of software frameworks for various functionalities required in robotics, such as hardware abstraction, device drivers, communication between processes, package management, and more. It facilitates the creation of modular and reusable code, enabling developers to focus on specific components of a robotic system without reinventing the wheel.

Figure 3.6 ROS Logo

3.3. RVIZ

RViz is a 3D visualization tool for the Robot Operating System (ROS). It allows users to visualize robot models, sensor data, and other information within a 3D environment, making it a crucial tool for debugging and understanding robot behavior.

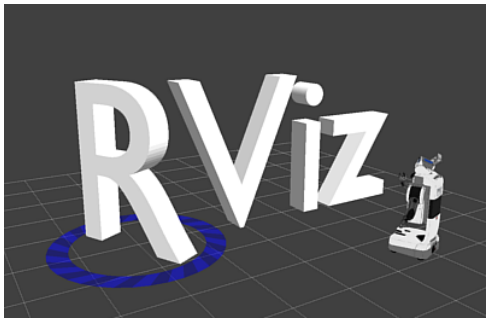


Figure 3.7 Rviz Logo

3.4. HUGGING FACE

Hugging Face is a machine learning (ML), Large Language Model (LLM), vision Language Models (VLMs) and data science platform and community that helps users build, deploy and train machine learning models. It provides the infrastructure to demo, run and deploy artificial intelligence (AI) in live applications. Users can also browse through models and data sets that other people have uploaded. Hugging Face is often called the GitHub of machine learning because it lets developers share and test their work openly. Hugging Face is known for its Transformers Python library, which simplifies the process of downloading and training ML models. The library gives developers an efficient way to include one of the ML models hosted on Hugging Face in their workflow and create ML pipelines. The platform is important because of its open source nature and deployment tools. It allows users to share resources, models and research and to reduce model training time, resource consumption and environmental impact of AI development.

Figure 3.8 Hugging Face logo

3.5. UNSLOTH

Unsloth.ai is an innovative AI startup that optimizes the training and fine-tuning of LLMs, VLMs. The platform boasts significant improvements in speed and memory usage, making it a valuable tool for AI researchers and developers. Unsloth.ai achieves these enhancements through advanced mathematical derivations and handwritten GPU kernels using OpenAI's Triton language.

Unsloth some key features and benefits:

- **Speed and Efficiency**: Unsloth.ai can accelerate the fine-tuning process up to 30 times compared to traditional methods. This is achieved without compromising accuracy, thanks to manual differentiation and optimized GPU kernels. For instance, tasks typically taking 23 hours on a Tesla T4 GPU can be completed in just over 2 hours using Unsloth.
- **Memory Optimization**: The platform significantly reduces memory usage, allowing larger batch sizes and longer context windows. This optimization is particularly beneficial for handling large datasets and complex models. Unsloth can slash peak memory usage by up to 74%, making it possible to fine-tune large models on hardware with limited resources.
- **Broad Compatibility**: Unsloth supports various GPUs from NVIDIA, AMD, and Intel, ensuring users can leverage the platform regardless of their hardware setup. It is also fully compatible with the Hugging Face ecosystem, enabling seamless integration with popular tools like transformers, PEFT, and TRL.
- **Open Source and Community-Driven**: Unsloth offers a free, open-source version that

allows users to experience its benefits without any initial cost. The platform also fosters a vibrant community of developers and researchers contributing to its continuous improvement. Users can access many resources, including detailed benchmarks and reproducible notebooks, on GitHub and the Unsloth blog.

Figure 3.9 Unsloth Logo

3.6. WEIGHTS & BIASES

Weights & Biases (W&B) is a platform designed to help machine learning (ML) and deep learning practitioners track experiments, visualize results, manage datasets, and collaborate more effectively . It's widely used by researchers and developers to streamline the ML development life cycle. Weights & Biases was used for experiment tracking, logging, and visualization of loss curves, accuracy, and other metrics during training.

Figure 3.10 Weights & Biases Logo

3.7. GOOGLE COLAB

Google Colaboratory, or Colab, is an as-a-service version of Jupyter Notebook that enables user to write and execute Python code through browser. Jupyter Notebook is a free, open source creation from the Jupyter Project. A Jupyter notebook is like an interactive laboratory notebook that includes not just notes and data, but also code that can manipulate the data. The code can be executed within the notebook, which, in turn, can capture the code output. Applications such as Matlab and Mathematica pioneered this model, but unlike those applications, Jupyter is a browser-based web application.

Figure 3.11 Colab Logo

3.8. PYTHON

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured, objectoriented, and functional programming. Python was used as the main programming language for all deep learning and robotics components due to its rich ecosystem and extensive support in AI and robotics development.

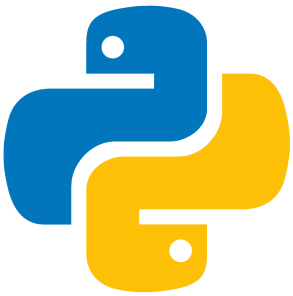


Figure 3.12 Python Logo

3.9. VISUAL STUDIO CODE

Visual Studio Code (VS Code) is a free, open-source, and highly customizable code editor developed by Microsoft. It's designed to run on Windows, macOS, and Linux, and it's known for its rich set of features and extensive ecosystem of extensions.

Chapter 3

METHODOLOGY

4.1. PROJECT STRUCTURE

Establishing a robust research methodology is foundational to the success of any experiment. To conduct experiments effectively, it's imperative to have a well-defined roadmap for the research process. This roadmap typically involves several key steps performed in a clear and sequential order. The research structure of the study is detailed in Figure 4.11.

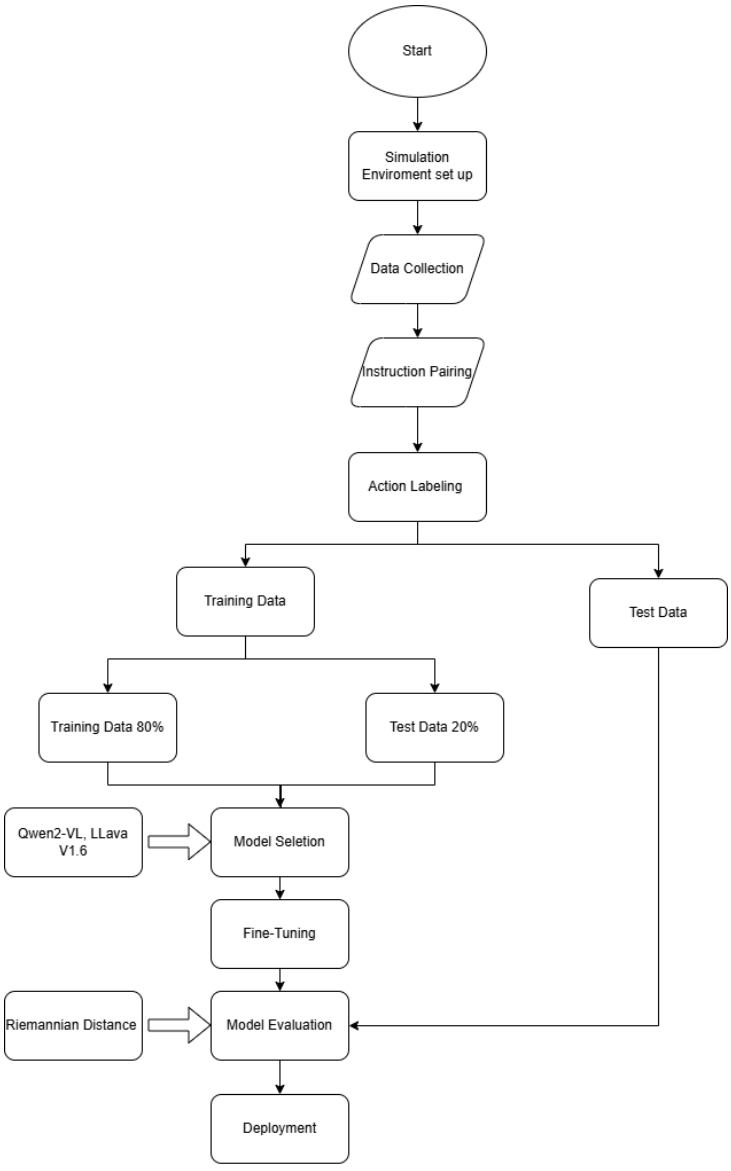


Figure 4.13 Project Structure

4.2. MODEL AND DATA USED

4.2.1. Data Collection

To fine-tune vision-language model, a custom dataset was created by simulating various spatial relationships between the robot and the target objects. Images were captured directly from the robot's front-facing camera within a Gazebo-based simulated environment. Each image was named based on the visibility and relative position of the target object. The dataset was organized into four main categories, with a consistent naming convention used to facilitate automatic labeling.

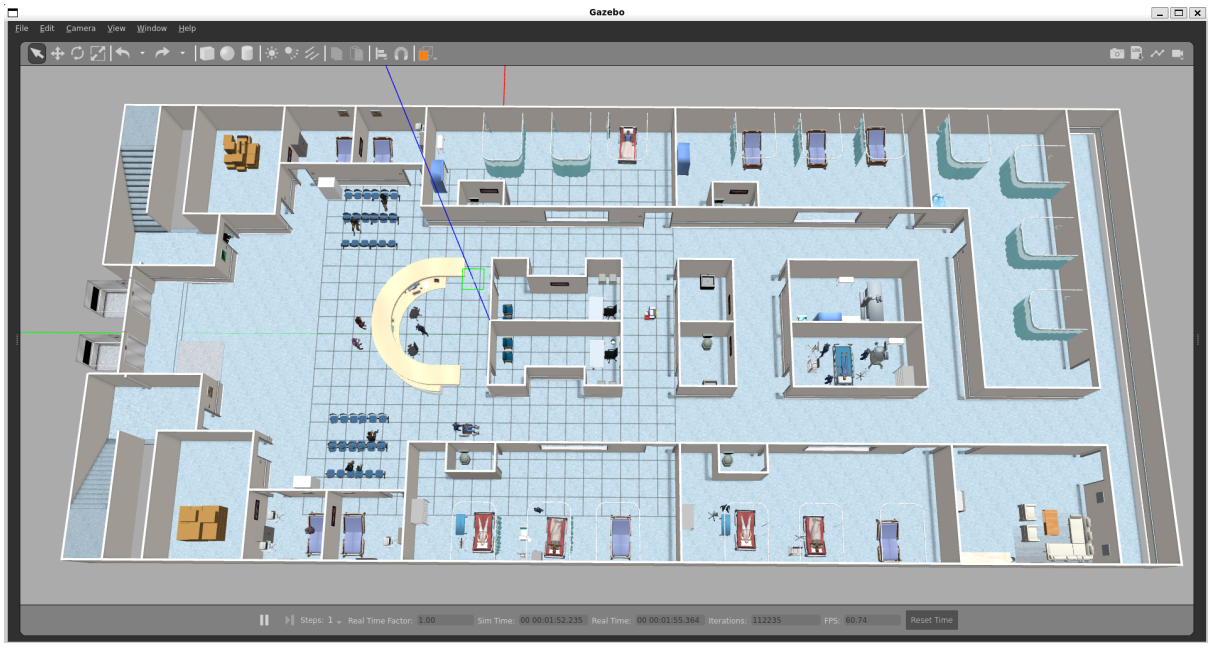


Figure 4.14 Hospital Simulation Environment in Gazebo

Object not visible In this scenario, the target object is either completely out of view or occluded, and the robot's camera captures empty scenes or unrelated objects. These images simulate the condition where the robot must rotate to begin a search.

Target Object centered the target object is fully visible and positioned at the center of the robot's camera view, ensuring that the robot can reach the object by moving straight forward. These images represent a clear goal state for navigation.To provide accurate distance supervision, a depth camera script written in Python was used to calculate the precise distance between the robot and the target object at the time of image capture. This distance value was then embedded directly into the image filename to facilitate labeling during data preprocessing.

Object Visible on the Left These images simulate situations where the target object is located in the left portion of the robot's camera frame, indicating the need to ratate left.

Object Visible on the Right Similar to the left case, this category contains images where the object appears in the right portion of the camera view, implying the robot should rotate to the right.

