Final Project SVLR Application

1st Aly Khater
Electrical and Computer Engineering
Santa Clara University
Santa Clara, United States
akhater@scu.edu

2nd Vincent Ying
Robotics and Automation
Santa Clara University
Santa Clara, United States
vying@scu.edu

***Abstract*—In this project, we combine the approach of Large Language Models and Visual Language Models to generate robotic actions, succinctly called Visual Language Action (VLA) models [1]. VLA models recently emerged as a powerful way to integrate visual perception, natural language understanding, and action generation to improve a robot's ability to interpret and execute tasks from multimodal input. This project explores the potential to improve robot learning by combining the interpretive abilities of large-scale vision language models with robotic manipulation tasks.**

***Index Terms*—Robotics, Vision Language Model**

I. Introduction

Robotic systems increasingly utilize Vision-Language Models (VLMs) and Large Language Models (LLMs) to interpret multimodal instructions and perform complex tasks. However, existing methods have limitations in their scalability and generalization, particularly when adapting to different scenarios without the need for significant retraining efforts. Scalable, Training-Free Visual Language Robotics (SVLR) provides training-free strategies to better leverage current pre-trained models.

This project delves into the integration of VLMs and LLMs using SVLR methodologies. By combining the pre-trained models, the goal is to enhance the robot's ability to interpret tasks while reducing the computational load required with additional training or fine-tuning. The motivation for this work comes from the need for robot systems that can adapt to various tasks and environments across multiple contexts.

II. Related Work

RT-1: Robotics Transformer for Real-World Control at Scale [2] is a robot control model that uses images and simple text instructions to decide how a robot should move and interact with objects. It learns from massive amounts of diverse robot data, allowing it to generalize and perform new tasks even in unfamiliar environments.

RT-2: VLA Models Transfer Web Knowledge to Robotic Control [3] expands on the previous RT-1 project by combining internet-scale vision-language knowledge with robotic control, allowing robots to understand commands they've never seen during training. By converting robot actions into textual tokens, Brohan et al. integrates semantic reasoning, enabling advanced multi-step reasoning tasks.

Interactive Lanugage is a framework enabling real-time, natural-language-guided robot control. It is trained via behavioral cloning on nearly 600,000 language-labeled trajectories. The policy allows iterative human guidance for complex tasks, like creating a smiley face or sorting blocks. It significantly expands robots' interactive capabilities in real-world tasks.

SayCan [4] is a robotic framework integrating Large Language Models (LLMs) with robot affordances introduced by Google. It enables robots to interpret high-level, abstract language instructions and translate them into executable actions by combining semantic reasoning and environmental feasibility. It works great on tasks that require intricate, multi-layered steps to complete.

AutoRT [5] is a framework designed to orchestrate a fleet of robots in an unstructured environment with minimal human intervention. It utilizes VLMS and LLMs to generate descriptions of its environment and proposed natural language instructions for potential tasks that the robot can accomplish in a feasible and safe manner.

Scalable Visual Language Robotics (SVLR) is a modular and scalable framework that [6] uses multiple AI models [7]–[10] to process visual input and language instructions, enabling robots to perform complex tasks. It achieves this by leveraging low parameter AI models to process visual inputs and language instructions, enabling robots to execute complex tasks without expensive training. The framework does not require retraining and runs on consumer GPUs. It allows for easy integration of new robotic tasks and robots by simply adding text descriptions and task definitions.

The overall architecture of SVLR is depicted in figure 1.

- Robot Info
- Perception Module
- Large Language Model
- Action Manager

In the **robot info** module, the robot's capabilities are detailed in a text format, which is provided to the LLM. It also contains the description and link to the robot tasks set. These tasks are selected and parametrized by the framework to execute given language instruction.

