Enabling Waypoint Generation for Collaborative Robots using LLMs and Mixed Reality

Cathy Mengying Fang[1,§], Krzysztof Zieliński[2,3,§], Pattie Maes[1], Joe Paradiso[1], Bruce Blumberg[3], Mikkel Baun Kjærgaard[2]

***Abstract*— Programming a robotic arm involves a lot of complexity, as it demands from the user to have a good command of its specific programming language and a clear awareness of its physical constraints. We propose a framework that simplifies robot deployment and reduces the entry barrier for the user. It uses Large-Language Model (LLM) for prompt processing, workspace understanding, and waypoint generation. It also employs Augmented Reality (AR) to provide visual feedback of the planned outcome. We showcase the effectiveness of our framework with a simple pick-and-place task, which we implement on a real robot. Moreover, we present an early concept of expressive robot behavior and skill generation that can be used to: communicate with the user and learn new skills (e.g., object grasping, door opening, etc.).**

I. Introduction

Manual programming has been the conventional way of setting up industrial and collaborative robots – using a specific input controller to define various robot tasks, such as setting waypoints, connecting I/O ports to auxiliary equipment, and creating program logic, etc. Usually, each robot arm manufacturer has a bespoke controller and programming language, which is often presented as a simple ladder diagram or graphical block representation. To overcome the limitations of robot-specific interface/language, more abstract programming methods have been developed, such as the Robot Operating System (ROS) which aims to provide a universal programming method.

The methods mentioned above require the robot programmer to have some basic computer science and robotics skills to grasp concepts such as singularities, joint space, inverse kinematics, etc. However, most of the world's manufacturing is done by Small and Medium Enterprises (SMEs) who often do not have such skills, which limits their ability to automate their processes. To lower the barrier for robot programming, more automated methods exist [1], such as Programming by Demonstration (PbD) which allows users to demonstrate robot skills to the system, rather than coding them.

In addition to PbD, recent advances in generative AI, specifically the creation of foundation models, allow researchers to solve diverse robotics tasks not confined by the tasks, environments, and robot types in the prior training data [2]. This presents an immense opportunity not only for scaling robotic operations but also for improvements in human-robot interaction, given the semantic nature of the I/O of the large language models. Specifically, this could further lower the barrier of laypersons to operate a collaborative robot.

In this paper, we propose a novel way to interact with a collaborative robot using natural language with LLMs. Our framework takes in the speech input from the user and in real-time generates a series of waypoints. The generated waypoints can be simultaneously previewed in an augmented reality headset overlaid on top of the physical environment, and the robot receives the waypoints directly as a result of the generation and follows the approved plan. Furthermore, we also demonstrate an early exploration of automatic robotic skill generation, leveraging the generative nature of the LLM framework. We show that our framework can generate expressive gestures such as "nodding", and this capability can be generalized to other skills such as the "pick" skill in a pick-and-place task.

II. Related Work

We start the Related Work section with current approaches to automating robot programming – "Programming by Demonstration". We then touch upon the use of Mixed Reality in assisting robot programming. Finally, we conclude with recent breakthroughs and experiments using foundation models like LLMs to scale up robotic tasks.

A. Programming by Demonstration Interfaces

There exist different ways to gather and transmit the trajectory data to the robotics system. For robot arms, it is typically kinesthetic teaching [3], where the user holds the robot's end-effector and moves it along the desired trajectory. Takano et al. [4] focus on learning and reproducing basic motion primitives. Caccavale et al. [5] additionally introduce structured cooperative tasks from human demonstrations. Kim et al. [6] implement three layers of kinesthetic teaching (joint, task and contact) for dual-arm robot control.

Kinesthetic teaching requires the human user to use both hands to manipulate the robot arm, which becomes more troublesome when the robot has more limbs, e.g., a humanoid robot. For that case, motion capture based approaches can be used to imitate human body movements [7], [8]. In this setup, the human teacher is equipped with sensors that are externally tracked with optical or magnetic tracking systems. The major challenge with thisapproach is the correspondence

[1]Authors are with MIT Media Lab catfang, pattie, joep@media.mit.edu

[2]Authors are with the Faculty of Engineering, the Maersk Mc Kinney Moller Institute, University of Southern Denmark krzi, mbkj@mmmi.sdu.dk