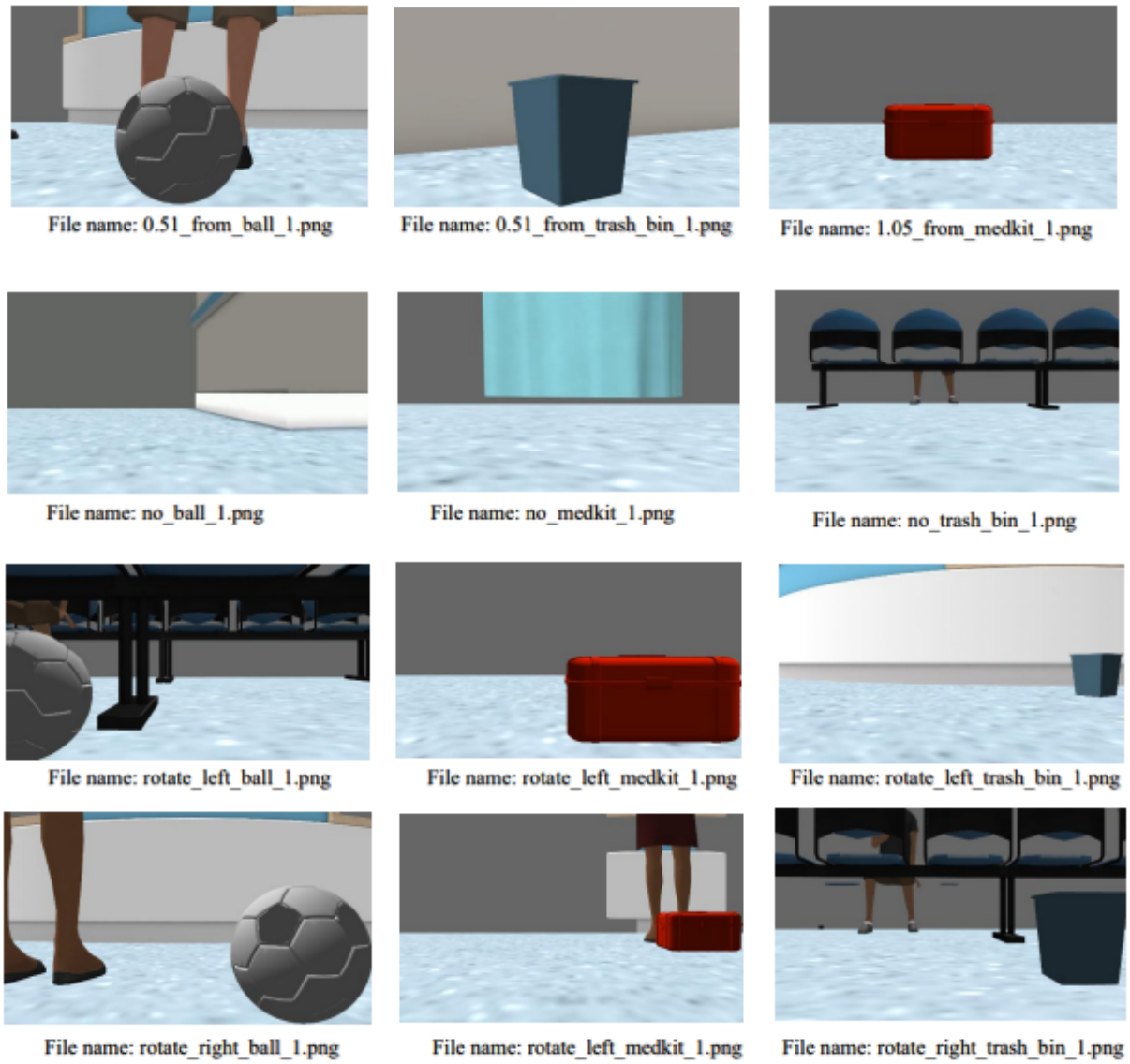


Figure 4.15 Sample Images

4.2.2. Data Annotations

To enable supervised fine-tuning of the vision-language model (VLM) for autonomous mobile robot control, each sample in the dataset was annotated with structured action commands

in response to a given textual instruction and corresponding image frame. The objective of these annotations is to train the VLM to generate the correct movement or rotation command based on the robot's visual input and the user's prompt.

Language Prompt

Each image is paired with a consistent instruction prompt, typically in the form of a goal-directed command, for example:

- Go to the ball slowly.
- Go to the ball as fast as you can.
- Find the medical kit.
- Move around until you see a medical kit.
- Move to locate the nearest trash bin.
- Find the nearest waste bin and go there.

These prompts are designed to simulate natural language instructions that a user might give to a robot during task execution.

Action Labels

For every image, text pair, a corresponding action label is created in JSON format. This label represents the robot's intended behavior based on its current visual and the user instruction. The structure includes two key components

- `"action"`: Specifies the type of action which are `"move"` or `"rotate"`
- `"params"`: Contains parameters for that action, such as movement distance or rotation angle

All annotations were programmatically generated using a custom Python script to ensure consistency and reduce manual effort. The process follows these steps:

- **Image Filename Parsing**:

  The script first iterates through all image files in the dataset and extracts filename using string parsing. The naming convention of the files provides clues to the label, for example:

  - **rotate_left_medkit_1.png**: Indicate labeling with rotation to left command.
  - **1.50_from_ball_67.png**: Indicate labeling move forward 1.5 meters command.

- **Instruction Pairing**

  For each image, a text instruction is automatically paired using a custom Python script

based on the object name. For example, any image related to the ”ball” will be paired with the ball instruction.

- **Label Generation**

  Based on the parsed filename, the script generates a corresponding JSON-formatted robot command

Examples of Annotations:

- If the object is directly in front at a known distance corresponding action label is:

  `{"action": "move", "params": {"distance": 1.5}}`

- If the object is seen on the left corresponding action label is:

  `{"action": "rotate", "params": {"angle": 10, "is_clockwise": false}}`

- If the object is on the right corresponding action label is:

  `{"action": "rotate", "params": {"angle": 10, "is_clockwise": true}}`

- If the object is not visible corresponding action label is:

  `{"action": "rotate", "params": {"angle": 30, "is_clockwise": false}}`

All image paths, instruction texts, and their corresponding action labels are saved into a CSV file, with three columns: `image_file_name`, `instruction`, `output`. This structured format is then used to load the data into the VLM fine-tuning pipeline. This dataset

Table 4.1. Overview of the Custom Robot Control Dataset

Target Object	No Target Object	Stop	Move Forward	Rotate Left	Rotate Right	Total
Trash Bin	100	50	100	50	50	350
Ball	100	50	100	50	50	350
Medical Kit	100	50	100	50	50	350
Total	300	150	300	150	150	**1050**

forms the foundation for vision-language models capble of understanding multimodel inputs and generating corresponding robotic control commands.

4.3. MODEL

For this project, two base vision-language model was chosen for fine-tuning, which are **Qwen2-VL-7B**, **Llava-v1.6-Mistral-7B**, open-source, multimodal model capable of processing both textual prompts and visual inputs. The selection of these models was guided by several practical and technical considerations aligned with the project’s goals and resource constraints.

Firstly, These two models offers strong balance between performance and efficiency.

With 7 billion parameters, it is significantly more lightweight than larger foundation models, making it feasible to fine-tune and deploy in environments with limited computational resources.

Secondly, the model is supported by the Unsloth framework, which provides tools for efficient and accelerated fine-tuning. Unsloth also enables memory-efficient training through integration with LoRA (Low-Rank Adaptation), allowing only a small subset of the model parameters to be updated during fine-tuning. This approach not only reduces hardware requirements but also significantly speeds up training time.

Finally, the open-source nature of **Qwen2-VL-7B** and **Llava-v1.6-Mistral-7B** ensures, flexibility, transparency, and reproducibility key qualities for system integration with custom robotic platforms.

4.3.1. Qwen2-VL-7B-Instruct

Qwen2-VL-7B-Instruct is the multimodal large language model series developed by Qwen team, Alibaba Cloud. Qwen2-VL is the latest version of the vision language models in the Qwen model families. As part of the Qwen series, Qwen2-VL come with agent that can operate on mobiles, robots with the abilities of complex reasoning and decision making, Qwen2-VL can be integrated with devices like mobile phones, robots, for automatic operation based on visual environment and text instructions. Qwen2-VL introduced several models simultaneously, including Qwen2-VL-2B and Qwen2-VL-7B under the Apache 2.0 license, as well as Qwen2-VL-72B under the Qwen license. In this project, cost-efficiency is a primary concern, so we exclusively focus on the 7-billionparameter family. This family includes various models, Qwen2-VL-7B, Qwen2-VL-7B-Instruct, Qwen2-VL-7B-Instruct-AWQ. Our primary interest lies in Qwen2-VL-7B-Instruct, as it is fine-tuned on an instruction dataset.

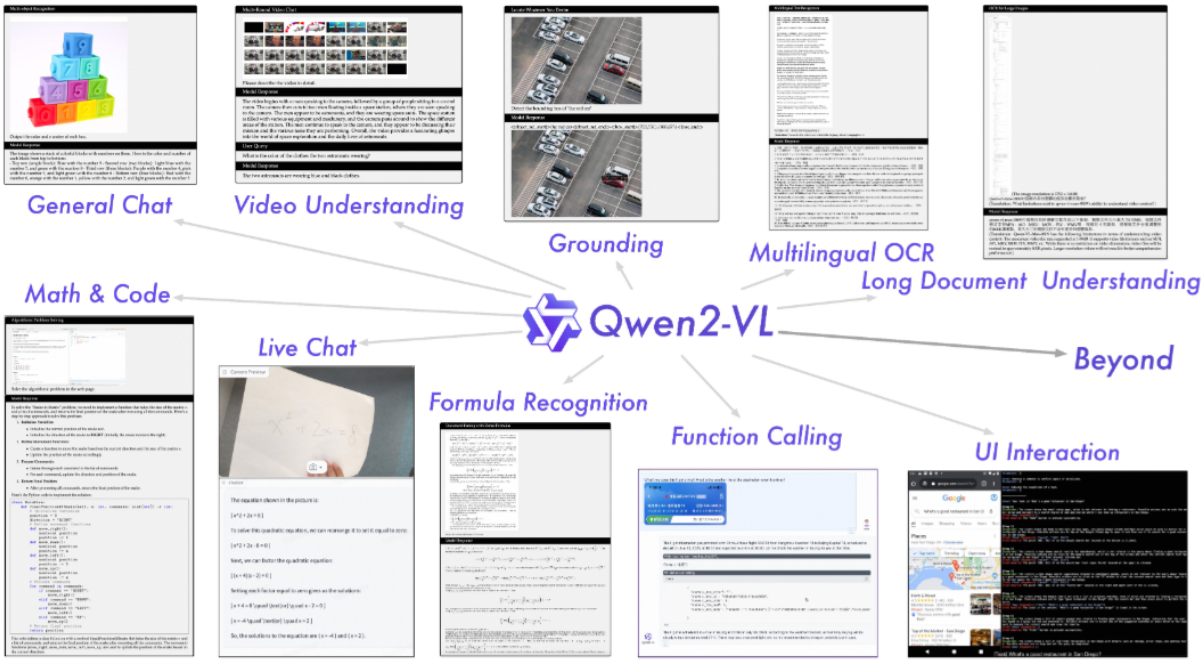


Figure 4.16 Qwen2-VL capabilities (Wang et al., 2024)

In Qwen2-VL, input images are tokenized and combined with the prompt text, then transformed into latent representations using a Vision Encoder before being fed into the QwenLM Decoder. It also supports videos, where up to 30 frames can be tokenized together. (Wang et al., 2024)

