Code Generation With Robotics-Oriented Vision Language Models for Robot Arms Applications.

Tran Quang Minh, Luu Trong Hieu, Nguyen Cong Khanh, Nguyen Quang Trung
Instructor - Mr.Tran Trong Toan
FPT University — VietDynamic JSC
Ho Chi Minh City, Vietnam
Emails: quantran102005@gmail.com, Luutronghieu0709@gmail.com, congkhanhtruongthi@gmail.com, trungnqse183108@fpt.edu.vn

***Abstract*—This report presents the outcome of our internship project of our team, who worked on a project entitled Code Generation with Large Language Models for Robot Arms Applications. The work focuses on applying Robotics-Oriented Vision Language Models (VLMs) to automatically generate and adapt control code for industrial robot arms. Throughout the project, the team implemented a system that integrates VLM-based code generation, task-specific prompt design, and real-time code validation for robotic motion control. The report discusses the design process, implementation details, encountered challenges, and experimental evaluations of the proposed approach. Results demonstrate that VLMs can significantly accelerate robot programming workflows while maintaining adaptability to various robot configurations. The source code and implementation details are publicly available at: GitHub/Code-gen-for-robot-arm-OJT-FALL-2025-FPT**

I. Introduction

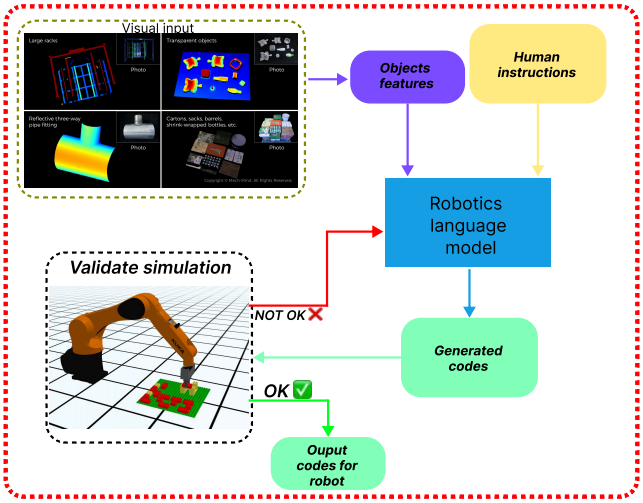
The rapid advancement of artificial intelligence (AI) and machine learning has led to the development of vision language models (VLMs) that can understand and generate human-like text. These models have shown remarkable capabilities in various natural language processing tasks, including code generation. The ability to generate code automatically has significant implications for the software development industry, particularly in specialized fields such as robotics. During our internship at VietDynamic JSC, we had the opportunity to work on a project focused on leveraging VLMs for code generation in robot arm applications. The project aimed to explore how VLMs can assist in automating the coding process, thereby improving efficiency and reducing the time required for software development in robotics. This report provides a comprehensive overview of our internship experience, including the tasks we undertook, the challenges we encountered, and the skills we developed. We also discuss the potential applications of VLMs in the robotics industry and their impact on future developments.

II. Related Work

Recent advancements in large language models (VLMs) have demonstrated their potential in various applications, including code generation for robotic systems. Notable works in this domain include Mu et al.'s RoboCodeX[1], which explores the use of VLMs to generate code for robotic tasks, showcasing the ability of these models to understand and execute complex instructions. Another significant contribution is the Robotic Programmer by Xie et al.[2], which focuses on video-instructed policy code generation for robotic manipulation, highlighting the integration of visual inputs with VLMs to enhance robotic capabilities. These studies underscore the transformative potential of VLMs in automating and optimizing code generation for robotics, paving the way for more efficient and intelligent robotic systems. Other relevant works include the development of Large-language models (VLMs) like MobileVLM[3], which are designed to handle multimodal inputs, making them suitable for applications that require both visual and textual understanding. The integration of VLMs in robotics can significantly enhance the interaction between robots and their environments, enabling more sophisticated and context-aware behaviors.

