MARCER: Multimodal Augmented Reality for Composing and Executing Robot Tasks

Bryce Ikeda*
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC, USA
bikeda@cs.unc.edu

Maitrey Gramopadhye*
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC, USA
maitrey@cs.unc.edu

LillyAnn Nekervis
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC, USA
lnekervis@unc.edu

Daniel Szafir
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC, USA
daniel.szafir@cs.unc.edu

2025 20th ACM/IEEE International Conference on Human-Robot Interaction (HRI) | 979-8-3503-7893-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/HRI61500.2025.10974232

***Abstract*—In this work, we combine the strengths of humans and robots by developing MARCER, a novel interactive and multimodal end-user robot programming system. MARCER utilizes a Large Language Model to translate users' natural language task descriptions and environmental context into Action Plans for robot execution, based on a trigger-action programming paradigm that facilitates authoring reactive robot behaviors. MARCER also affords interaction via augmented reality to help users parameterize and validate robot programs and provide real-time, visual previews and feedback directly in the context of the robot's operating environment. We present the design, implementation, and evaluation of MARCER to explore the usability of such systems and demonstrate how trigger-action programming, Large Language Models, and augmented reality hold deep-seated synergies that, when combined, empower users to program general-purpose robots to perform everyday tasks.**

***Index Terms*—End-user Robot Programming; Human-Robot Collaboration; Large Language Models; Augmented Reality;**

I. Introduction

Robots excel at simple and repetitive tasks, leading to their widespread adoption within manufacturing and warehouse settings [1]–[3]. However, these situations often require an expert programmer to design and validate robot programs in a structured environment, making it challenging to adapt them to new scenarios. Consequently, in domestic settings, robots are limited to basic, single-purpose systems in non-invasive roles such as vacuum cleaners or lawn mowers [4]–[6]. To expand their role within the home, general-purpose robots must be capable of executing a diverse set of complex and potentially interactive or collaborative tasks while communicating with everyday users to understand and conform to their preferences [7]–[15]. These tasks might range from practical chores such as washing dishes or putting away groceries to social interactions such as greeting people who walk into the home [16]–[18]. In pursuit of this goal, we develop **MARCER**: **M**ultimodal **A**ugmented **R**eality for **C**omposing and **E**xecuting **R**obot Tasks.

*These authors contributed equally to this work.

Fig. 1: MARCER, a multimodal robot programming system that combines natural language and augmented reality to build trigger-action rules. A large language model combines commands such as "Move the groceries on the table to the shelves" with environmental context to generate reactive robot programs. AR allows users to set constraints and preview actions through digital twins, directly in their workspace.

MARCER enables users to build trigger-action programs using natural language, while offering customization and visual feedback via augmented reality. Our system helps bridge the gap between users, who know their own high-level goals, and robots, which possess various low-level capabilities.

MARCER aims to improve robot programming, which can be a cognitively taxing skill for end-users. One of our insights is to leverage Trigger-Action Programming (TAP) to simplify this process, enabling end-users to construct reactive programs without prior coding experience [19]. Through reactive programs, users define trigger-action pairs where, upon meeting the trigger conditions, the related actions are executed. The success of TAP has led the robotics community to begin developing TAP systems that enable non-expert users to craft

reactive social robot behaviors [20] and coordinate robot actions in collaborative tasks [21], [22]. These systems provide initial evidence that end-users with little or no prior experience may quickly learn and successfully apply the TAP paradigm to robot programming.

While promising, prior TAP systems have required users to manually specify TAP rules and Action Plans on a physical interface, such as a handheld tablet or static computer screen. Instead, we propose incorporating Large Language Models (LLMs) to generate rules to be executed on a robot from natural language input. When combined with contextual information from the environment, LLMs have been shown to generate action plans that can be successfully executed by robots [8]–[15], [23], [24]. Building on prior work [15], [23], MARCER leverages environment information to generate robot rules from user speech. Critically, users can also provide verbal feedback, ensuring the generated rules align with their expectations.

We designed MARCER to enhance the programming process by providing visual feedback for *sensemaking*, the process of constructing meaning from information [25]. As robots operate in the real physical world, MARCER provides Augmented Reality (AR) visualizations to convey *in situ* programming information, which can be a more effective way to present robot data compared with 2D interfaces [26], [27] and enhance user abilities in grasping contextual details essential for task completion [28]–[31]. In MARCER, we design and evaluate the first integration of TAP, verbal task specification, and AR visual feedback in a single holistic system.