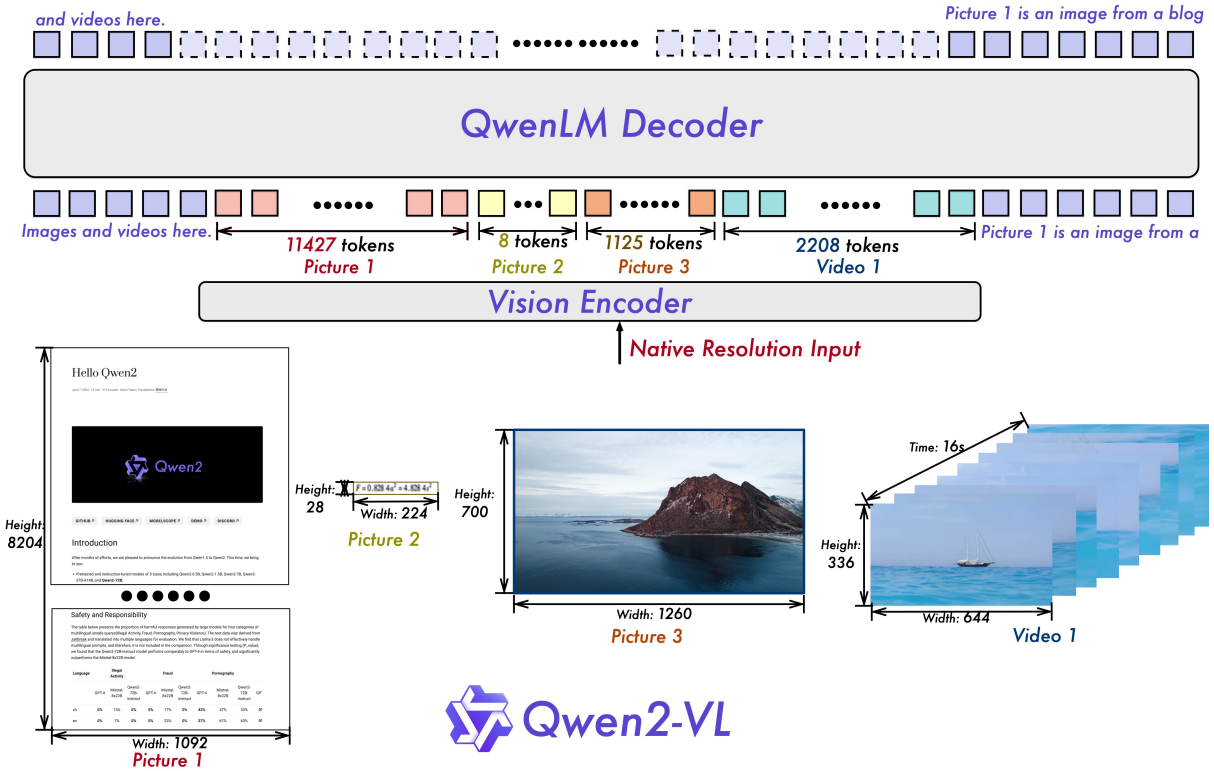


Figure 4.17 Qwen2-VL (Wang et al., 2023)

4.3.2. LLaVA-NeXT

Figure 4.18 LLaVa-NeXT Logo (Liu et al., 2024)

LLaVa-NeXT (also called LLaVa-1.6) improves upon LLaVa-1.5 by increasing the input image resolution and training on an improved visual instruction tuning dataset to improve OCR and common sense reasoning. LLaVa combines a pre-trained large language model with a pre-trained vision encoder for multimodal chatbot use cases. LLaVA 1.6 improves on LLaVA 1.5 BY, Using Mistral-7B (a powerful and efficient LLM developed by Mistral AI) and Nous-Hermes-2-Yi-34B which has better commercial licenses, and bilingual support, More diverse and high quality data mixture, Dynamic high resolution. (Liu et al., 2024)

4.4. PIPELINE & PRACTICAL INSIGHTS

Moving from pre-trained language models to practical applications typically requires resource-intensive fine-tuning processes. In this section, we delve into how our project tackled these challenges, with a primary focus on harnessing the HuggingFace ecosystem, the ”PEFT” library, the Low Rank Adaptation (LoRA) technique and traing pipeline with unsloth frameworks.

4.4.1. HuggingFace and the PEFT Library

HuggingFace, a prominent entity within the AI community, serves multiple roles. It acts as a repository for a vast collection of pre-trained machine learning models, thereby making these models accessible on a global scale. But HuggingFace also provides an extensive suite of tools and programming libraries that facilitate the training and inference of these models, ef-

Table 4.2. Parameters and their descriptions for LoRA (Low-Rank Adaptation).

Parameter	Description
r (rank)	Dimension of the low-rank matrices used for weight updates. Typically between 4-32. Lower values provide more compression but potentially less expressiveness.
lora_alpha	Scaling factor for LoRA layers, usually set to 2x the rank value. Higher values result in stronger adaptation effects.
lora_dropout	Dropout probability for LoRA layers, typically 0.05-0.1. Higher values help prevent overfitting during training.
bias	Controls training of bias terms. Options are “none”, “all”, or “lora_only”. “none” is most common for memory efficiency.
target_modules	Specifies which model modules to apply LoRA to. Can be ”vision_layers”, ”language_layers”, ”attention_modules”, ”mlp_modules”. More modules enable greater adaptability but increase memory usage.

fectively establishing itself as a central hub for the AI community. Within this ecosystem, the ”PEFT” library, or Parameter-Efficient Fine-Tuning, emerges as a pivotal resource. It is tailored specifically for the efficient adaptation of pre-trained language models to a diverse range of downstream applications. What distinguishes PEFT is its ability to achieve this adaptation without necessitating the fine-tuning of the entire model’s parameters. Instead, PEFT concentrates on fine-tuning a relatively small number of additional model parameters, thus leading to a significant reduction in computational and storage costs. A substantial portion of these cost savings is achieved through the implementation of LoRA.

4.4.2. Training Pipeline

In this project the training pipeline for fine-tuning VLMs model follow a systematic process, this help to ensure the efficient adaptation of the proposed model to our robotic task and avoid issues like running out of GPU memory and infinity generation when inference the fine-tuned model.

Our training pipeline involve of the following steps:

- **Dataset Loading**

  we began by loading the prepared dataset of images file name which is anotated with robot action command to train and evaluation data class.

- **Converting Data to Conversation Format**

Most VLMs model are trained to process data as dialogues or message-based interactions so we ned to convert our data to conversation format, this ensure that the model would receive inputs in the same conversational format it was trained on.

- **Loading Model and Tokenizer**

  Instruction tuned models was loaded from HuggingFace's Model Hub. This included both instruct model and model's tokenizer.

- **Applying the Model's Chat Template**

  Chat templates are essential for structuring interactions between language models and users. Properly format the conversations is crucial for getting the best results from proposed model.

- **LoRA Integrating**

  As mentined above to reduce resource intensive training we only fine-tune a small subset parameter of the model. After loading the model we apply LoRA to model using peft library.

- **Setting Hyperparameters**

  Hyperparameters such as learning rate, per device train batch size,gradient accumulation steps, warm up steps, optimizer, number of train steps, and scheduler, weight decay were configured. Mixed-precision training (fp16) was also enabled to further optimize GPU utilization.

- **Training the Model and Monitoring**

  After setting Hyperparameters we train the model which only LoRA adapter parameters so the model don't lose it knowledge from pre-train, metrics like training and validation loss were tracked in real time using Weights & Biases. This enabled close monitoring of overfitting and ensured the model was learning as expected.

- **Saving Model Checkpoints**

  We can set to save model checkpoint at every certain step or we save check point at the last, the model check point will be use to test the model input and output.

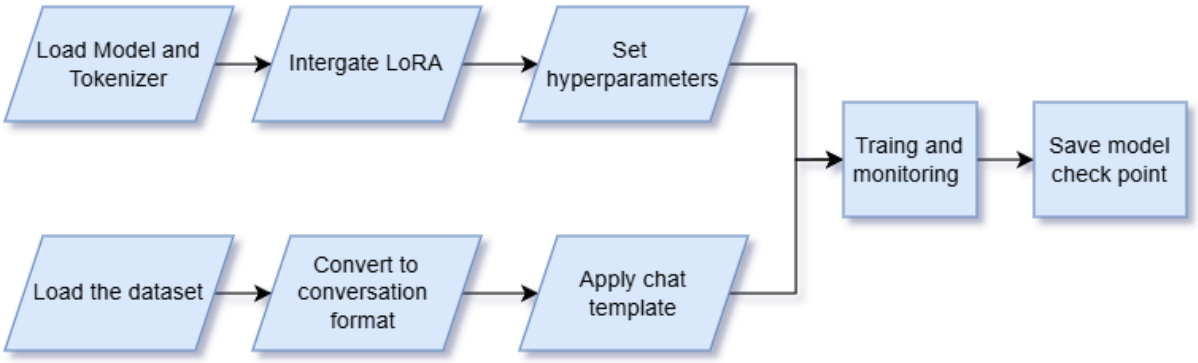


Figure 4.19 Training Pipeline

4.4.3. Memory Usage Considerations

Even when employing LoRA, we sometime encounter issues with GPUs running out of memory during training processes, to address memory challenges, we explored various techniques highlighted in the HuggingFace documentation. These techniques include:

- **Model Quantization**

  Reducing M_{model} by storing the model in lower precision, such as 16, 8, or 4 bits. In our experiment all proposed model are quantized to 4 bits, Reducing the number of bits means the resulting model requires less memory storage, consumes less energy also the model precision.

- **Gradient Accumulation**

  Gradient accumulation is a technique where we can train on bigger batch sizes than our machine would normally be able to fit into memory. This is done by accumulating gradients over several batches, and only stepping the optimizer after a certain number of batches have been performed.

- **Gradient Checkpointing**

  This technique aims to reduce the number of activations saved during training. Given that activations are a significant source of memory usage in transformers, this approach proves highly beneficial. In our we Gradient Checkpointing that is provided by Unsloth.

- **Mixed Precision Training**

  While primarily used to accelerate training, mixed precision training increases memory usage by approximately 50 percent.

4.5. RIEMANNIAN DISTANCE

To evaluate the performance of the fine-tuned vision-language model, we us the **Riemannian Distance**. In our context, where the model outputs structured robot commands (movement distances and rotation angles), the Riemannian Distance provides a principled approach to measure the similarity between predicted and ground-truth commands, especially for rotational components. The formula of **Riemannian Distance** is:

$$d = \sqrt{(x_{\text{pred}} - x_{\text{label}})^2 + (y_{\text{pred}} - y_{\text{label}})^2 + \alpha \cdot (\theta_{\text{pred}} - \theta_{\text{label}})^2} \quad \textbf{(Eq. IV.1.)}$$

Where:

- $x_{\text{pred}}, y_{\text{pred}}, \theta_{\text{pred}}$ are the predicted position and orientation,
- $x_{\text{label}}, y_{\text{label}}, \theta_{\text{label}}$ are the ground truth position and orientation,
- α is a weighting factor that controls the importance of rotation compared to translation. In our implementation we use $\alpha = 1$

In this context, *translation* refers to the robot's movement in the 2D plane specifically the change in its position along the x and y axes. It does not include rotation or orientation.

```python
def riemanian_distance(x_pred, y_pred, t_pred, x_label, y_label, t_label):
    """
    Formula to compute the distance between 2 robot's poses in 2D movement
    """
    alpha =
    d = math.sqrt((x_pred - x_label)**2 + (y_pred - y_label)**2 + alpha * ((t_pred - t_label)**2))
    return d
```

Figure 4.20 Riemannian Distance Implement in Python

Here are some function to calculate Riemannian Distance in robot movement distances and rotation angles context.