[3]Authors are with Universal Robots A/S krzi, brbl@universal-robots.com

[§] Equal contribution

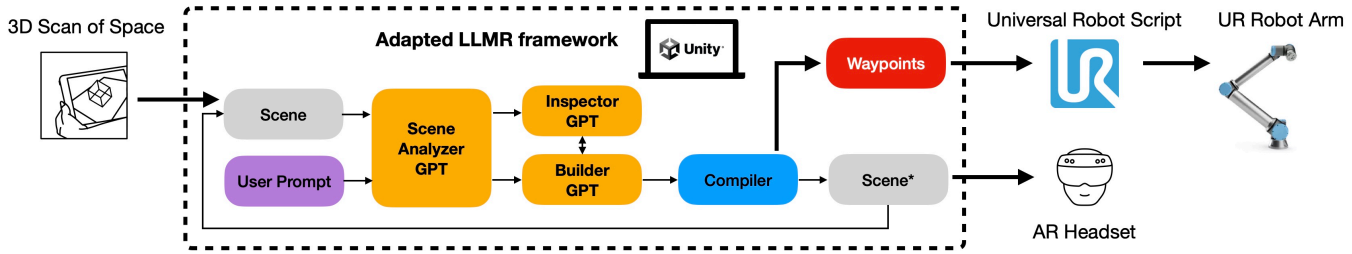

Fig. 1. Overview of our framework. Starting from the top left, a 3D scene (optionally scanned if does not exist already) and a user prompt are fed to adapted LLMR framework, which is an orchestration of prompt-engineered GPT modules. The adapted framework outputs the trajectory based on the user prompt, which is converted to Universal Robot script, readable by the robot arm. The user also sees the rendered trajectory in the AR headset.

problem where the demonstrator and imitator do not have the same kinematic and dynamic properties (e.g., human vs. humanoid). Alternatively, prior work used different mappings [9] or tools [10], [11]to mediate the difference in the properties.

Teleoperation does not face the correspondence problem (i.e., the operator is constrained to the robot's sensors), and it also allows the operator to perform the task remotely. The human teacher requires some type of input device and feedback such as joysticks [12] or custom control devices [13], [14] to operate the robot. *Mobile ALOHA* [15] introduces whole-body teleoperation where the user is physically tethered to the device and can control both arms and the wheels.

Speech is another interface for PbD, although not so popular due to its ambiguities and limited command dictionary for robot control. Norberto [16] and Tasevski et al. [17] use voice-controlled industrial robot to perform pick-and-place, and welding tasks with a fixed set of voice commands. We propose a novel way for automatic robot programming, as it utilizes voice modality for robot control with natural language processing, hence it does not force the user to memorize a constrained set of robot commands.

B. Augmented Reality in Robotics

Augmented Reality (AR) is well researched topic as an interface for Human-Robot Interaction [18], robot control and teleoperation, etc. Quintero et al. [19] use head-mounted display to allow users to create and preview robot trajectory. Gong et al. [20] create robot instructions using projection-based AR for grasping tasks. While Stadler et al. [21] explore handheld AR interface for industrial robot programming. Zielinski et al. [22] augment robot arm with contextual data, such as software limits, joint temperatures, end-effector coodinate frame, etc. Chen et al. [23] control mobile robot with drag-and-drop prompt using head-mounted AR display.

C. Situating LLM with the Real World for Robotics

Kawaharazuka et al. overview the application of foundation Large Language Models (LLM) and Vision-Language Models (VLM) in real-world robotics [2]. *Palm-E* is one such example of a general-purpose multimodal language model that is trained for embodied reasoning (e.g., robot planning) [24]. Besides training new foundation models with new modalities, many can already solve many challenges in robotics by adapting existing foundation models. *ProgPrompt* uses LLM to help with task planning and generate a correct sequence of steps in different contexts [25]. Ahn et al. focus on a key limitation of using LLM to drive robotic behavior, which is the awareness of the robot's capabilities. They ground the output to the capabilities of the robot such that the output is actually feasible to execute [26]. *AutoRT* solves the challenge of lack of data grounded in the physical world by scaling up the deployment of auto-instructing robots using existing foundation models [27]. On the other hand, *SpatialVLM* solves the same problem by developing an automatic 3D spatial visual QA data generation framework [28]. Despite recent advances, they point out that the current models are still not suitable for tasks that require collaboration with humans [2].

Our work hopes to contribute to the improvement of human-robot collaboration and interaction. We do so by building on top of the Large Language Model for Mixed Reality (LLMR) framework [29], a framework that allows real-time natural language prompting of interactive 3D scenes. We take advantage of the ease of integration of the LLM-based framework within the mixed reality environment. The user can instruct the robot arm with natural language and preview the results directly in an AR headset, for example.

III. Framework Overview

Our framework encompasses the integration of LLMs, the Unity game engine, the HoloLens 2 AR headset, and URScript (proprietary Universal Robots' programming language) to achieve real-time AI-assisted planning of robot waypoints (Figure 1). We now discuss each component of the framework in detail.