The framework requires two inputs: the user's language instruction and an image of the environment. These inputs are processed through two modules for processing, the LLM

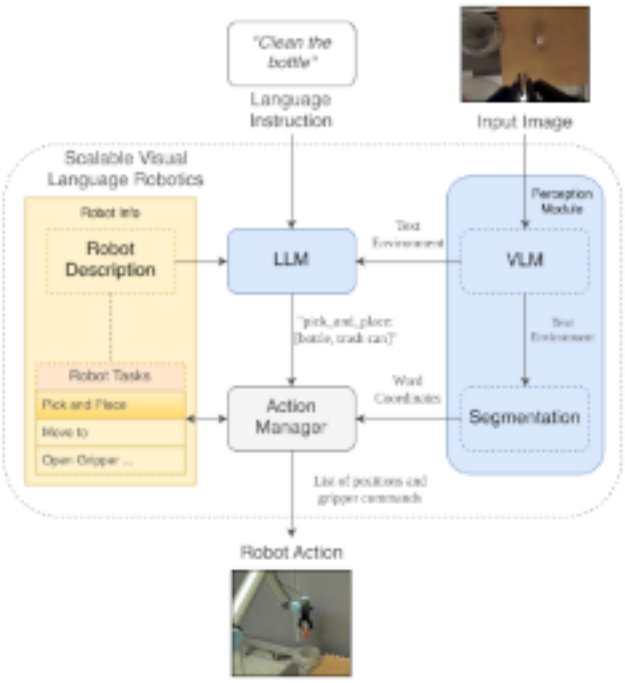

Fig. 1. SVLR Architecture

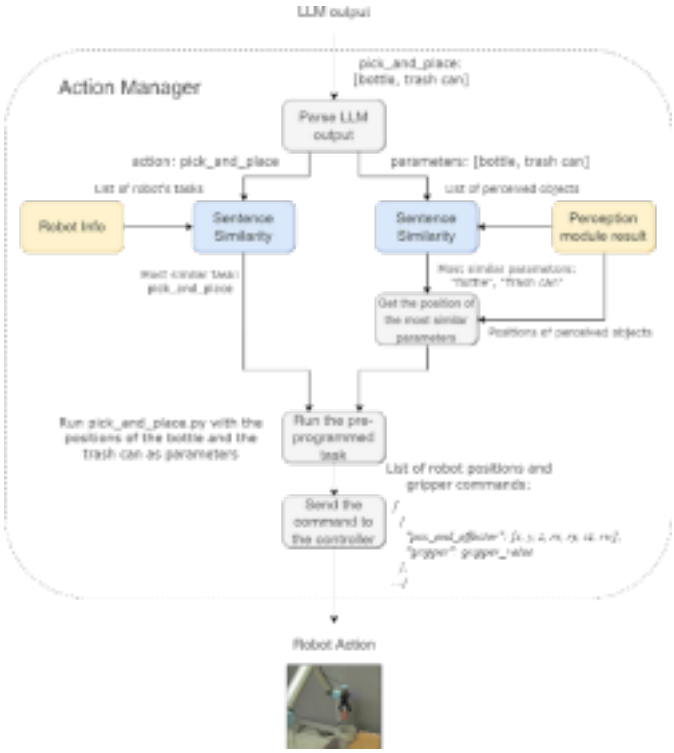

Fig. 2. Action Manager From LLM output to robot action

and perception module. The **LLM** determines the sequence of tasks required to fulfill the instructions in the given text.

The **perception module** feeds an image to the VLM in order to determine what objects are on the scene and then passes those objects to a segmentation model to get the centroid calculations. Then, utilizing the camera with a known offset position between the camera and the robot, the perception module converts the pixels from the image frame to real world coordinates based off of the UR10e base frame, which will give the coordinates for where the robot needs to move.

The **action manager** takes an LLM output from a text prompt to determine which actions the system needs to perform according to the objects and description provided from the user and VLM. The LLM then sends the actions to the sentence transformer, which has a predefined amount of actions it can do. In parallel, the perception module sends coordinates to the module, combines the output of the actions and the positions of the objects to run the specified task. In this figure 2, running a pick and place task given the objects known positions as parameters. This generates the robot positions and commands to move the robot from the controller, which generates a robot action.

III. Methods

Our project implements the following steps:

1) Utilize a camera to provide an image from the end-effector of the UR10e to the SVLR VLM
2) Incorporate SVLR for image segmentation and object detection of the live environment
3) Send feedback from the camera and our generated actions to create trajectories
4) Execute on given trajectories with UR10e robot arm to perform tasks

A. System Components

Hardware and software listing of experimental setup,

- UR10e Robot
- Robotiq 2F-85
- Polyscope 5
- Ubuntu 24.04(Noble Numbat) with Real-time kernel
- Robot Operating System 2 (Jazzy)
- RVIZ

1) Hardware Components: A remote GPU server is utilized to perform inference with VLM and LLM models in the SVLR framework. The SSH protocol is utilized to access the server. Transfer of the image needed for the VLM is accomplished through scp from the host to remote server. Network sockets are utilized to setup a communication link between the server and the host PC for the transfer of data.