Since VLM's prediction is in JSON format, We first need validate the correctness of form of VLM's prediction.

```python
def get_distance_and_angle(cmd: dict):
    polar_coord = [0.0, 0.0]
    distance = float(cmd['params'].get('distance', 0.0))
    angle = cmd['params'].get('angle', '')
    if cmd['action'] == 'move':
        polar_coord[0] += distance
    elif cmd['action'] == 'rotate':
        unit = math.radians(angle)
        direction = cmd['params'].get('is_clockwise', True)
        if direction: polar_coord[1] += unit
        else: polar_coord[1] -= unit
    return polar_coord
```

Figure 4.21 Python Function to Get Distance and Angle

After that, we deduce the polar coordinates corresponding to the prediction. In case that the prediction is not valid, we return the coordination of [0.0, 0.0] which indicate that the robot does not move. After we got the polar coordinates, we updates the robot pose. In 2D movement, the pose of robot is characterised by its position (x, y) and orientation (t) In our setup, either radius (first component) or orientation (second component) is not null. In case, both radius and orientation are not null, we encounter an error.

```python
def validate_and_get_polar_coord(pred_cmd: str, label_cmd: str):
    try:
        dict_pred_cmd = eval(pred_cmd)
        assert isinstance(dict_pred_cmd, dict)
        assert 'action' in dict_pred_cmd
        assert 'params' in dict_pred_cmd
        for k in eval(label_cmd)['params']:
            assert k in dict_pred_cmd['params']
        return get_distance_and_angle(dict_pred_cmd)
    except:
        print("Unexpected error:", sys.exc_info()[0])
        return [0.0, 0.0]
```

Figure 4.22 Python Function to Validate and get Polar Coordinate

```python
def update_robot_pose(x, y, t, polar_coord):
    if polar_coord[0] != 0.0 and polar_coord[1] == 0.0:
        x += polar_coord[0]*math.cos(t)
        y += polar_coord[0]*math.sin(t)
        return x, y, t
    elif polar_coord[0] == 0.0 and polar_coord[1] != 0.0:
        t += polar_coord[1]
        return x, y, t
    elif polar_coord[0] != 0.0 and polar_coord[1] != 0.0:
        print("error", polar_coord)
        return x, y, t
    else:
        return x, y, t
```

Figure 4.23 Robot Pose Update Implement in Python

```python
def update_polar_coord(to_update, for_update):
    return [to_update[0] + for_update[0], to_update[1] + for_update[1]]
```

Figure 4.24 Python Function to Update Polar Coordinate

In this project the evaluation of fine-tuned model performance is conducted as follow:

- **Initialization**
  - Sets the robot's starting pose (x, y, θ) = (0.0, 0.0, 0.0) for each example.
  - Initializes lists to store results.
- **Loop Over Test Dataset**: For each data sample
  - Extract the list of ground-truth command strings.
  - Generate k predictions from the VLM, in our implementation, we use k = 10.
  - Parse and convert each prediction to polar coordinates using `validate_and_get_polar_coord()`.
  - Votes for identical predicted amount polar coordinate sequences.
- **Majority Voting**

  - The most frequently generated sequence (by coordinate signature) is selected as the final prediction.

- **Pose Simulation**
  - Applies each command (predicted and ground truth) sequentially to simulate the robot's updated pose using `update_robot_pose()`.
  - Cumulatively updates polar coordinates using `update_polar_coord()`.
- **Evaluation:**
  - Computes the Riemannian distance between the final predicted and true poses.
  - Records predicted and true distance and rotations.
- **Caculate Error**
  - The error in distance is computed as:

$$\texttt{Distance Error} = \texttt{predicted distance} - \texttt{ground-truth distance}$$

  - The error in rotation is computed as:

$$\texttt{Rotation Error} = \texttt{predicted rotation} - \texttt{ground-truth rotation}$$

Chapter 4

EXPERIMENT CONFIGURATION

5.1. Training Model

As explained in the preceding sections, our primary objective is the development of an VLM optimized for the for autonomous mobile robot. To accomplish this goal, we employ a finetuning approach on the custom robot dataset. The table of our most relevant trials is presented here, and we delve into a comprehensive analysis of these trials in the subsequent sections. In addition to the variable hyperparameters, it is essential to highlight the constants within our experimental setup:

5.1.1. Hardware Specifications

All training procedures are executed on a computational setup comprising NVIDIA GeForce RTX 4080 SUPER GPU with 16 gigabytes of RAM.

Table 5.1. Hardware and Platform Specifications for Model Training

Category	Specification
Platform	
Operating System	Linux
PyTorch Version	2.7.0+cu126
CUDA Version	8.9
CUDA Toolkit	12.6
Hardware	
GPU	NVIDIA GeForce RTX 4080 SUPER
Number of GPUs	1
GPU Memory	16 GB

5.1.2. LoRa Configuration

To evaluate the impact of fine-tuning configuration on model performance, we conducted three separate training trials for each of the two selected models: Qwen2-VL-7B and LLaVA-v1.5-Mistral-7B. Each trial varied in the LoRA-specific hyperparameters, By varying these values

16, 32, and 64 we explored how parameter-efficient fine-tuning impacts the generalization of each model.

Table 5.2. LoRa Parameter Configuration

Model	Trial No.	Num of Parameters	Trainable Parameters	LoRA dim. (r)	LoRA Dropout Rate	No. LoRA Params
Qwen2-VL-7B	1	7B	5.8M	16	0.0	16
Qwen2-VL-7B	2	7B	101.7M	32	0.0	32
Qwen2-VL-7B	3	7B	203.4M	64	0.0	64
llava-V1.6-Mistral-7B	1	7B	49M	16	0.0	16
llava-V1.6-Mistral-7B	2	7B	98M	32	0.0	32
llava-V1.6-Mistral-7B	3	7B	196M	64	0.0	64

5.1.3. Hyperparameters

Based on practical Vision fine-tuning highlight in Unsloth documentation we set hyperparameters Learning rate, per device train batch size,gradient accumulation steps, warm up steps, optimizer, number of train steps, and scheduler, weight decay to all the same across each trial, the value of the configured hyperameters are shown in Table 4.4. The table below summarizes the configurations used in each trial:

Table 5.3. The hyperparameter configuration across different trials

Model	Trial No.	No. step	Training time (hrs)	Batch size	Gradient acc. steps	Gradient checkpointing	Warmup steps	Optimizer	Scheduler	Weight decay	Learning rate
Qwen2-VL	1	200	1.20	2	4	Yes	10	adamw_8bit	Linear	0.01	2e-4
Qwen2-VL	2	200	1.20	2	4	Yes	10	adamw_8bit	Linear	0.01	2e-4
Qwen2-VL	3	200	1.20	2	4	Yes	10	adamw_8bit	Linear	0.01	2e-4
Llava-VL	1	200	2.00	2	4	Yes	10	adamw_8bit	Linear	0.01	2e-4
Llava-VL	2	200	2.00	2	4	Yes	10	adamw_8bit	Linear	0.01	2e-4
Llava-VL	3	200	2.00	2	4	Yes	10	adamw_8bit	Linear	0.01	2e-4

5.1.4. Data Spliting

All models are consistently trained on the exact same dataset, which is the custom robot dataset as described in Section 5.2. The train/test split has also been the exact same for each trial: 80% training data and 20% test data split, generated with a fixed random seed of ’42’.

5.1.5. Finetuning Qwen2-VL

In our pursuit to finetune the Qwen2-VL-7B-Instuct model, we conducted three trials, each comprising 200 max step. The training duration for each of these 200 steps trials was approximately 1.20 hours.

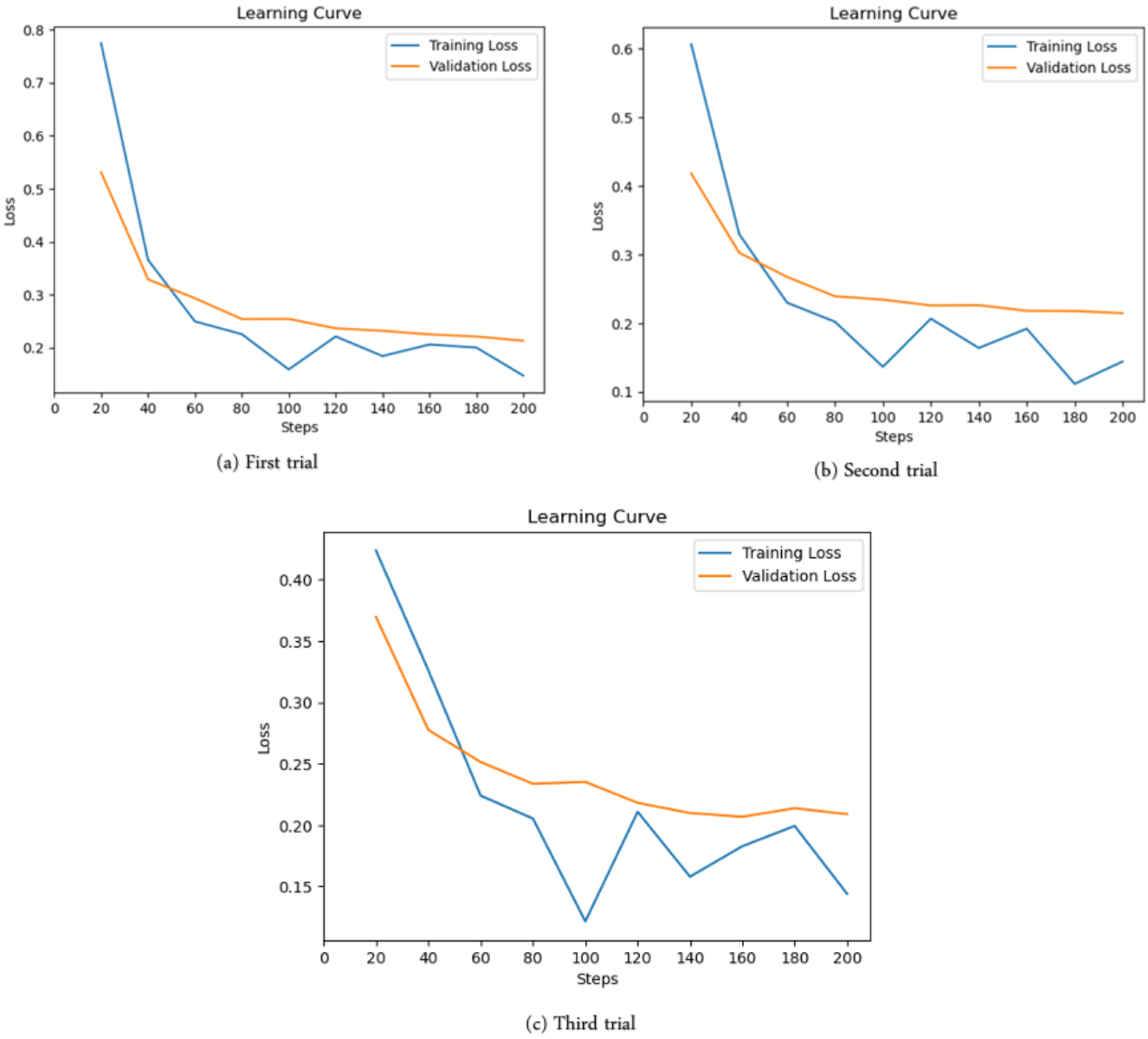


Figure 5.25 Both the training- and validation loss from our 3 Qwen2-VL Finetuning trials.

At the start of the training, the initial loss on the Qwen2-VL model's validation set was measured at 3.310246. The performance of these three trials is summarized as follows:

In figure 5.25a, the results of the first finetuning trial are presented. In this trial, both the training and validation losses decrease steadily and stabilize early in the training process. The training loss is around 0.75 at step 20 and drops to about 0.2 by step 200, while the validation loss follows a similar trend, settling around 0.25. Notably, the gap between training and validation losses remains small throughout training, indicating that the model is not overfitting and is generaliz-

ing well. This suggests that with fewer LoRA parameters, the model has just enough capacity to learn useful patterns from the data without memorizing noise.

In figure 5.25b, the results of the second finetuning trial are presented. This trial shows faster reduction in training loss compared to Trial 1, reaching approximately 0.1 by step 200. The validation loss also improves significantly, dropping to around 0.2. However, there is a more noticeable gap between training and validation losses, especially toward the end of training, which suggests a moderate level of overfitting. Additionally, the training loss curve exhibits slight fluctuations, indicating some instability during training. With more parameters, the model gains more representational power but at the cost of increased risk of overfitting. This trial strikes a reasonable balance between performance and generalization.

In figure 5.25c. In this trial, the model achieves the lowest training loss dropping below 0.15 indicating strong fitting capability due to its high capacity. However, the validation loss does not improve as much, plateauing around 0.2, and the gap between training and validation performance is the largest among all trials. This indicates clear signs of overfitting. Moreover, the training loss becomes increasingly unstable after step 100, suggesting that the model might be sensitive to hyperparameter settings or learning dynamics when using a large number of LoRA parameters. While the model learns complex patterns effectively, its generalization ability suffers, and additional techniques like regularization, early stopping, or reducing parameter count could help improve performance on unseen data.