Digital representation of the physical scene – We start by creating a replica of the physical scene in Unity, a rendering engine [30]. We used Polycam [31] to capture a 3D model of the real-world objects and environment, and we imported the 3D model of the Universal Robots (UR10e) [32] (Figure 3). The user can alternatively provide the scan of the physical world (most industrial plants likely have such models). The scan is used as a reference for the collider boundaries we manually configured in Unity, a manual step that can be refined and replaced with a sophisticated scene understanding model [33]. We named the scene objects and the robot's sub-components according to their semantic meaning. We also set up a reachability sphere that represents the conservative upper bound of the reach of the robot's endpoint. At the moment, we assume the physical objects are static (i.e., they do not move when the robot is moving and thus are not

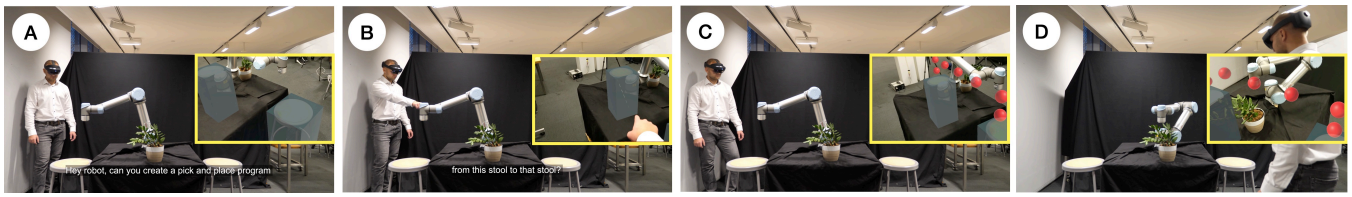

Fig. 2. Example interaction between the user and the collaborative robot arm enabled by our framework. A-B: The user is wearing a HoloLens2 AR headset and instructs the robot to create a pick-and-place program between two stools. C: Our framework (running within Unity on a separate laptop) generates a series of waypoints (indicated as the red spheres) and the waypoints are streamed to preview in the AR headset. D: Once the user is satisfied with the waypoints, the robot receives the command from Unity and then follows the waypoints.

tracked). Future work can use real-time capture methods (e.g., OptiTrack or Aruco Markers) to update the scene in Unity. Combined with the user prompt, the scene is provided to our LLM system to generate a series of waypoints, which we discuss next.

LLM integration – We use our framework for generating the viable waypoints. We built upon LLMR [29]. LLMR is designed for simulating scenarios in virtual environments. We pushed the real-world implication of LLMR by connecting the simulated outcome to a physical robot. We fine-tuned a specific prompt for LLMR that provides the context of the task. An example prompt is in the Appendix I. We also optimized and improved the Scene Analyzer component in LLMR to pay more attention to the robot and the relevant objects. The outputs of this step are the code that generates a series of sequential waypoints and the visualization of these waypoints in the virtual scene. A sample of the generated code is in the Appendix II. In the iteration process of fine-tuning the prompt, we found that the generated trajectory pattern does not vary significantly across runs unless the prompt describes very specific constraints (such as specific obstacles to avoid or constraints of the robot arm motion).

Visualization of the scene in AR – We deploy our framework's run-time compiler on a PC that acts as the server, and we build upon existing streaming protocols (holographic remoting for a HoloLens 2) [34], [35] to stream the generated results to the head-mounted display.

Virtual robot control in Unity – For the system to test if a point is reachable or not, we have implemented a simplified inverse kinematics (IK) controller in Unity. As an argument, it takes waypoint positions in Cartesian space w.r.t. (with reference to) the robot base and returns joint angles.

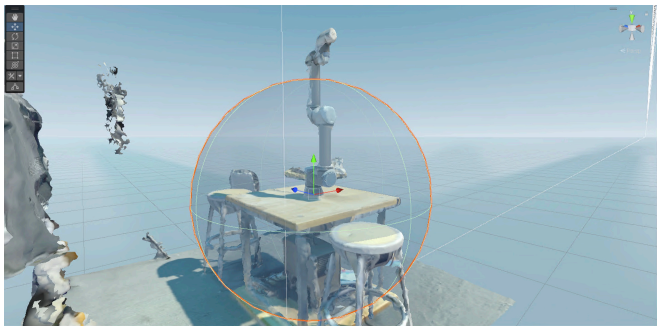

Fig. 3. A pre-scanned scene is loaded in the Unity environment, along with a model of the collaborative robot arm from Universal Robots. A translucent reachability sphere is shown to indicate the conservative estimation of the maximum reach of the robot arm.

The controller is based on gradient descent to calculate the trajectory between waypoints but does not account for singularities.

Real robot control from Unity – To run our framework on the real robot, we used Universal Robot's proprietary Primary Interface for two-way communication with the arm in Unity. Once we obtain satisfactory waypoints, we can directly upload them to the robot. In this mode, there is no need to provide custom inverse kinematics, since we utilize Universal Robots' control box for trajectory planning in-between waypoints.

IV. Example Scenario

A common task is pick-and-place, where the robot arm needs to move between two end points repeatedly while avoiding colliding with objects in the room or itself. Typically, the user would need to manually program the robot (first in simulation and then in the physical environment).