III. Methodology

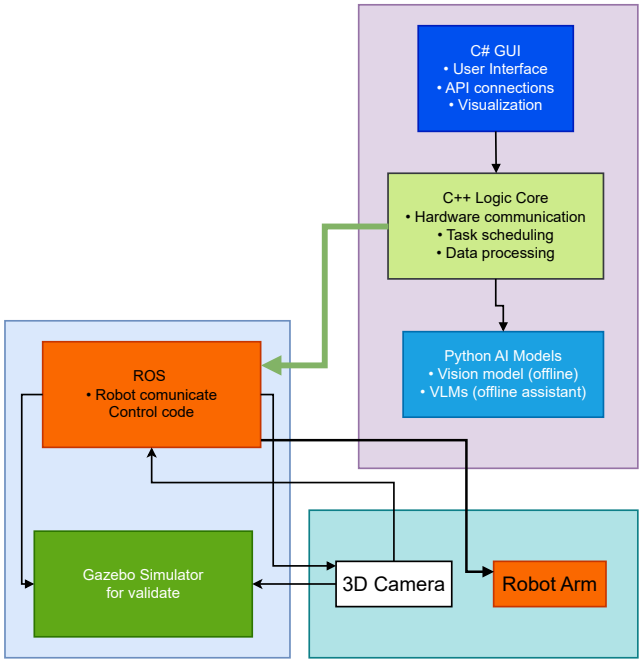
A. Overview



We leverage the capabilities of vision language models (VLMs) to generate code for robot arm applications. Combined with visual information from dedicated sensors like Mech-EYE we could enhance the understanding of the environment and improve the accuracy of the generated code. The overall pipeline consists of several key components: Intel Realsense 3D camera, VLMs, and a simulator for validating the generated code without the need of physical hardware.

The Intel Realsense camera captures high-resolution images and 3D point clouds of the robot's surroundings, providing essential visual context for the VLMs. The VLMs, such as MobileVLM[3], RoboCodeX[1] or specialized models for robotic manipulation like GeminiRobotics[4], are then employed to generate code based on the visual data and specific task requirements. More comprehensive tasks such as video instructions for robotic manipulation are also considered [2]. Finally, the generated code is executed in a controlled environment built with ROS2 and GAZEBO to validate its functionality and performance.

B. Systems Design



We designed a modular system architecture to facilitate the integration of various components involved in the code generation process. The architecture consists of the following modules:

- **GUI Module:** This module is responsible for user interaction. It allows users to input task specifications, view visual data from the Mech-EYE or Intel Realsense cameras, and monitor the code generation process. user can also select the VLM they want to use for code generation, do work with camera like calibration, capturing image, point cloud, etc.
- **Logic Core Module:** This module manages the communication with the various system components, ensuring smooth data flow and coordination between modules. It also handles all Logic task like communication with hardwares, processing user inputs, data, etc.
- **VLM and Vision Integration Module:** This module interfaces with the selected vision language models and vision models. It handles the input of visual data and task specifications, and it retrieves the generated code from the models.
- **Simulate and Validate Module:** This module let user simulate and assess the performance of the generated code based on predefined metrics such as accuracy, efficiency, and robustness. It provides feedback for further refinement of the VLMs generated code and the overall system.

C. Robotics-Oriented Vision Language Models

1) Vision Language Models: The robotics team is responsible for the integration and fine-tuning of the large language models (VLMs) used for code generation. They ensure that the selected VLMs are properly configured to handle the specific requirements of robot arm applications. The team also works on optimizing the performance of the VLMs by fine-tuning them on relevant datasets and adjusting hyperparameters to improve code generation accuracy. Initially, we planned to use MobileVLMV2 [5] as the base model for our code generation tasks due to its efficiency and effectiveness in handling multimodal inputs. The model was fine-tuned on a custom dataset of robotic tasks to enhance its performance in generating context-aware code. It was also integrated into the system to provide real-time code generation capabilities, allowing for dynamic adaptation to changing task requirements and environmental conditions. However, due to the severe shortage of operational robot data necessary for fine-tuning, adapting the model effectively became infeasible. The scarcity of domain-specific datasets for robot control and manipulation tasks limited the model's ability to generalize and perform robust code generation for complex robotic operations. Consequently, after encountering these limitations we transitioned to utilizing Gemini Robotics. Gemini Robotics is an advanced Vision-Language-Action model designed specifically for robotics applications, with strong embodied reasoning and 3D spatial understanding. This model provides more accurate, reactive, and dexterous control code generation suited for real-world robotic systems, and can adapt flexibly to a wide range of manipulation tasks and robotic platforms.[6], [4] Recent