Contributions: We present MARCER, a novel robot programming system that merges the capabilities of natural language and AR to offer a seamless multimodal experience. We showcase a deployment of MARCER on a Fetch mobile manipulator robot [32] and evaluate our system with 15 participants. Our key contributions are as follows: (1) A system design for multimodal robot programming that integrates verbal commands and AR to interface with users, (2) An exploration into our system's capabilities and usability, and (3) An open source implementation of our system found at https://github.com/hri-ironlab/MARCER.git. We demonstrate how the capabilities of multimodal systems open new possibilities for interactions between people and robots.

II. Related Work

MARCER integrates end-user robot programming, LLM-powered natural language interaction, and mixed reality interfaces to create a multimodal system for human-robot interaction. Below, we discuss relevant work in each of these areas.

A. End-User Robot Programming

End-user robot programming tools help users without coding experience program robots. These tools use various programming paradigms such as block-based [33]–[35], goal-oriented [36], behavior trees [37], [38], or flow-based [39], [40] programming. A shared focus among these approaches is visual programming, where action primitives are presented as nodes that users connect to form action plans. Visual programming abstracts away the complexity of low-level joint control, making it easier for users to create robot programs. Another approach gaining traction is Trigger Action Programming (TAP), commonly used in Internet of Things (IoT) applications like SmartThings, If-This-Then-That, and Zapier. TAP has since expanded into end-user programming for home automation and interactive applications [41]–[46] and recently for personalized social robot behavior [20], and human-robot collaboration [21], [22]. Our work builds on this by combining TAP with AR and linking natural language commands directly to the real world.

B. Large Language Models for Robot Planning

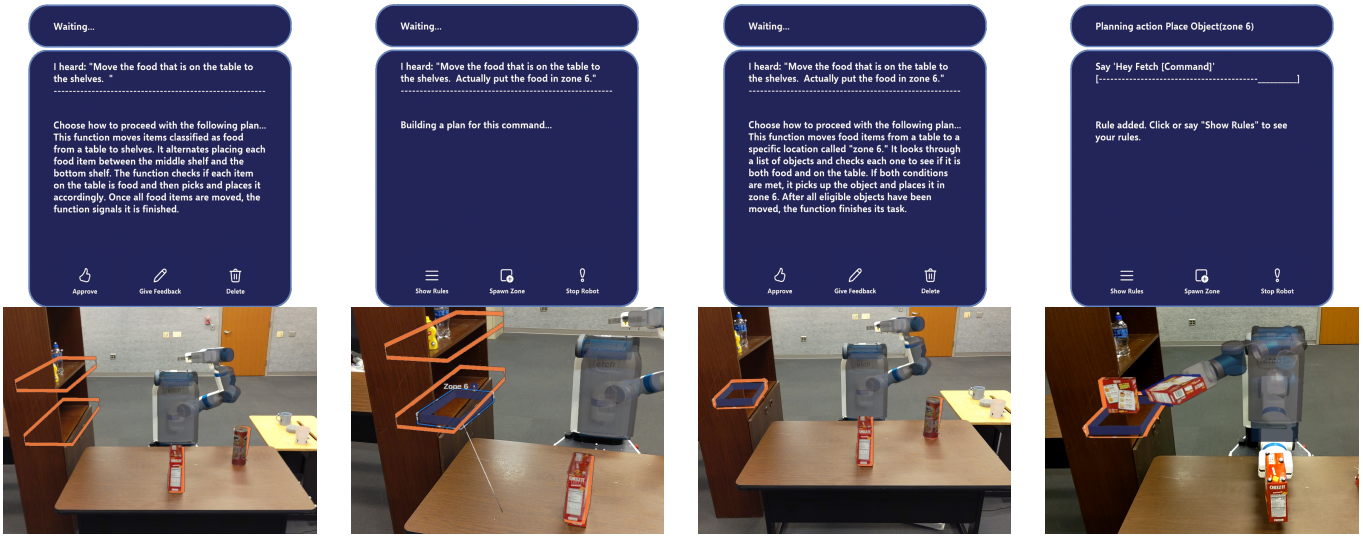
The scientific community is actively investigating how to automatically parse and ground natural language into actionable steps for robots. Large Language Models, which are trained using internet-scale text datasets, appear particularly well-suited to this task and have demonstrated considerable abilities to comprehend diverse concepts and generalize across domains with minimal examples [15], [23], [47]–[57]. MARCER follows a promising approach suggested by recent work where users prompt a LLM with high-level tasks, which, combined with the context of the environment, is used to generate template action plans in terms of known robot behaviors (pre-programmed low-level actions such as pick, place, etc.) [8]–[14]. For instance, Berk Karli et al. 2024 [24] show how approaching robot programming with an LLM may enable users to provide minimally constrained text commands. MARCER extends such systems by enabling interaction via spoken natural language, rather than typed text, which we believe is a more intuitive and hands-free approach to robot programming, and embedding this interaction within a TAP paradigm.