2) Software Components: Software components needed for physical application of SVLR with the UR10e robot arm.

- **URSim** provides a safe, interactive environment of the UR10e robot in a simulated environment. There is no risk of collision between the robot arm and actual physical objects. It is relatively simple to transfer anything run in simulation to the actual robot arm, requiring only a simple change of IP address. Running the simulation furnishes a local IP for access.

- **URCaps** is needed to work with the actual robot arm. It needs to be running on the robot with a network connection between the robot and the host PC. This program requires UR Driver to be concurrently active on the host PC, which initializes the drivers and prepares the robot controllers for use. We utilized the Scaled Joint Trajectory controller, which is the safest controller option as it will trigger an emergency stop if the robot arm goes beyond safety configurations.
- **MoveIt** is the most popular manipulation and motion planning software that provides interactive 3D visualizations.

B. Experimental Setup

Initial setup of the UR10e is as follows,

1) To properly utilize the robot, we referred to user manual for setup with appropriate settings, such as passwords and safety configurations. Remote control was not possible until these settings were set.
2) Ensure automatic IP assignment with DHCP protocal with the lab router over the network.
3) Open port access the Robotiq Gripper, to send commands remotely from a pc to the robot to actuate the gripper.
4) Enable Real Time Kernel Processing, splitting control and I/O into two different interfaces that don't block each other.
5) Install ROS2 (Jazzy) on host PC.

C. System Flow

Application of the SVLR framework with an UR10 robot arm is in figure 3. Application flow operates under the following steps,

1) Robot: URCaps will be programmed to accept the IP from the HostPC.
2) Host PC: The UR Robot Driver will be started on the Host PC. This will allow the Robot to be connected with the Host PC.
3) Remote Server: From the Host PC, the user will SSH into the Remote server to start the main LLM-VLM code. This will connect the Host PC directly to the Remote Server.
4) Remote Server: An image will then be taken for inferencing, which will then detect and segment what objects exist in the image.
5) The LLM will ask the user for input on what it wants to do to generate an actionable dictionary to follow.
6) Save the actionable dictionary in a Json file and send it over the network to the host PC.
7) The host PC converts the Json file into data that can be used to send commands to the UR10e.
8) Observe action execution by the UR10e.

IV. Experimental Results

Figure 4 depicts image capture from the depth camera and segmentation of any detected objects. We qualitatively assess the robot performance in pick-and-place tasks. Trials were run to measure how fast the robot can perform a task and its accuracy of being able to pick and place an object. We provide feedback to visually inspect the process, examining the responsiveness and smoothness of robot actions in the environment, and detail our own observations.

V. Conclusion

This project showcases the effectiveness of LfD and VLA models in robot task execution. By using our human demonstration data, VLM, and SVLR segmentation, we enable the UR10 robot to perform actionable tasks such as pick and place. Evaluation based off metrics and human feedback allows for a deeper understanding of the systems performance, limitations, and challenges for future work.

Having to use a remote server is vastly different than what the github repo provided. We had to learn how to utilize network sockets to run SVLR with a GPU server. Network security port settings for the robot need to be adjusted to allow external control.

Calibrations of the end effector needs to be finetuned. We have the robot moving in a controlled area, but the calibration for robot arm pose needs to be adjusted so that the robot moves towards the actual object.

A. Future Work

1) Calibrations
2) Try bigger models
3) Qualitative and Quantitative analysis using different models
4) Use a depth camera to generate a 3D segmented map to get a map of the objects
5) The UR10e performs the actions.

Acknowledgment

Thanks to Professor Kyriani for her advice and guidance and Krishna Kodpur for his help with hardware support.