5.1.6. Finetuning Llava-V1.6-Mistral-7B-HF

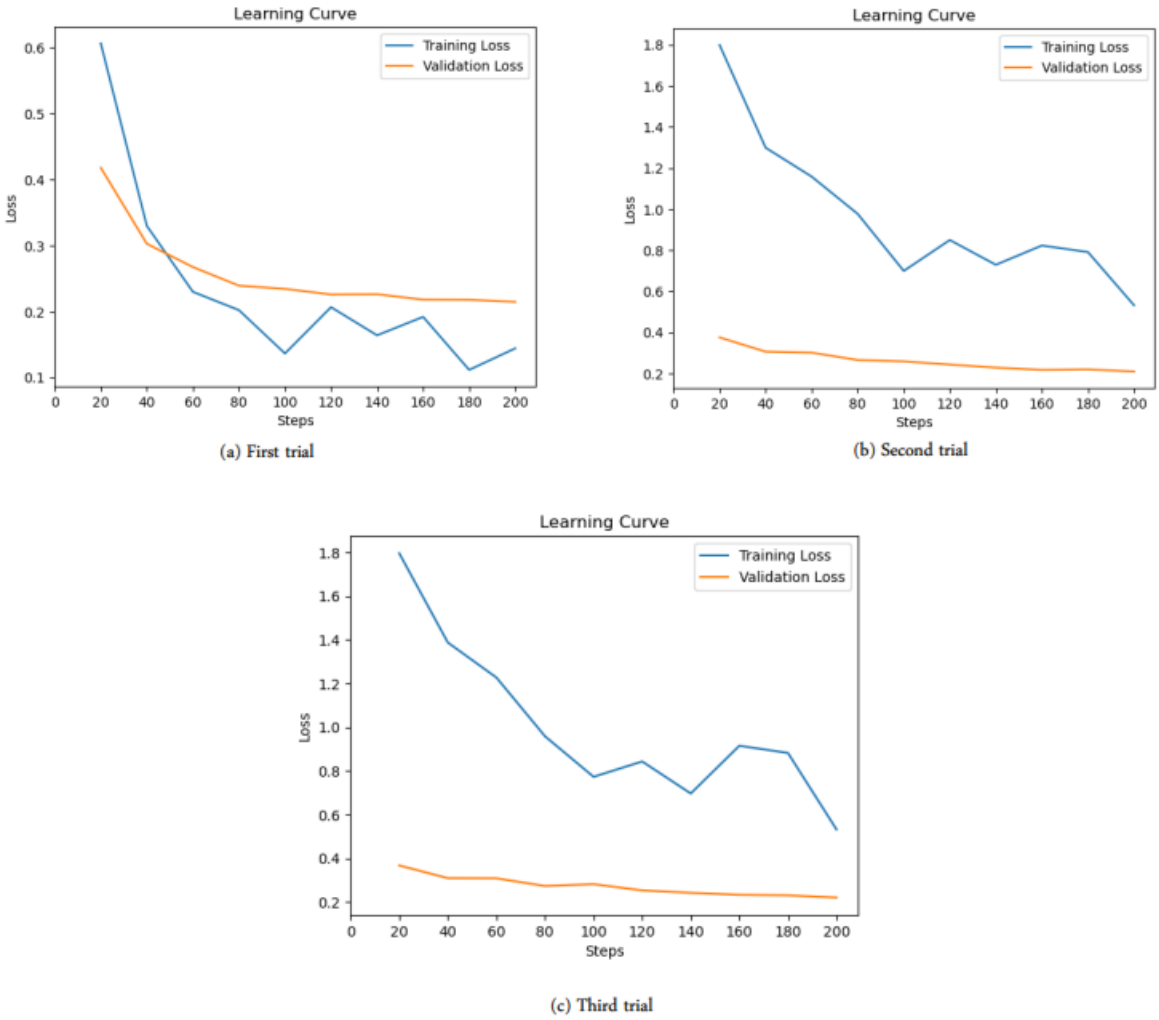


Figure 5.26 Both the training- and validation loss from our 3 Llava-V1.6-Mistral-7B-HF Finetuning trials.

Finetuning Llava V1.6 comprised three trials with each trial differing by number of Lora parameter. The performance of these three trials is summarized as follows:

In figure 5.26a, the results of the first finetuning trial are presented. In this trial, the training loss shows a consistent and relatively smooth decline from the beginning to the final step, indicating effective learning. The validation loss closely tracks the training loss with slight fluctuations but overall follows a decreasing trend as well.

In figure 5.26b, the results of the first finetuning trial are presented. The validation loss is consistently lower than the training loss, which is unusual. Although both losses decrease, the

training loss remains higher across all steps. This suggests strong regularization or training noise affecting only the training phase.

In figure 5.26b, the results of the first finetuning trial are presented. This trial also shows the validation loss lower than training loss throughout. The training loss has large fluctuations after step 100, indicating unstable training.

5.1.7. Summary

Upon reviewing these trials, we note that the loss curves of the Qwen2-VL trials demonstrate smoother progression compared to the Llava-V1.6, given the above insights, Qwen2-VL is the more robust, generalizable model base experimental. Ultimately, the final checkpoint from the first trial of Qwen2-VL was chosen for subsequent testing and deployment due to its stable training process with minimal overfitting.

we acknowledge that further research and resources could lead to models better fitted on the data. Nevertheless, our current trials provide models that sufficiently allow us to assess the effectiveness of our proposed methodology towards achieving our project objectives.

Chapter 5

RESULT AND DISCUSSION

In the chapter we will present the outcome of this project project against the predefined objectives.

6.1. DATASET DEVELOPMENT

We successfully constructed a custom robot specific task dataset, this dataset consisted of 990 images captured from the robot's front-facing camera in a simulated indoor environment. This dataset enabled the model to learns pattern between image and text instruction and corresponding robot command.

P6

	A	B	C
1	image_file_name	instruction	output
2	0.82_from_medkit_87.png	Identify and go to the medical kit.	{"action": "move", "params": {"distance": 0.41}}
3	rotate_right_medkit_72.png	Hunt for the medical kit and move in its direction.	{"action": "rotate", "params": {"angle": 10, "is_clockwise": True}}
4	0.41_from_ball_76.png	Go over to that ball immediately.	{"action": "move", "params": {"distance": 0.0}}
5	no_medkit_20.png	Explore the surroundings for a medical kit.	{"action": "rotate", "params": {"angle": 30, "is_clockwise": False}}
6	0.44_from_medkit_56.png	Search and move toward the medical kit.	{"action": "move", "params": {"distance": 0.0}}
7	rotate_left_25.png	Go pick up the ball (if possible) now.	{"action": "rotate", "params": {"angle": 10, "is_clockwise": False}}
8	2.33_from_ball_137.png	Can you go to the ball immediately.	{"action": "move", "params": {"distance": 1.92}}
9	rotate_left_103.png	Head over to that ball for me when it's safe.	{"action": "rotate", "params": {"angle": 10, "is_clockwise": False}}
10	rotate_left_trash_bin_12.png	Find the closest trash discard point and go there.	{"action": "rotate", "params": {"angle": 10, "is_clockwise": False}}
11	rotate_right_medkit_30.png	If you see a medical kit, approach it.	{"action": "rotate", "params": {"angle": 10, "is_clockwise": True}}
12	rotate_right_49.png	Roll to the ball when it's safe.	{"action": "rotate", "params": {"angle": 10, "is_clockwise": True}}
13	2.40_from_trash_bin_25.png	Go to the closest trash disposal station.	{"action": "move", "params": {"distance": 1.99}}
14	no_trash_bin_38.jpg	Drive to the closest waste disposal unit.	{"action": "rotate", "params": {"angle": 30, "is_clockwise": False}}
15	2.31_from_trash_bin_24.png	Proceed toward the nearest trash collection bin.	{"action": "move", "params": {"distance": 1.9}}
16	rotate_right_medkit_43.png	Move until the medical kit is detected.	{"action": "rotate", "params": {"angle": 10, "is_clockwise": True}}
17	0.43_from_medkit_46.png	Search for a medical kit nearby.	{"action": "move", "params": {"distance": 0.0}}
18	rotate_right_13.png	Approach the ball immediately.	{"action": "rotate", "params": {"angle": 10, "is_clockwise": True}}
19	2.68_from_trash_bin_151.png	Go toward the closest trash disposal station.	{"action": "move", "params": {"distance": 2.27}}
20	0.95_from_medkit_37.png	Look for a medical kit and go to it.	{"action": "move", "params": {"distance": 0.54}}
21	0.65_from_trash_bin_5.png	Go to the closest trash can.	{"action": "move", "params": {"distance": 0.24}}
22	1.63_from_trash_bin_137.png	Move toward the closest trash disposal unit.	{"action": "move", "params": {"distance": 1.22}}
23	1.82_from_medkit_122.png	Explore until the medical kit is in view.	{"action": "move", "params": {"distance": 1.41}}
24	rotate_left_trash_bin_49.png	Go toward the closest trash repository.	{"action": "rotate", "params": {"angle": 10, "is_clockwise": False}}
25	0.46_from_medkit_73.png	Explore the surroundings for a medical kit.	{"action": "move", "params": {"distance": 0.05}}
26	0.41_from_trash_bin_60.png	Navigate towards the closest garbage can.	{"action": "move", "params": {"distance": 0.0}}

Figure 6.27 Dataset in CSV format

6.2. MODEL ADAPTATION

We Successfully fine-tuned pre-trained VLM model (**Qwen2-VL-7B-Instruct**) that can translates image from robot front-facing camera and text instructions into accurate and context-aware commands robot commands.

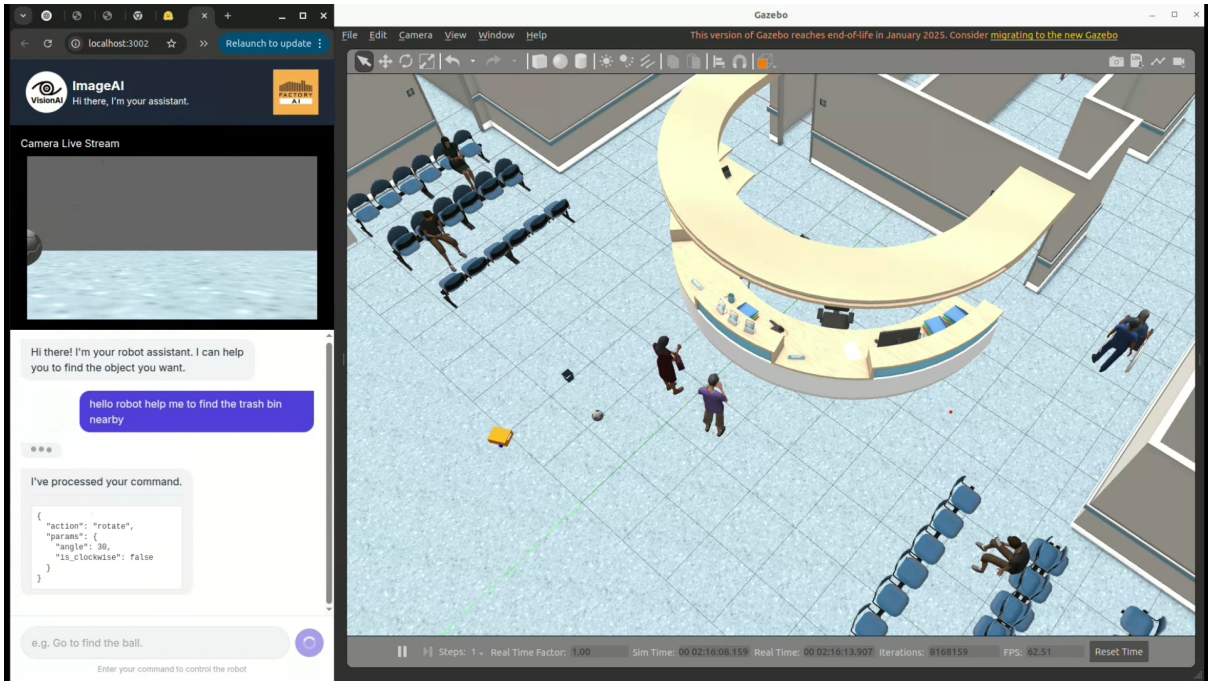


Figure 6.28 Examples of Model Input and Predicted Robot Command Output

6.3. AUTONOMOUS NAVIGATION

A simple autonomous navigation policy was designed to allow the robot to search, detect, and approach specific target objects, in our experiment the object are soccer ball, medical kit, trash bin, using model predictions. The robot first rotates to explore the scene until the target object is detected. Once detected, the robot aligns itself with the object and moves forward to approach it.