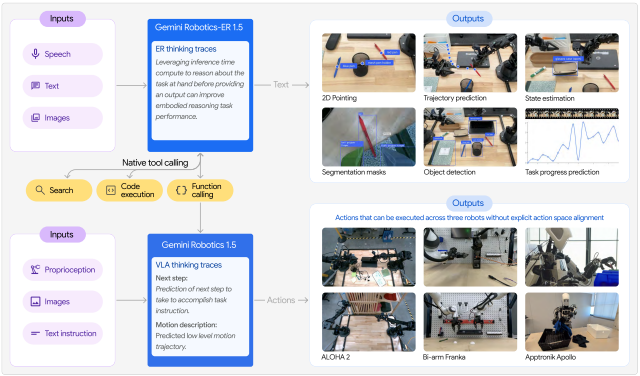


Fig. 1: Overview of Gemini Robotics 1.5 abilities[4]. The model integrates visual inputs with language understanding to generate action plans for robotic manipulation tasks.

developments in Gemini Robotics 2.0[4] have further pushed the boundaries of generalist robots by incorporating advanced

embodied reasoning, thinking, and motion transfer capabilities. This iteration enhances the model's ability to understand complex environments and execute sophisticated tasks, making it a promising choice for our code generation needs in robot arm applications. But the current available version Gemini Robotics 1.5 have the abilities to process visual inputs it self but it is limited to 2D image inputs only, which may not provide sufficient spatial context for accurate code generation in complex robotic tasks. By integrating a dedicated vision model that can capture and interpret 3D spatial information from the environment, we can enhance the overall performance of the Gemini Robotics model. This integration will allow the system to better understand the spatial relationships and dynamics of objects within the robot's workspace, leading to more precise and effective code generation for robotic manipulation tasks. Though the model come with self contained vision capabilities, a dedicated local 3D vision model is necessary when it meight reduce latency and cost associated with cloud-based processing. Additionally, a specialized vision model can be fine-tuned to the specific environmental conditions and object types encountered in our robot arm applications, thereby improving the accuracy and reliability of visual perception. This tailored approach allows for better integration with the Gemini Robotics model, ultimately enhancing the overall system's performance in generating context-aware and effective code for robotic tasks.

2) Robotics Code Synthesis: Based on the visual inputs and task specifications provided by the user, We utilize the Gemini Robotics model to generate code that control the robot arm's movements and actions. The generated code is then subjected to a verification process to ensure its correctness and reliability. The verification process involves the output code of the Main generation module is passed to a validation module that take the generated code and the original task specifications as inputs. This module will analyze the code to check for logical consistency, adherence to task requirements, and potential errors. If any discrepancies or issues are identified, the module will provide feedback to the generation module for refinement and improvement. This iterative process in 2 continues until the generated code meets the desired standards of accuracy and reliability, ensuring that the final output is suitable for deployment in real-world robotic applications.

D. Vision Models

Camera team is responsible for the integration and calibration of the Intel Realsense cameras. They ensure that the visual data captured by these cameras is accurately processed and made available for the VLMs. Camera team also works on optimizing the camera settings to enhance image quality and depth perception, which are crucial for effective code generation. The main tasks of camera team is to develop a custom Vision model that can accurately detect and recognize objects in the robot's environment and provide relevant visual information to support the VLMs process. We plan to use YOLOv11[7] as the base model for our vision tasks due to its balance between speed and accuracy. The model will be fine-tuned on our custom dataset of industrial objects to improve its performance in the specific context of robot arm applications. The vision model will be integrated into the system to provide real-time object detection and recognition capabilities, which are essential for generating accurate and context-aware code.