References

[1] E. Collaboration, A. O'Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, A. Tung, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Gupta, A. Wang, A. Kolobov, A. Singh, A. Garg, A. Kembhavi, A. Xie, A. Brohan, A. Raffin, A. Sharma, A. Yavary, A. Jain, A. Balakrishna, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Wulfe, B. Ichter, C. Lu, C. Xu, C. Le, C. Finn, C. Wang, C. Xu, C. Chi, C. Huang, C. Chan, C. Agia, C. Pan, C. Fu, C. Devin, D. Xu, D. Morton, D. Driess, D. Chen, D. Pathak, D. Shah, D. Büchler, D. Jayaraman, D. Kalashnikov, D. Sadigh, E. Johns, E. Foster, F. Liu, F. Ceola, F. Xia, F. Zhao, F. V. Frujeri, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi, G. Berseth, G. Kahn, G. Yang, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. B. Amor, H. I. Christensen, H. Furuta, H. Bharadhwaj, H. Walke, H. Fang, H. Ha, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Drake, J. Peters, J. Schneider, J. Hsu, J. Vakil, J. Bohg, J. Bingham, J. Wu, J. Gao, J. Hu, J. Wu, J. Wu, J. Sun, J. Luo, J. Gu, J. Tan, J. Oh, J. Wu, J. Lu, J. Yang, J. Malik, J. Silvério, J. Hejna, J. Booher, J. Tompson, J. Yang, J. Salvador, J. J. Lim, J. Han, K. Wang, K. Rao, K. Pertsch,