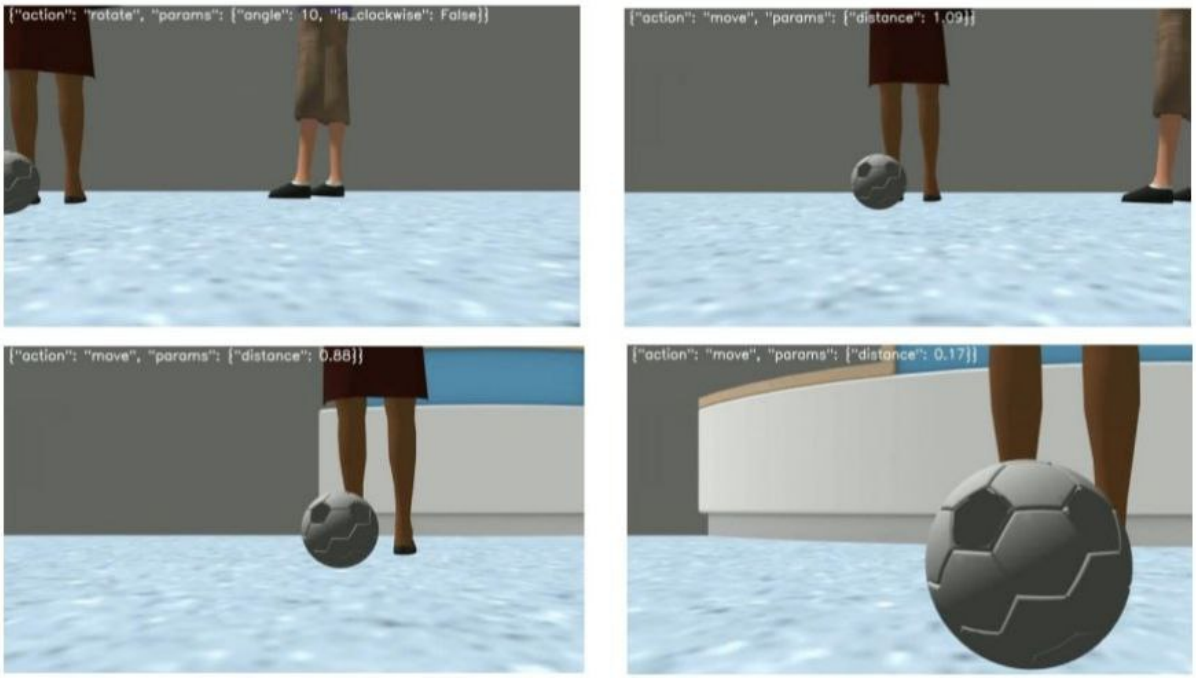


Figure 6.29 Sequence of Robot Navigation Steps During Object Search and Approach

6.4. MODEL EVALUATION

The model performance was evaluated by uising Riemannian, since our robot can do only two action rotate, and move forward. So this evaluation focus on two things, which are distance error and rotation error. A histogram was plotted to visualize the distribution of distance errors across the test set.

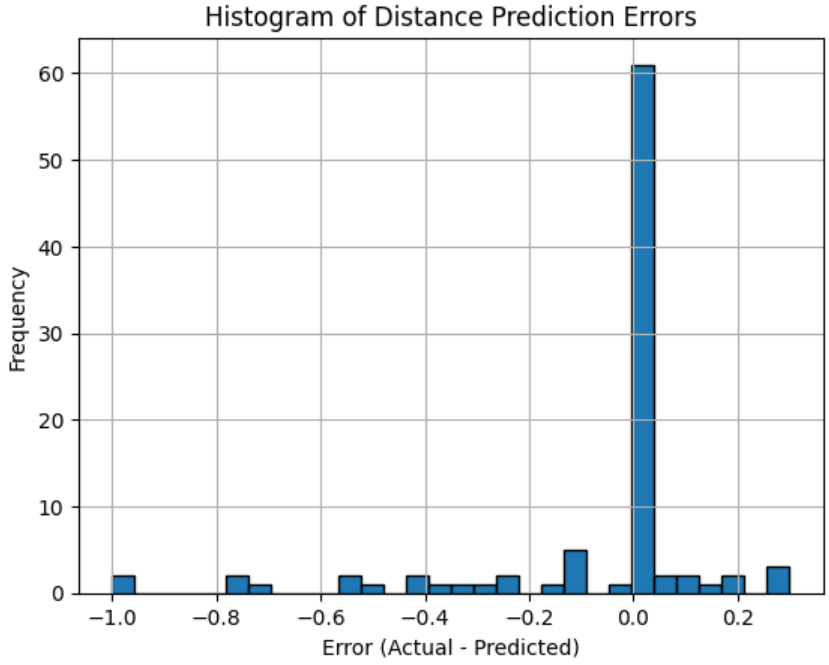


Figure 6.30 Histogram of Distance Prediction Errors

Similarly, the rotation errors were computed using the Riemannian Distance, reflecting the angular deviation between the predicted and true orientations.

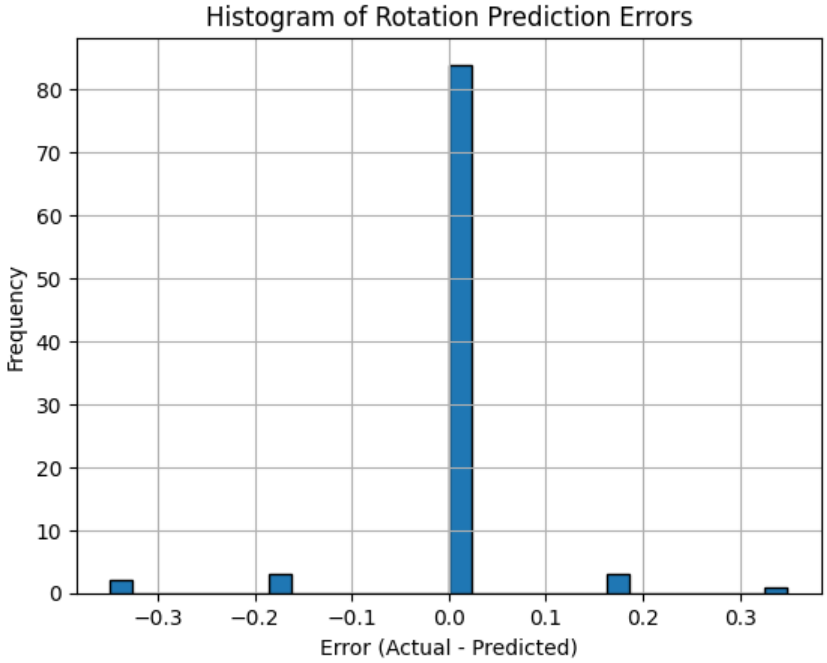


Figure 6.31 Histogram of Rotation Prediction Errors

based on the histogram, the model demonstrates high reliability in rotation control, which is essential for object alignment and exploration. In the other hand distance prediction, is slightly noisier and could be improved with further research.

6.5. PROTOTYPE AND MODEL INTEGRATION

The integration of the model and prototype in this project involves the following steps.

- **User Interface:**
  - A **custom-built web interface** allows real-time interaction.
  - Users can input **high-level natural language commands** (e.g., "go to the red ball").
  - A **live video stream** from the robot's front-facing camera provides real-time visual feedback.
- **Backend Server (Flask):**
  - Upon receiving a user command, the Flask backend:
    * **Captures** the current camera frame from the simulation.
    * **Combines** the image with the user's text prompt.
- **Vision-Language Model Inference:**
  - The combined input (image + instruction) is passed to a **fine-tuned Vision-Language**

Model (VLM).

- The VLM generates a **structured JSON command**
- The output represents a high-level, executable command grounded in visual and linguistic input.

- **Robot Control Execution:**
  - The JSON command is parsed to extract control parameters (e.g., rotation angle, movement distance).
  - These parameters are sent to a **ROS2 control node**, which executes the corresponding action in Gazebo.
- **Real-Time Monitoring and Feedback:**
  - Users observe the robot's behavior via the **Gazebo interface** and the **web interface**.
  - This forms a **closed instruction-perception-action loop**:
    1. User provides a high-level instruction
    2. System captures visual input
    3. VLM interprets and generates a command
    4. Robot acts and user monitors the result

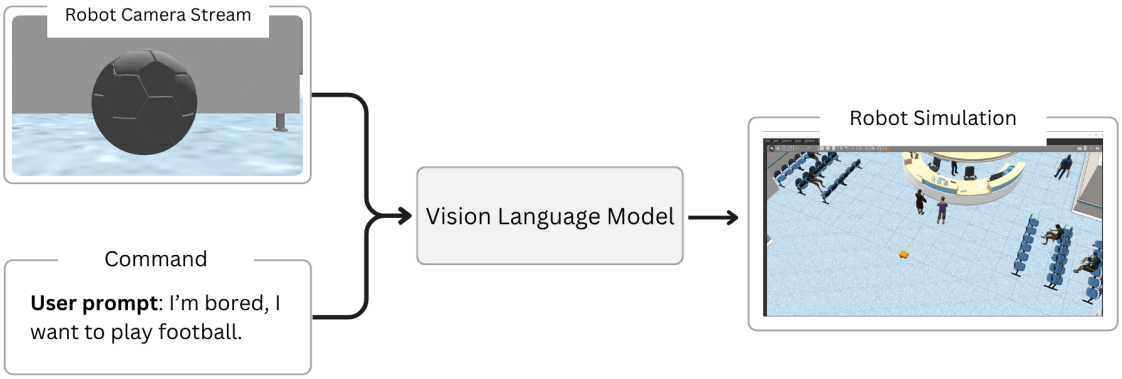


Figure 6.32 System workflow

Chapter 5

CONCLUSION AND FUTURE WORK

In this study, we successfully demonstrated the fine-tuning of the Qwen2-VL-7B vision-language model (VLM) for autonomous mobile robot control. By leveraging the capabilities of VLMs, we aimed to bridge the gap between visual input and natural language instructions, enabling a more intuitive and efficient interaction with the robot. The integration of both vision and language processing allows the robot to perform tasks tasks such as object detection, navigation, and movement commands based solely on visual cues and textual instructions.

Our methodology included the use of advanced fine-tuning techniques, such as LoRA, to efficiently adapt the pre-trained Qwen2-VL-7B model to our specific task. This approach proved to be effective in maintaining the model's performance while significantly reducing the computational cost and memory requirements, a crucial factor for practical robotic applications.

The dataset we used for fine-tuning was carefully curated to simulate real-world scenarios, where the robot must autonomously detect and move towards an object based on visual input. By exposing the model to various situations, such as tracking objects in dynamic environments, we ensured that the robot could handle a wide range of real-time challenges. The model's ability to understand and act on natural language instructions, in combination with its vision capabilities, significantly enhances the robot's autonomy.

Furthermore, our evaluation showed that the fine-tuned model is capable of executing a diverse set of actions, such as rotating and moving towards objects, based on simple language commands. These results highlight the potential of VLMs in robotic control systems, making robots more adaptable and efficient in performing tasks without the need for complex programming or manual input.

In conclusion, this project has demonstrated the feasibility and effectiveness of fine-tuning VLMs for autonomous mobile robot control. The successful integration of vision and language models into the robotic framework opens up numerous possibilities for future developments in autonomous systems. As we continue to explore and refine these models, we foresee even more advanced applications, such as multi-modal interactions, real-time decision-making, and the ability to work in highly dynamic environments. The findings of this research contribute to the growing body of knowledge in the field of robotics and artificial intelligence,

offering a promising direction for the development of intelligent, autonomous systems capable of operating in complex, real-world settings.

Future work could be improve model performance, particularly in precise distance estimation by increase sample of dataset, to enhance accuracy and robustness, explore the potential to transition the model from simulation to practical robotic applications.