We envision the following end-to-end interaction where the user can achieve this with natural language instructions. The user wants to create a custom pick-and-place program between two places. They can point at the two locations and say "between this stool and that stool" (Figure 2 A-B). Our framework takes in the user prompt and the physical scene to create a detailed instruction. The instruction and scene information go through the LLM framework and the framework outputs a series of waypoints that are then visualized in the Augmented Reality headset (visualized as the red spheres, Figure 2 C). The user is able to see and verify the proposed waypoints in-situ. After the user confirms, the waypoints are directly sent to the robot, where the robot's endpoint follows the generated sequence (Figure 2 D).

V. Early Exploration: Few-shot Generation of Expressive Robot Behavior and Skills

Overlaying visualization of the robot's trajectory in the real world is one of the ways that helps human users to directly manipulate and communicate with the robot. However, there lacks a channel and modality, such as gesturing [36], for the robot to communicate back to the human user. We were also inspired by how the collaborative robot can be programmed to perform alongside a human dancer [37].

Following the extended animation generation capability of LLMR [38], we also experimented with using LLM to generate anthropomorphic expressions of the robot arm. We provided some context about the robot's joint limitations and a few examples of hand-crafted animations. We used

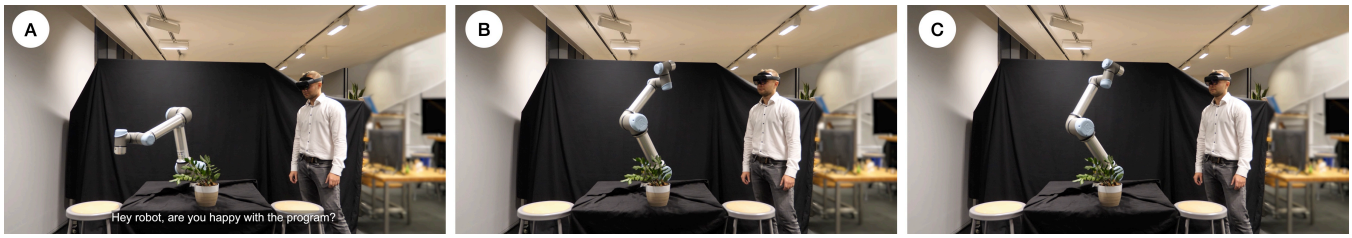

Fig. 4. An example of an expressive response from the robot back to the user. A: The user asks the robot if it is happy with the generated program. B-C: The robot responds by nodding back at the user, where the nodding animation is generated by our framework.

the gpt4-1106-preview model with a temperature of 0.1. A sample of the generated joint positions is in the Appendix IV. We imagine a scenario where the robot is asked whether they are pleased with the result, they could nod or shake their head back to the user (Figure 4). We consider "nodding" and "shaking" the head to be similar to the skill of "grasping". Different from other LLM-based generative approach that interpolates between different pre-existing skills, we generate new anthropomorphic animations with a few-shot example in the metaprompt. The resulting animations are still exploratory, as the robotic arm's inherent limitation in degrees of freedom cannot result in complex expressions. In addition, unlike animals or humans where ample examples of different behaviors exist, robot expressions are more idiosyncratic, and thus the quality of the generated output is up to the human's interpretation.

VI. Limitations and Future Work

Singularities – The current LLM system is unaware of the singularities that might emerge from the generated trajectory. The reachability sphere is one way that we prevent the robot from going beyond its range of motion. We used the Universal Robot's IK to allow the robotic arm to reach between the endpoints. Our framework could be incorporated as part of a path planning pipeline to eliminate singularities.

Trajectory Optimization – At the moment our LLM system only generates one possible path during each run. When the LLM's temperature is set to zero, the generated solution is mostly consistent (as a higher temperature usually results in a non-deterministic behavior). Unless the physical constraints of the robot or the target positions change, the LLM's solution stays consistent across generations. However, we do not enforce any optimization parameters, such as ensuring the shortest cycle time or least energy consumption. Our solution is not meant to be compared with the mathematical approach to derive the most optimized path. Rather, focus on creating a draft trajectory that is directly in the place of the user's working environment. Future work might combine the advantages of both approaches.

Feedback from the human – An immediate improvement to our framework is incorporating direct feedback from the human. Specifically, the human can not only directly edit the trajectory by dragging the generated waypoints in the Augmented Reality scene but also "teach" the robot why the changes are preferred. Akin to PbD, the user has the ultimate say in what the trajectory looks like, but our framework would allow the user to overview the entire trajectory in-situ and use language to provide context. A user study is merited to understand the ideal way to engage the user in this process such that the agency of the user is preserved in AI-assisted path planning.

Feedback from the world – Our current system requires manual alignment between the physical and virtual scene, which can be easily solved using trackable markers [39], [40]. However, if an object moves in the scene, the changes would need to be synchronized and updated in the virtual scene. The current computer vision based solution systems lack sophisticated tracking of 3D objects, and running a machine learning model for object classification is computationally expensive. One workaround is to use a combination of vision and language to query the scene [41] or co-embed text and image pixels in the CLIP feature space to achieve scene understanding [42].