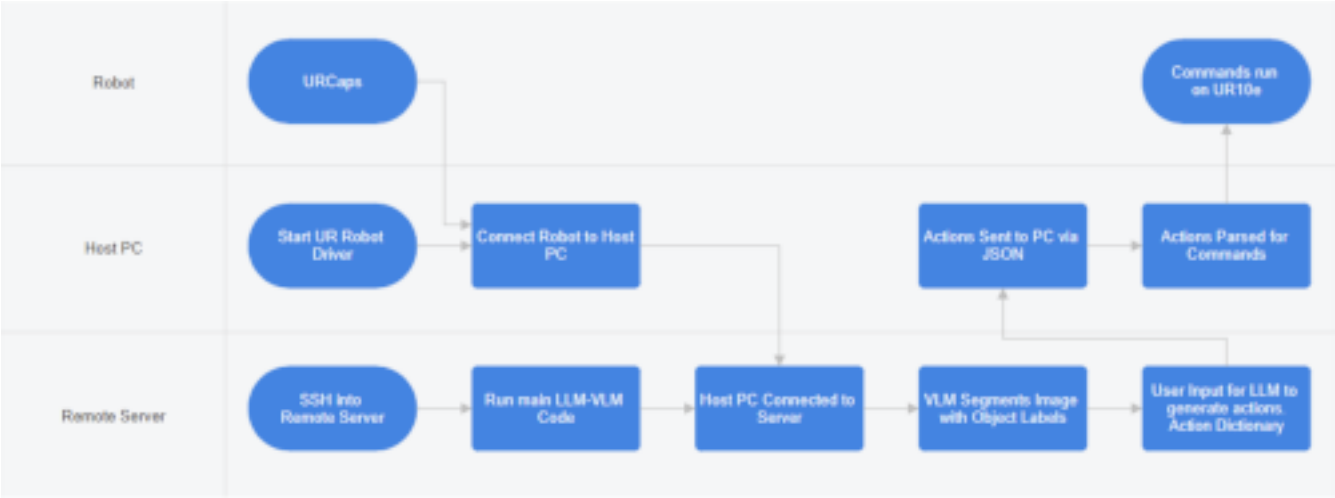

Fig. 3. SVLR Application Components

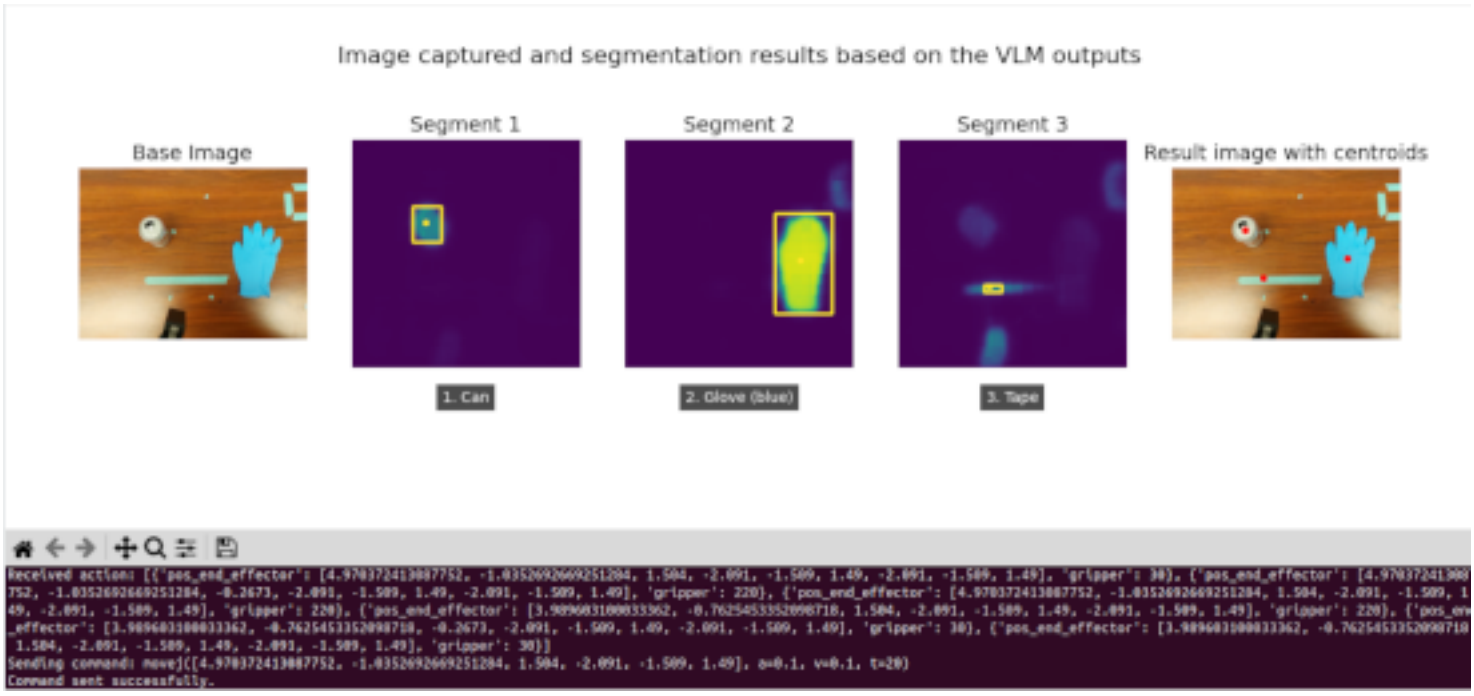

Fig. 4. Perception Processing

K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Black, K. Lin, K. Zhang, K. Ehsani, K. Lekkala, K. Ellis, K. Rana, K. Srinivasan, K. Fang, K. P. Singh, K.-H. Zeng, K. Hatch, K. Hsu, L. Itti, L. Y. Chen, L. Pinto, L. Fei-Fei, L. Tan, L. J. Fan, L. Ott, L. Lee, L. Weihs, M. Chen, M. Lepert, M. Memmel, M. Tomizuka, M. Itkina, M. G. Castro, M. Spero, M. Du, M. Ahn, M. C. Yip, M. Zhang, M. Ding, M. Heo, M. K. Srirama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Liu, N. D. Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, O. Bastani, P. R. Sanketi, P. T. Miller, P. Yin, P. Wohlhart, P. Xu, P. D. Fagan, P. Mitrano, P. Sermanet, P. Abbeel, P. Sundaresan, Q. Chen, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Mart'in-Mart'in, R. Baijal, R. Scalise, R. Hendrix, R. Lin, R. Qian, R. Zhang, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Lin, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Tulsiani, S. Song, S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Kumar, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Pang, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Dou, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, Z. Fu, and Z. Lin, "Open x-embodiment: Robotic learning datasets and rt-x models," 2024. [Online]. Available: https://arxiv.org/abs/2310.08864

[2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu,

U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, "Rt-1: Robotics transformer for real-world control at scale," 2023. [Online]. Available: https://arxiv.org/abs/2212.06817

[3] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," 2023. [Online]. Available: https://arxiv.org/abs/2307.15818