REFERENCES

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, et al. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control, arXiv preprint arXiv:2307.15818, 2023. https://robotics-transformer2.github.io

[2] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution. arXiv preprint arXiv:2409.12191, 2024. Available at: https://github.com/QwenLM/Qwen2-VL

[3] Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, Hang Li, and Tao Kong. Vision-Language Foundation Models as Effective Robot Imitators. Preprint, arXiv:2311.01378, 2023. https://arxiv.org/abs/2311.01378

[4] Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LORA: Low-Rank Adaptation of Large Language Models . Preprint, arXiv:2106.09685, 2021. https://arxiv.org/abs/2106.09685

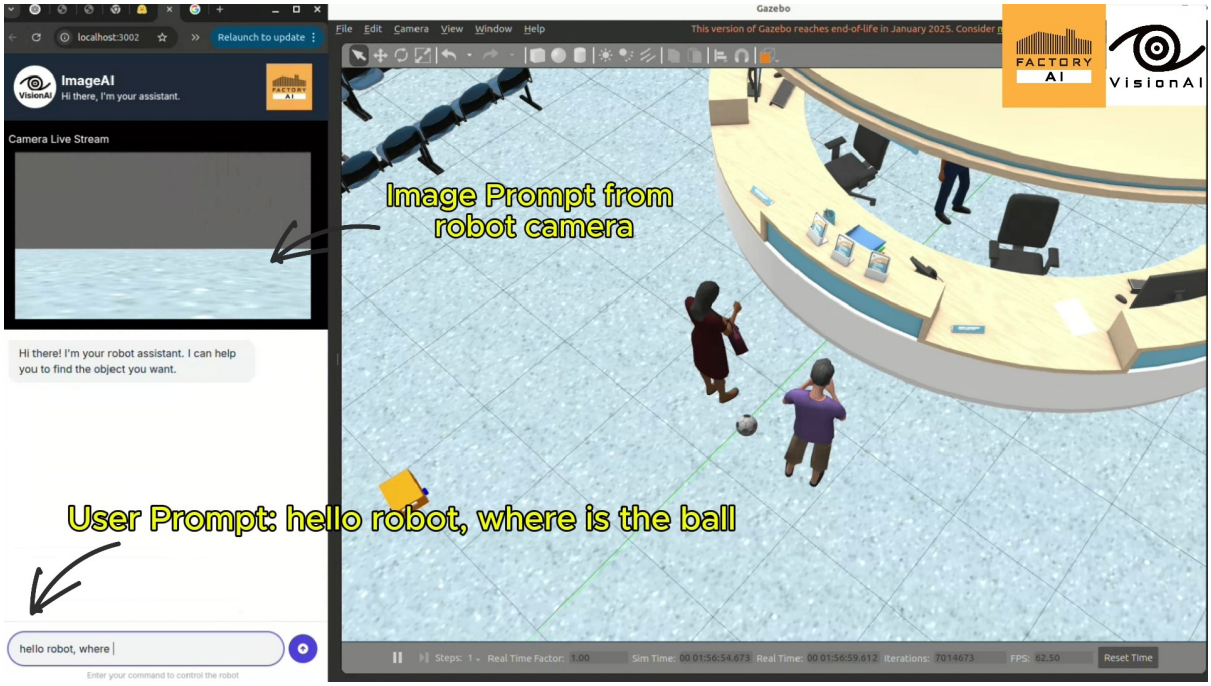
[5] Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Fine-Tuning Large Language Models: A Practical Guide to Adapting Pre-Trained Language Models for Specific Tasks . Technical Report, 2024.

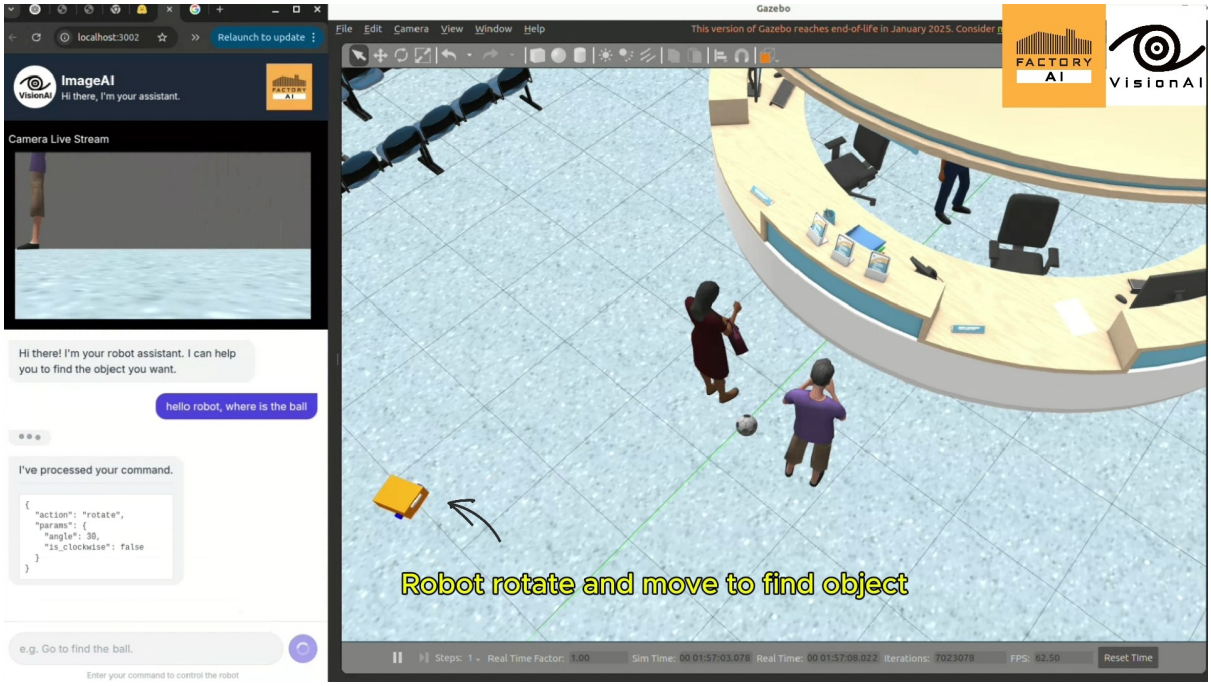
[6] Medium article: "Unsloth.ai: Revolutionizing AI Model FineTuning". https://medium.com/aimonks/unsloth-ai-revolutionizing-ai-model-fine-tuning-579c0afbddce

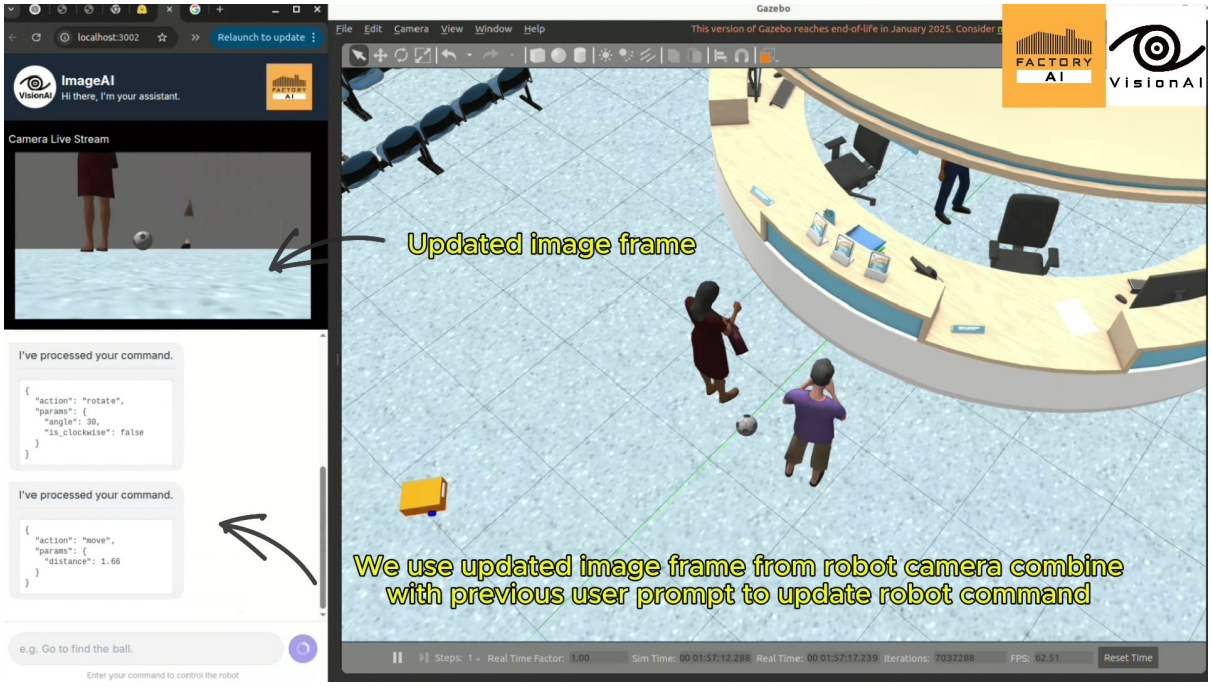
APPENDICES

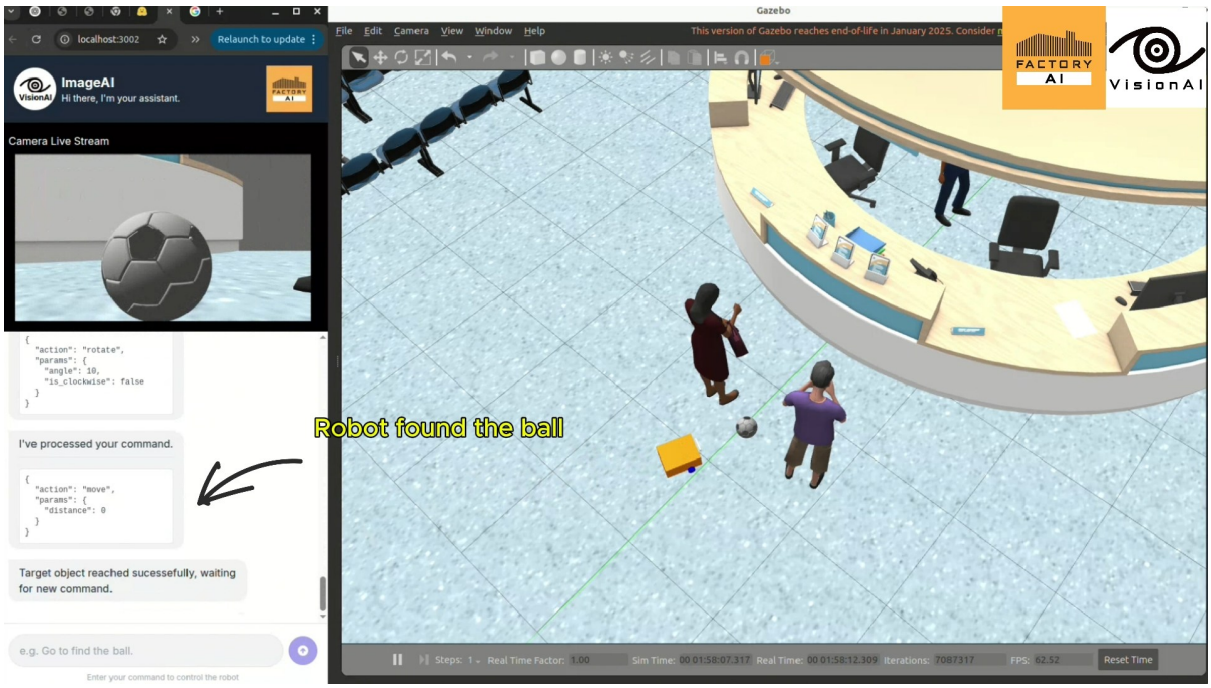
Robot Navigation and Object Interaction in Simulation Environment.

- One object search.