Another integration would be sensor data from external sensors deployed in the real world scene. At the moment, the cobot stops when it senses that it has hit something, it would be helpful to incorporate these types of information in the LLM system and to further equip the LLM abilities to perceive the world.

VII. Conclusion

We present an end-to-end framework that addresses the difficulties for users without much prior knowledge to program a collaborative robot. Building on top of the existing framework LLMR that can create interactive scenes in a simulated mixed-reality environment with natural language input, we adapted the input prompt such that a user can use natural language to instruct a collaborative robot. Our adapted framework supports the initial step of path planning, which is the generation of waypoints. The framework considers the unique constraints coming from the environment (e.g., avoid colliding with physical objects) and the robot itself (e.g., maximum reach). We also show initial exploration of the framework's ability to generate new "skills" in the form of animations with few-shot examples. The contribution of this work lies in its potential to leverage existing LLMs to work with embodied systems and environments at human-scale. We propose that this work bridges the gap between the human's and the robot's perception of the world with the ability for both to speak the same "language" (code) in the mixed reality environment.

REFERENCES

[1] G. Ajaykumar, M. Steele, and C.-M. Huang, "A Survey on End-User Robot Programming," *ACM Computing Surveys*, vol. 54, pp. 1–36, Nov. 2022.

[2] K. Kawaharazuka, T. Matsushima, A. Gambardella, J. Guo, C. Paxton, and A. Zeng, "Real-world robot applications of foundation models: A review," *arXiv preprint arXiv:2402.05741*, 2024.

[3] M. Saveriano, S.-i. An, and D. Lee, "Incremental kinesthetic teaching of end-effector and null-space motion primitives," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, (Seattle, WA, USA), pp. 3570–3575, IEEE, May 2015.

[4] W. Takano and Y. Nakamura, "Real-time Unsupervised Segmentation of human whole-body motion and its application to humanoid robot acquisition of motion symbols," *Robotics and Autonomous Systems*, vol. 75, pp. 260–272, Jan. 2016.

[5] R. Caccavale, M. Saveriano, A. Finzi, and D. Lee, "Kinesthetic teaching and attentional supervision of structured tasks in human–robot interaction," *Autonomous Robots*, vol. 43, pp. 1291–1307, Aug. 2019.

[6] P. K. Kim, J.-H. Bae, H. Park, D.-H. Lee, J.-H. Park, M.-H. Baeg, and J. Park, "Dual-arm robot box taping with kinesthetic teaching," in *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, (Xian, China), pp. 555–557, IEEE, Aug. 2016.

[7] D. Kulić, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura, "Incremental learning of full body motion primitives and their sequencing through human motion observation," *The International Journal of Robotics Research*, vol. 31, pp. 330–345, Mar. 2012.

[8] A. Ude, C. G. Atkeson, and M. Riley, "Programming full-body movements for humanoid robots by observation," *Robotics and Autonomous Systems*, vol. 47, pp. 93–108, June 2004.

[9] A. Skoglund, B. Iliev, and R. Palm, "Programming-by-Demonstration of reaching motions—A next-state-planner approach," *Robotics and Autonomous Systems*, vol. 58, pp. 607–621, May 2010.

[10] R. Hanifi Elhachemi Amar, L. Benchikh, H. Dermeche, O. Bachir, and Z. Ahmed-Foitih, "Trajectory reconstruction for robot programming by demonstration," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, p. 3066, June 2020.

[11] B. Maric, M. Polic, T. Tabak, and M. Orsag, "Unsupervised optimization approach to in situ calibration of collaborative human-robot interaction tools," in *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, (Karlsruhe, Germany), pp. 255–262, IEEE, Sept. 2020.

[12] P.-C. Yang, K. Sasaki, K. Suzuki, K. Kase, S. Sugano, and T. Ogata, "Repeatable Folding Task by Humanoid Robot Worker Using Deep Learning," *IEEE Robotics and Automation Letters*, vol. 2, pp. 397–403, Apr. 2017.

[13] K. Kuklinski, K. Fischer, I. Marhenke, F. Kirstein, M. V. Aus Der Wieschen, D. Solvason, N. Kruger, and T. R. Savarimuthu, "Teleoperation for learning by demonstration: Data glove versus object manipulation for intuitive robot control," in *2014 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, (St. Petersburg), pp. 346–351, IEEE, Oct. 2014.

[14] S. Kitagawa, S. Hasegawa, N. Yamaguchi, K. Okada, and M. Inaba, "Online tangible robot programming: interactive automation method from teleoperation of manipulation task," *Advanced Robotics*, vol. 37, pp. 1063–1081, Aug. 2023.

[15] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation," *arXiv preprint arXiv:2401.02117*, 2024.

[16] J. Norberto Pires, "Robot-by-voice: experiments on commanding an industrial robot using the human voice," *Industrial Robot: An International Journal*, vol. 32, pp. 505–511, Dec. 2005.