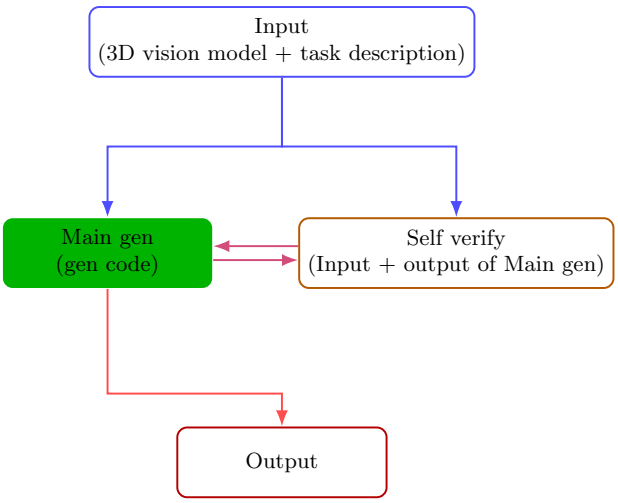


Fig. 2: Code synthesis process using VLMs with self-verification mechanism. The system generates code based on visual inputs and task specifications, followed by a verification step to ensure correctness and reliability.

However, we later found out that the YOLOv11 model was too large and computationally intensive for our limited hardware resources. Then we decided to switch to a custom lightweight model based on MobileNetV2[8] architecture. This model was an experimental model that our team leader developed when he worked at FPT University AiTA-lab[9], which was optimized for speed and efficiency while still maintaining a reasonable level of accuracy.

1) Classification Experiment: We conducted an experiment to evaluate the performance of our custom MobileNetV2-based model on a classification task. The experiment involved training the model on a dataset of industrial objects and CIFAR10[10] dataset to assess its accuracy and efficiency.

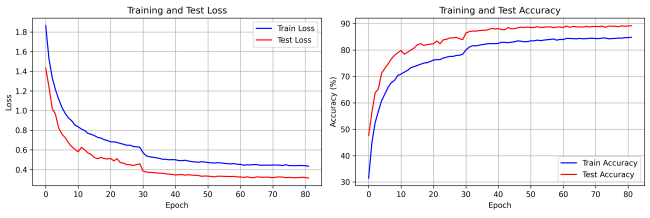


Fig. 3: Training curves of our MobileNetV2-based model on CIFAR10 datasets. The model demonstrates competitive performance while maintaining efficient training dynamics.

Train and test result on CIFAR10 dataset in figure 3 shows the training curves of our MobileNetV2-based model on the CIFAR10 dataset. This model have 0.34M parameters and 0.06

GFLOPs, which is significantly smaller and more efficient than many state-of-the-art models while it manintains competitive performance. The small size and low computational requirements of our model make it suitable for deployment on devices with limited resources, such as embedded systems used in robotics and it illustrates the abilities of cost-effective models in real-world applications.

2) Object Detection Experiment: We also conducted an experiment to evaluate the performance of our custom MobileNetV2-based model on an object detection task. We used the Pascal VOC[11] dataset for this experiment, which is a widely used benchmark for object detection tasks. Model was modified by added **SSD(Single Shot MultiBox Detector)** head on top of the MobileNetV2-based backbone to enable object detection capabilities.

E. Validation and Simulation

The application includes a built-in *Inverse Kinematics* (IK) solver based on the **Levenberg-Marquardt (LM)** algorithm with *numerical Jacobian* computation. This solver enables real-time simulation of basic robot arm movements for popular industrial models such as ABB, AUBO, and others. While the built-in simulator supports rapid environment setup and motion testing, it lacks advanced physics simulation essential for realistic interaction and dynamics validation. To address this, the system is integrated with **ROS** and **Gazebo**, providing a comprehensive physics-based simulation environment. This seamless workflow enables thorough validation of the generated IK code

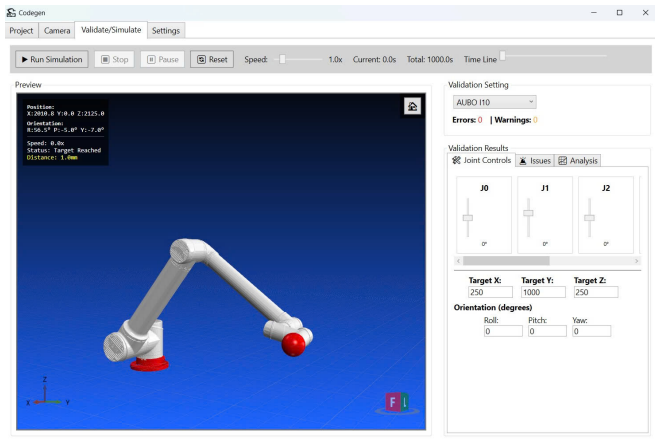


Fig. 4: Built-in simulation environment for testing robot movements.