[4] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng, "Do as i can, not as i say: Grounding language in robotic affordances," 2022. [Online]. Available: https://arxiv.org/abs/2204.01691

[5] M. Ahn, D. Dwibedi, C. Finn, M. G. Arenas, K. Gopalakrishnan, K. Hausman, B. Ichter, A. Irpan, N. Joshi, R. Julian, S. Kirmani, I. Leal, E. Lee, S. Levine, Y. Lu, I. Leal, S. Maddineni, K. Rao, D. Sadigh, P. Sanketi, P. Sermanet, Q. Vuong, S. Welker, F. Xia, T. Xiao, P. Xu, S. Xu, and Z. Xu, "Autort: Embodied foundation models for large scale orchestration of robotic agents," 2024. [Online]. Available: https://arxiv.org/abs/2401.12963

[6] M. Samson, B. Muraccioli, and F. Kanehiro, "Scalable, training-free visual language robotics: A modular multi-model framework for consumer-grade gpus," 2025. [Online]. Available: https://arxiv.org/abs/2502.01071

[7] T. Lüddecke and A. S. Ecker, "Image segmentation using text and image prompts," 2022. [Online]. Available: https://arxiv.org/abs/2112.10003

[8] M. Abdin, J. Aneja, H. Awadalla, A. Awadallah, A. A. Awan, N. Bach, A. Bahree, A. Bakhtiari, J. Bao, H. Behl, A. Benhaim, M. Bilenko, J. Bjorck, S. Bubeck, M. Cai, Q. Cai, V. Chaudhary, D. Chen, D. Chen, W. Chen, Y.-C. Chen, Y.-L. Chen, H. Cheng, P. Chopra, X. Dai, M. Dixon, R. Eldan, V. Fragoso, J. Gao, M. Gao, M. Gao, A. Garg, A. D. Giorno, A. Goswami, S. Gunasekar, E. Haider, J. Hao, R. J. Hewett, W. Hu, J. Huynh, D. Iter, S. A. Jacobs, M. Javaheripi, X. Jin, N. Karampatziakis, P. Kauffmann, M. Khademi, D. Kim, Y. J. Kim, L. Kurilenko, J. R. Lee, Y. T. Lee, Y. Li, Y. Li, C. Liang, L. Liden, X. Lin, Z. Lin, C. Liu, L. Liu, M. Liu, W. Liu, X. Liu, C. Luo, P. Madan, A. Mahmoudzadeh, D. Majercak, M. Mazzola, C. C. T. Mendes, A. Mitra, H. Modi, A. Nguyen, B. Norick, B. Patra, D. Perez-Becker, T. Portet, R. Pryzant, H. Qin, M. Radmilac, L. Ren, G. de Rosa, C. Rosset, S. Roy, O. Ruwase, O. Saarikivi, A. Saied, A. Salim, M. Santacroce, S. Shah, N. Shang, H. Sharma, Y. Shen, S. Shukla, X. Song, M. Tanaka, A. Tupini, P. Vaddamanu, C. Wang, G. Wang, L. Wang, S. Wang, X. Wang, Y. Wang, R. Ward, W. Wen, P. Witte, H. Wu, X. Wu, M. Wyatt, B. Xiao, C. Xu, J. Xu, W. Xu, J. Xue, S. Yadav, F. Yang, J. Yang, Y. Yang, Z. Yang, D. Yu, L. Yuan, C. Zhang, C. Zhang, J. Zhang, L. L. Zhang, Y. Zhang, Y. Zhang, Y. Zhang, and X. Zhou, "Phi-3 technical report: A highly capable language model locally on your phone," 2024. [Online]. Available: https://arxiv.org/abs/2404.14219

[9] Z. Chen, J. Wu, W. Wang, W. Su, G. Chen, S. Xing, M. Zhong, Q. Zhang, X. Zhu, L. Lu, B. Li, P. Luo, T. Lu, Y. Qiao, and J. Dai, "Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks," 2024. [Online]. Available: https://arxiv.org/abs/2312.14238

[10] e. a. N. Reimers, J. Gante. (1999) HuggingFace all-minilm-l6-v2. [Online]. Available: https://huggingface.co/sentence-transformers/all-minilm-l6-v2