[17] J. Tasevski, M. Nikolic, and D. Miskovic, "Integration of an industrial robot with the systems for image and voice recognition," *Serbian Journal of Electrical Engineering*, vol. 10, no. 1, pp. 219–230, 2013.

[18] R. Suzuki, A. Karim, T. Xia, H. Hedayati, and N. Marquardt, "Augmented reality and robotics: A survey and taxonomy for ar-enhanced human-robot interaction and robotic interfaces," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pp. 1–33, 2022.

[19] C. P. Quintero, S. Li, M. K. Pan, W. P. Chan, H. Machiel Van Der Loos, and E. Croft, "Robot Programming Through Augmented Trajectories in Augmented Reality," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Madrid), pp. 1838–1844, IEEE, Oct. 2018.

[20] L. L. Gong, S. K. Ong, and A. Y. C. Nee, "Projection-based Augmented Reality Interface for Robot Grasping Tasks," in *Proceedings of the 2019 4th International Conference on Robotics, Control and Automation*, (Guangzhou China), pp. 100–104, ACM, July 2019.

[21] S. Stadler, K. Kain, M. Giuliani, N. Mirnig, G. Stollnberger, and M. Tscheligi, "Augmented reality for industrial robot programmers: Workload analysis for task-based, augmented reality-supported robot control," in *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, (New York, NY, USA), pp. 179–184, IEEE, Aug. 2016.

[22] K. Zielinski, K. Walas, J. Heredia, and M. B. Kjargaard, "A Study of Cobot Practitioners Needs for Augmented Reality Interfaces in the Context of Current Technologies," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, (Vancouver, BC, Canada), pp. 292–298, IEEE, Aug. 2021.

[23] J. Chen, B. Sun, M. Pollefeys, and H. Blum, "A 3d mixed reality interface for human-robot teaming," *arXiv preprint arXiv:2310.02392*, 2023.

[24] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, *et al.*, "Palm-e: An embodied multimodal language model," *arXiv preprint arXiv:2303.03378*, 2023.

[25] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: program generation for situated robot task planning using large language models," *Autonomous Robots*, vol. 47, no. 8, pp. 999–1012, 2023.

[26] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[27] M. Ahn, D. Dwibedi, C. Finn, M. G. Arenas, K. Gopalakrishnan, K. Hausman, B. Ichter, A. Irpan, N. Joshi, R. Julian, *et al.*, "Autort: Embodied foundation models for large scale orchestration of robotic agents," *arXiv preprint arXiv:2401.12963*, 2024.

[28] B. Chen, Z. Xu, S. Kirmani, B. Ichter, D. Driess, P. Florence, D. Sadigh, L. Guibas, and F. Xia, "Spatialvlm: Endowing vision-language models with spatial reasoning capabilities," *arXiv preprint arXiv:2401.12168*, 2024.

[29] F. De La Torre, C. M. Fang, H. Huang, A. Banburski-Fahey, J. A. Fernandez, and J. Lanier, "Llmr: Real-time prompting of interactive worlds using large language models," *arXiv preprint arXiv:2309.12276*, 2023.

[30] Unity Technologies, "Unity game engine," 2024.

[31] Polycam, "Polycam," 2024.

[32] Universal Robot A/S, "Ur10e," 2024.

[33] Microsoft, "Scene Understanding," 2024.

[34] Microsoft, "Mixed reality mobile remoting," 2023.

[35] U. Technologies, "Unity render streaming," 2019.

[36] K. Mahadevan, J. Chien, N. Brown, Z. Xu, C. Parada, F. Xia, A. Zeng, L. Takayama, and D. Sadigh, "Generative expressive robot behaviors using large language models," *arXiv preprint arXiv:2401.14673*, 2024.

[37] Universal Robot, "Dancing through the pandemic: How a quantum physicist taught a cobot to dance," 2024.

[38] H. Huang, F. De La Torre, C. M. Fang, A. Banburski-Fahey, J. Amores, and J. Lanier, "Real-time animation generation and control on rigged models via large language models," *arXiv preprint arXiv:2310.17838*, 2023.

[39] Microsoft, "Hololens2forcv samples," 2024.

[40] Microsoft, "Qrcode tracking overview," 2024.

[41] OpenAI, "Vision," 2024.

[42] S. Peng, K. Genova, C. M. Jiang, A. Tagliasacchi, M. Pollefeys, and T. Funkhouser, "Openscene: 3d scene understanding with open vocabularies," 2023.