1) Validation Setup: To evaluate the performance of the code generation system, we used GAZEBO simulator integrated with ROS2. The simulator provides a realistic environment for testing the generated code without the need for physical hardware. The main outcomes is make sure the generated code can successfully control the robot arm to perform the specified tasks.

2) Validate Environment Setup: The validation environment consists of a virtual robot arm model, a set of objects to interact with, and predefined tasks that the robot arm needs to accomplish. The robot arm model is designed to mimic the kinematics and dynamics of a real-world industrial robot, allowing for accurate testing of the generated code. We setup several virtual cameras in the simulation environment to capture visual data from different perspectives. These cameras provide the necessary visual context for the Self verify module of code synthesis system allowing it to analyze the execution of generated code.

IV. Results and Discussion

A. Experiment Setup

To assess the effectiveness of our code generation system, we designed a series of experiments comparing the performance of models with and without 3D vision context. We selected a set of representative robotic tasks that require precise manipulation and control of the robot arm. The experiments were conducted in a controlled environment using the GAZEBO simulator integrated with ROS2. The main outcomes is measured based on two metrics: time taken to complete the task and the number of attempts required to achieve successful execution.

B. Analysis on Performance of 3D Vision Context

TABLE I: Comparison of models with and without 3D vision context

3D Vision Context	Time(s)	Attempts
Yes	–	–
No	–	–

C. Comparison of GenCode System and Manual Coding

TABLE II: Performance comparison between code generation system and manual coding

Method	Time(s)	Attempts
GenCode System	–	–
Manual Coding	–	–

V. Conclusion

VI. Future Work

VII. Acknowledgment

Our team would like to express sincere gratitude to **Viet-Dynamic JSC** for providing the opportunity and resources to carry out this project. We would also like to extend our special thanks to our instructor, **Mr. Tran Trong Toan**, for his dedicated guidance, valuable feedback, and continuous support throughout the internship. His expertise and encouragement were essential in shaping our understanding and successful application of Robotics-Oriented Vision Language models in robotics.

REFERENCES

[1] Y. Mu, J. Chen, Q. Zhang, S. Chen, Q. Yu, C. Ge, R. Chen, Z. Liang, M. Hu, C. Tao *et al.*, "Robocodex: Multimodal code generation for robotic behavior synthesis," *arXiv preprint arXiv:2402.16117*, 2024.

[2] S. Xie, H. Wang, Z. Xiao, R. Wang, and X. Chen, "Robotic programmer: Video instructed policy code generation for robotic manipulation," *arXiv preprint arXiv:2501.04268*, 2025.

[3] X. Chu, L. Qiao, X. Lin, S. Xu, Y. Yang, Y. Hu, F. Wei, X. Zhang, B. Zhang, X. Wei *et al.*, "Mobilevlm: A fast, strong and open vision language assistant for mobile devices," *arXiv preprint arXiv:2312.16886*, 2023.

[4] G. R. Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl *et al.*, "Gemini robotics: Bringing ai into the physical world," *arXiv preprint arXiv:2503.20020*, 2025.

[5] X. Chu, L. Qiao, X. Zhang, S. Xu, F. Wei, Y. Yang, X. Sun, Y. Hu, X. Lin, B. Zhang *et al.*, "Mobilevlm v2: Faster and stronger baseline for vision language model," *arXiv preprint arXiv:2402.03766*, 2024.

[6] A. Abdolmaleki, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, A. Balakrishna, N. Batchelor, A. Bewley, J. Bingham, M. Bloesch *et al.*, "Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer," *arXiv preprint arXiv:2510.03342*, 2025.

[7] G. Jocher and J. Qiu, "Ultralytics yolo11," 2024. [Online]. Available: https://github.com/ultralytics/ultralytics

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[9] AI Technology and Application Research Lab. (2025) Home. [Online; accessed 10-October-2025]. [Online]. Available: https://aita-lab.github.io/

[10] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[11] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.