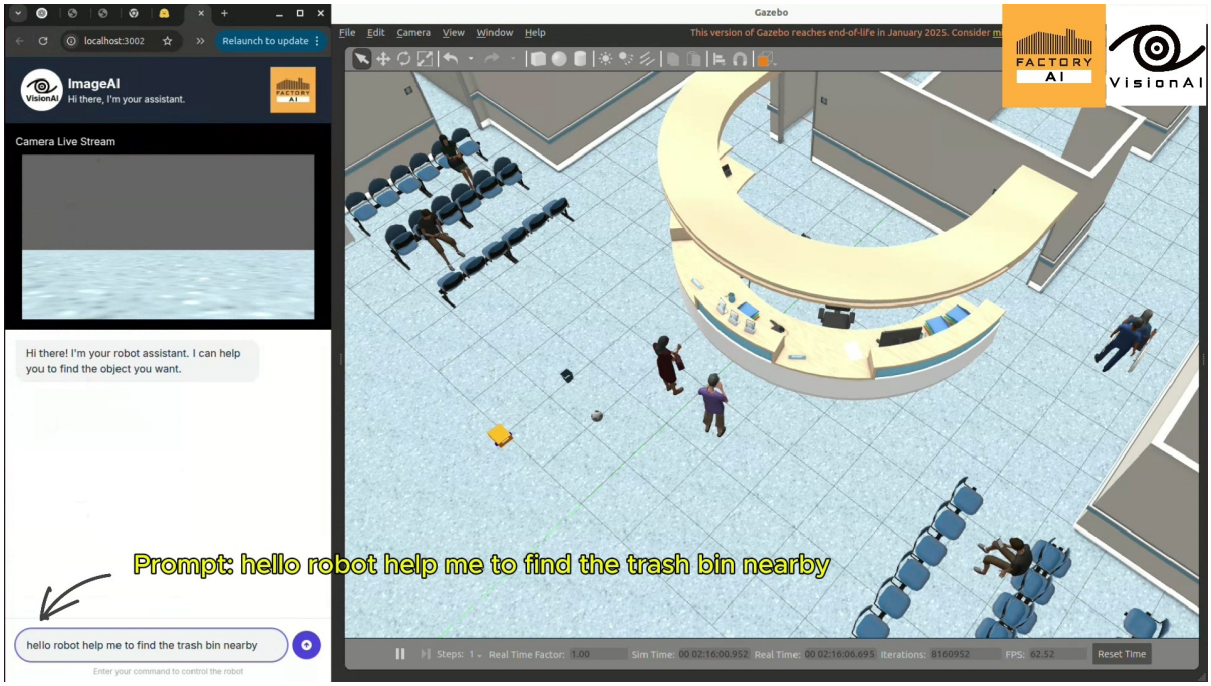


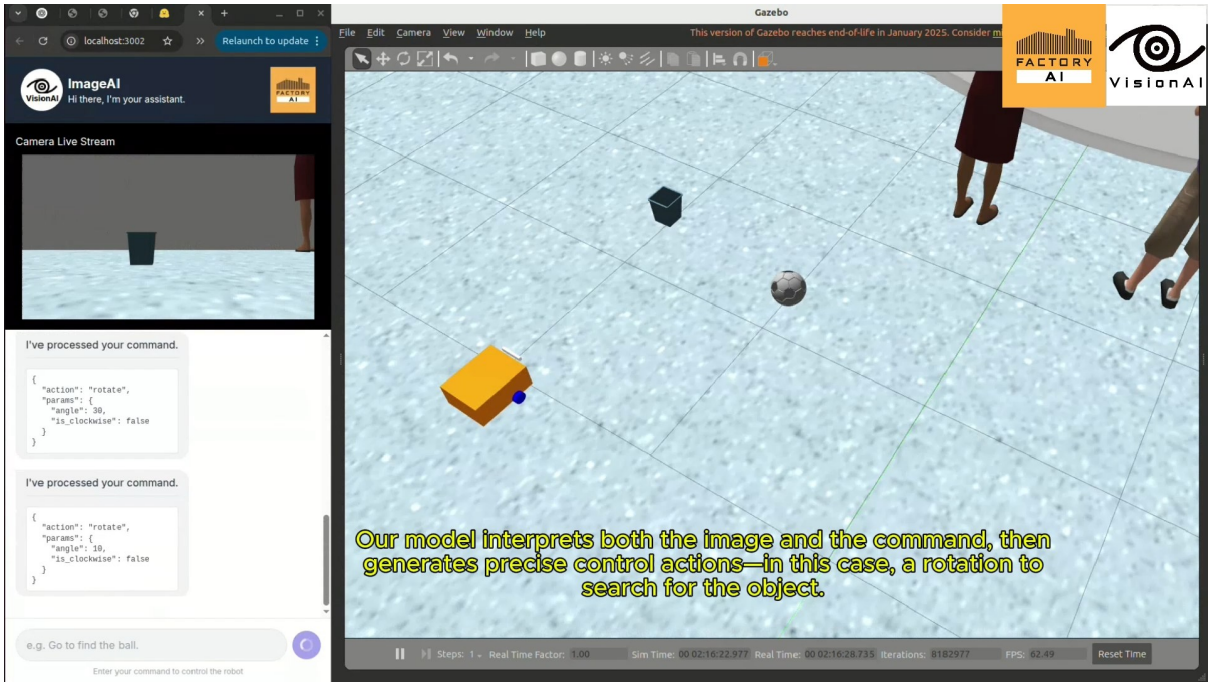


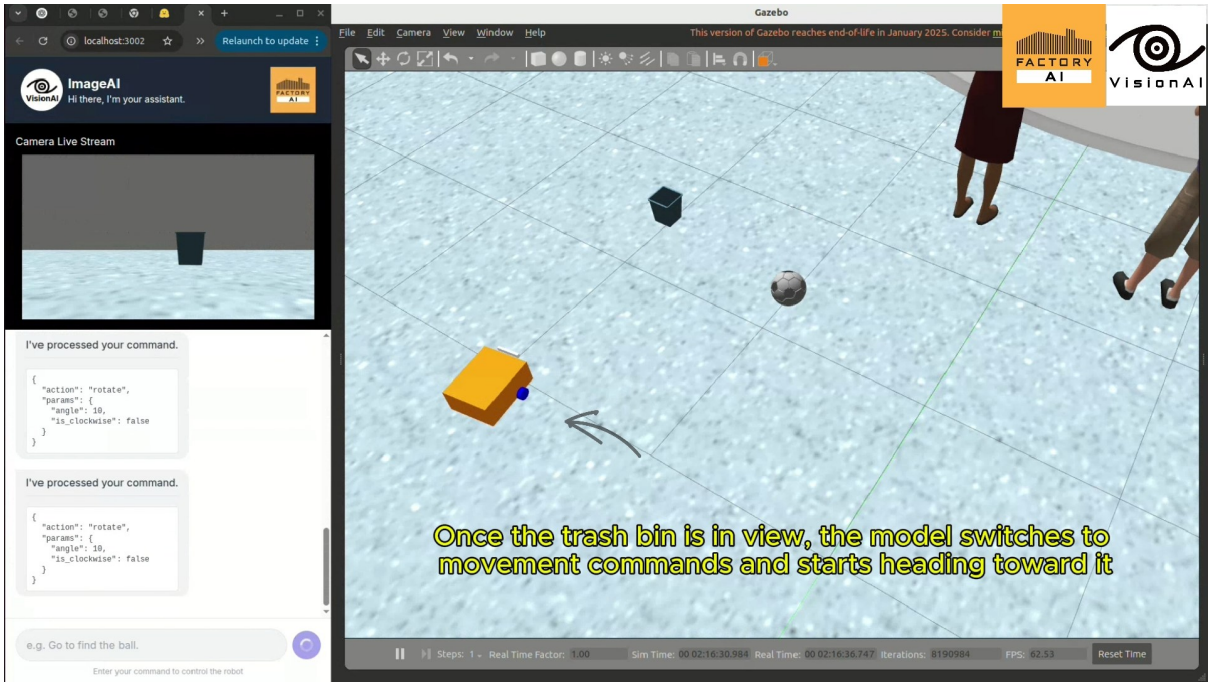


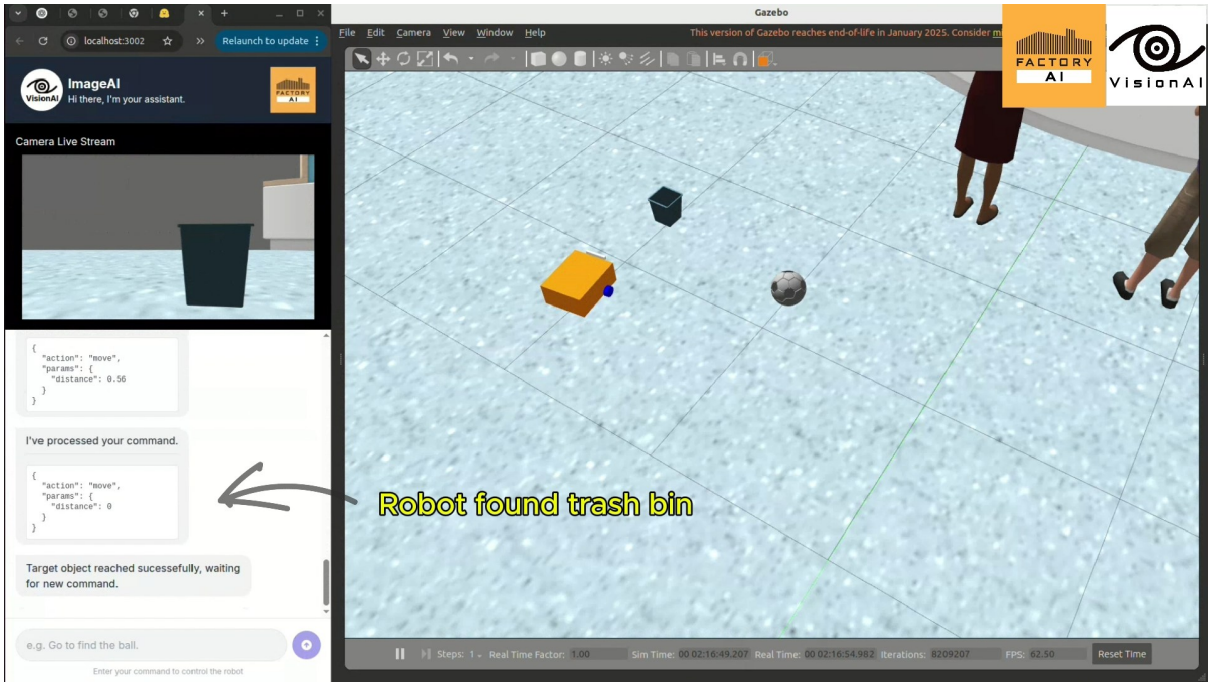


- Two object search.


localhost:3002
Relaunch to update
ImageAI
Hi there, I'm your assistant.
Camera Live Stream
Hi there! I'm your robot assistant. I can help you to find the object you want.
hello robot help me to find the trash bin nearby
Enter your command to control the robot
Gazebo
File Edit Camera View Window Help
This version of Gazebo reaches end-of-life in January 2025. Consider m
FACTORY AI
VisionAI
Steps: 1 Real Time Factor: 1.00 Sim Time: 00 02:16:00.952 Real Time: 00 02:16:06.695 Iterations: 8160952 FPS: 62.52
Reset Time
Prompt: hello robot help me to find the trash bin nearby


localhost:3002
Relaunch to update
ImageAI
Hi there, I'm your assistant.
Camera Live Stream
I've processed your command.
{
"action": "rotate",
"params": {
"angle": 30,
"is_clockwise": false
}
}
I've processed your command.
{
"action": "rotate",
"params": {
"angle": 10,
"is_clockwise": false
}
}
e.g. Go to find the ball.
Enter your command to control the robot
Gazebo
File Edit Camera View Window Help
This version of Gazebo reaches end-of-life in January 2025. Consider m
FACTORY AI
VisionAI
Steps: 1 Real Time Factor: 1.00 Sim Time: 00 02:16:22.977 Real Time: 00 02:16:28.735 Iterations: 8182977 FPS: 62.49
Reset Time
Our model interprets both the image and the command, then generates precise control actions—in this case, a rotation to search for the object.


localhost:3002
Relaunch to update
ImageAI
Hi there, I'm your assistant.
Camera Live Stream
I've processed your command.
{
"action": "rotate",
"params": {
"angle": 10,
"is_clockwise": false
}
}
I've processed your command.
{
"action": "rotate",
"params": {
"angle": 10,
"is_clockwise": false
}
}
e.g. Go to find the ball.
Enter your command to control the robot
Gazebo
File Edit Camera View Window Help
This version of Gazebo reaches end-of-life in January 2025. Consider m
FACTORY AI
VisionAI
Once the trash bin is in view, the model switches to movement commands and starts heading toward it
Steps: 1 Real Time Factor: 1.00 Sim Time: 00 02:16:30.984 Real Time: 00 02:16:36.747 Iterations: 8190984 FPS: 62.53
Reset Time


localhost:3002
Relaunch to update
ImageAI
Hi there, I'm your assistant.
Camera Live Stream
{
"action": "move",
"params": {
"distance": 0.56
}
}
I've processed your command.
{
"action": "move",
"params": {
"distance": 0
}
}
Target object reached sucessefully, waiting for new command.
e.g. Go to find the ball.
Enter your command to control the robot
Gazebo
File Edit Camera View Window Help
This version of Gazebo reaches end-of-life in January 2025. Consider m
FACTORY AI
VisionAI
Robot found trash bin
Steps: 1 Real Time Factor: 1.00 Sim Time: 00 02:16:49.207 Real Time: 00 02:16:54.982 Iterations: 8209207 FPS: 62.50
Reset Time