Appendix I
Prompt for the adapted LLMR framework

An example of a modified user prompt that allows the framework to understand the scene and instructions.

```
You are controlling a robot arm named UR10e.
Your goal is to find at least 5 waypoints that
allows the robot end point to move from the TOP
surface of stool 1 to the TOP surface of stool
2. The first and final waypoint should be the
start and end destinations. The middle waypoints
should allow the robot to travel in air but
these waypoints should avoid colliding with all
objects (like table) in the scene. The middle
waypoints should form some sort of arc or curve.
The robot's end point can not reach beyond the
reachability sphere. You should first establish
the mesh boundaries of the objects in the scene.
Do not move the robot. Visualize the final
waypoints as small red sphere with unique names.
```

Appendix II
Example code output of the LLM framework

```
using UnityEngine;

public class WaypointsGenerator : Widgets
{
    public GameObject stool1;
    public GameObject stool2;
    public GameObject table;
    public GameObject reachabilitySphere;
    public GameObject robotEndPoint;
    public GameObject waypointPrefab;
    public int waypointCount = 5;
    private Bounds stool1Bounds;
    private Bounds stool2Bounds;
    private Bounds tableBounds;
    private Bounds reachabilitySphereBounds;

    void Start()
    {
        summary = "This script generates
        ↪  waypoints for the robot end point to
        ↪  move from the top of stool 1 to the
        ↪  top of stool 2, avoiding collision
        ↪  with the table.";

        stool1 = GameObject.Find("Stool_1");
        stool2 = GameObject.Find("Stool_2");
        table = GameObject.Find("Table");
        reachabilitySphere = GameObject.Find("Re⌋
        ↪  achabilitySphere");
        robotEndPoint =
        ↪  GameObject.Find("Robot_EndPoint");

        waypointPrefab = GameObject.CreatePrimit⌋
        ↪  ive(PrimitiveType.Sphere);
        waypointPrefab.name = "WaypointPrefab_"
        ↪  + System.Guid.NewGuid().ToString();
        waypointPrefab.GetComponent<Renderer>().⌋
        ↪  material.color =
        ↪  Color.red;
        waypointPrefab.transform.localScale *=
        ↪  .1f;
        Destroy(waypointPrefab); // We only
        ↪  needed it for the prefab

        stool1Bounds =
        ↪  GetMeshSize.GetRenderBounds(stool1);
        stool2Bounds =
        ↪  GetMeshSize.GetRenderBounds(stool2);
        tableBounds =
        ↪  GetMeshSize.GetRenderBounds(table);
        reachabilitySphereBounds = GetMeshSize.G⌋
        ↪  etRenderBounds(reachabilitySphere);

        GenerateWaypoints();
    }

    void GenerateWaypoints()
    {
        Vector3 start = stool1Bounds.center +
        ↪  new Vector3(0,
        ↪  stool1Bounds.extents.y, 0);
        Vector3 end = stool2Bounds.center + new
        ↪  Vector3(0, stool2Bounds.extents.y,
        ↪  0);

        for (int i = 0; i < waypointCount; i++)
        {
            float t = (float)i / (waypointCount
            ↪  - 1);
            Vector3 waypointPosition =
            ↪  Vector3.Lerp(start, end, t);

            // Make sure the waypoint is above
            ↪  the table and within the
            ↪  reachability sphere
            if (waypointPosition.y <=
            ↪  tableBounds.max.y ||
            ↪  !reachabilitySphereBounds.Contai⌋
            ↪  ns(waypointPosition))
            {
                waypointPosition.y = Mathf.Max(w⌋
                ↪  aypointPosition.y,
                ↪  tableBounds.max.y + 0.1f);
                waypointPosition =
                ↪  reachabilitySphereBounds.Clo⌋
                ↪  sestPoint(waypointPosition);
            }

            // Create the waypoint
            GameObject waypoint =
            ↪  Instantiate(waypointPrefab,
            ↪  waypointPosition,
            ↪  Quaternion.identity);
                Debug.Log(waypointPosition);
            waypoint.name = "Waypoint_" + i;
        }
    }
}
```

Appendix III
Animation prompt

```
You're an animator who will be provided the
joints on a rigged 3D model of a robotic arm,
and you have to rotate the joints to produce the
requested animation. The joints will be given as
a JSON string that outlines the object hierarchy.
You have to output a string that specifies the
rotation of movement for each joint.

Each subsequent line of the string starts with
the joint name, then each vector in the format
(timestamp,rotation), specifing a key frame for
the animation. Rotation is in radians.


# Example:
The object you will animate is a **robot**.
Object JSON:
```

```
name:Robot,children:[name:z-up,children:[name:ro
ot,children:[name:__base,rotation:(0.0),children
:[name:__shoulder,rotation:(0.0),children:[name:
__elbow,rotation:(0.0),children:[name:__wrist-1,
rotation:(0.0),children:[name:__wrist-2,rotation
:(0.0),children:[name:__wrist-3,rotation:(0.0)]]
]]]]]]
__base clockwise direction: (1.0)
__shoulder clockwise direction: (1.0)
__elbow clockwise direction: (-1.0)
__wrist-1 clockwise direction: (1.0)
__wrist-2 clockwise direction: (1.0)
__wrist-3 clockwise direction: (1.0)

Instruction: animate the robot to bow

Robot/z-up/root/__base,(0.000,-1.571),(1.020,-1.
571),(2.020,-1.571),(3.020,-1.571),(4.020,-1.571
),(5.020,-1.571)
Robot/z-up/root/__base/__shoulder,(0.000,-1.571)
,(1.020,-1.571),(2.020,-1.570),(3.020,-1.570),(4
.020,-1.570),(5.020,-1.571)
Robot/z-up/root/__base/__shoulder/__elbow,(0.000
,0.000),(1.020,-0.472),(2.020,-1.519),(3.020,-2.
089),(4.020,-1.358),(5.020,-0.315)
Robot/z-up/root/__base/__shoulder/__elbow/__wris
t-1,(0.000,-1.571),(1.020,-1.239),(2.020,-0.504)
,(3.020,-0.104),(4.020,-0.617),(5.020,-1.349)
Robot/z-up/root/__base/__shoulder/__elbow/__wris
t-1/__wrist-2,(0.000,1.571),(1.020,1.571),(2.020
,1.571),(3.020,1.570),(4.020,1.571),(5.020,1.571)
Robot/z-up/root/__base/__shoulder/__elbow/__wris
t-1/__wrist-2/__wrist3,(0.000,0.000),(1.020,0.00
0),(2.020,0.000),(3.020,0.000),(4.020,0.000),(5.
020,0.000)


# Example:
The object you will animate is a **robot**.
Object JSON:
name:Robot,children:[name:z-up,children:[name:ro
ot,children:[name:__base,rotation:(0.0),children
:[name:__shoulder,rotation:(0.0),children:[name:
__elbow,rotation:(0.0),children:[name:__wrist-1,
rotation:(0.0),children:[name:__wrist-2,rotation
:(0.0),children:[name:__wrist-3,rotation:(0.0)]]
]]]]]]
__base clockwise direction: (1.0)
__shoulder clockwise direction: (1.0)
__elbow clockwise direction: (-1.0)
__wrist-1 clockwise direction: (1.0)
__wrist-2 clockwise direction: (1.0)
__wrist-3 clockwise direction: (1.0)

Instruction: animate the robot shaking as if
it's saying no

Robot/z-up/root/__base,(0.000,-1.514),(1.000,-1.
514),(2.000,-1.514)
Robot/z-up/root/__base/__shoulder,(0.000,-0.931)
,(1.000,-0.932),(2.000,-0.932)
Robot/z-up/root/__base/__shoulder/__elbow,(0.000
,-1.281),(1.000,-1.281),(2.000,-1.281)
Robot/z-up/root/__base/__shoulder/__elbow/__wris
t-1,(0.000,-0.904),(1.000,-0.904),(2.000,-0.904)
Robot/z-up/root/__base/__shoulder/__elbow/__wris
t-1/__wrist-2,(0.000,1.172),(1.000,1.612),(2.000
,1.752)
Robot/z-up/root/__base/__shoulder/__elbow/__wris
t-1/__wrist-2/__wrist3,(0.000,0.000),(1.000,0.00
0),(2.000,0.000)
```

APPENDIX IV

ANIMATION OUTPUT

```
animate the robot saying yes
[0,0,0.5,0,0,0]
[0,0,1.0,0,0,0]
[0,0,0.5,0,0,0]
[0,0,0,0,0,0]
[0,0,0.5,0,0,0]
[0,0,1.0,0,0,0]
[0,0,0.5,0,0,0]
[0,0,0,0,0,0]

animate the robot purring like a cat
[0,0,0,0,0,1]
[0,0,0,0,0,0.5]
[0,0,0,0,0,1]
[0,0,0,0,0,0.5]
[0,0,0,0,0,1]
[0,0,0,0,0,0.5]
[0,0,0,0,0,1]
[0,0,0,0,0,0.5]


the robot is laughing hysterically
[0,0,0,0,0,0]
[0.2,-0.1,0.3,0.1,-0.2,0.3]
[-0.2,0.1,-0.3,-0.1,0.2,-0.3]
[0.2,-0.1,0.3,0.1,-0.2,0.3]
[-0.2,0.1,-0.3,-0.1,0.2,-0.3]
[0.2,-0.1,0.3,0.1,-0.2,0.3]
[-0.2,0.1,-0.3,-0.1,0.2,-0.3]
[0,0,0,0,0,0]


the robot is so disappointed and embarrassed
[0,0,0,0,0,0]
[0.2,-0.5,1.2,-0.8,0,0]
[0,-0.8,1.5,-1.2,0,0]
[0.2,-0.5,1.2,-0.8,0,0]
[0,-0.8,1.5,-1.2,0,0]
[0,0,0,0,0,0]
[0,-0.3,0.5,-0.3,0,0]
[0,0,0,0,0,0]
[0,-0.3,0.5,-0.3,0,0]
[0,0,0,0,0,0]
```