C. AR Programming Feedback

Providing visual feedback to users during programming and debugging is crucial to enhance user experience and aid in identifying and resolving issues. In designing MARCER, we leverage AR to provide this feedback and related contextual information *in situ* within a user's environment. Several modalities for AR exist, ranging from video overlays on traditional computer screens [58]–[60], mobile tablet AR [61]–[64], projector overlays [65]–[67], and augmented reality head mounted displays (ARHMD) [68]–[74]. Recent research has shown that programming tools may utilize AR to improve situational awareness, system usability, and overall user interactions (see [27], [46], [75]–[79] for recent surveys of mixed reality robotics). Therefore, in MARCER we employ an ARHMD to deliver programming feedback in a hands-free experience.

D. Multimodal Robot Programming Interfaces

Prior research has investigated how to provide users with rich sensory information regarding robots and their environments by combining different modalities [80], [81]. For example, prior work has explored interfaces that combine force sensing with vision [82], gaze with voice [83], speech with gestures [84], or natural language text with block-programming [33]. Most similar to our work is that of Marin et al. 2005 [85], Akan



(a) Initial planning phase (b) Adding a location preference (c) Approving the new plan (d) Executing the plan

Fig. 2: As depicted in Fig. 1, a user begins with a command such as, "Move the food that is on the table to the shelves." (a) MARCER then generates an associated trigger-action rule while highlighting relevant objects and surfaces through AR (orange wireframes). In this case, (b) the user provides feedback, specifying preferred placement locations using 3D virtual zones within the workspace (blue rectangles) and verbal feedback ("Actually put the food in zone 6"). (c) The updated rule is presented with visual highlights, and the system awaits the user's approval or further edits. (d) Upon approval, users can preview actions via AR digital twins showing the current, future, and final states of relevant objects, and simulated robot movements.

et al. 2011 [86], and Huang et al. 2019 [87] who implement voice commands and AR to facilitate robot programming. In contrast to these systems, which rely on 2D AR overlays or language commands for single action primitives, our system combines an ARHMD with a LLM to provide 3D spatial input and feedback. This allows users to freely interact with their workspace, while building TAP rules through natural language, enabling users to create programs that address real-world tasks.

III. System Design

MARCER affords multimodal user interaction where users provide input through both natural language (i.e., spoken commands to specify goals, refine plans, and interact with the AR menu) and gestures, which are used to define 3D AR zones relevant to particular robot activities, while also receiving visual AR feedback. Apart from manipulating the 3D zones, users may interact with the interface in a hands-free manner by utilizing verbal input. An example workflow for a direct command is depicted in Fig. 2, while an example for building a trigger-action command is depicted as follows:

1) A user creates a *trigger zone* by saying "spawn zone." When an object or user enters this zone, any associated actions will be executed. Users can create multiple zones, such as one for indicating where a robot can grab objects and another for where to place them. In addition to zones, users can define triggers relevant to locations, time, or both.
2) To generate a *TAP rule*, the user says the key phrase "Hey Fetch," followed by a conditional command such as, "If I place groceries in zone one, move them to the middle shelf." The system then plans a function, comprised of action primitives, for moving the groceries from zone one to the middle shelf while highlighting relevant objects, surfaces, or zones. These highlights help users quickly preview which elements are involved in the plan.
3) Users can *edit* their initial command. For instance, if the user wants the groceries placed in a specific area on the shelf, they can position a second zone on a subsection of the shelf and say, "Place them in zone two instead." This prompts the system to adjust its plan and highlight the newly relevant items. Once satisfied, users can approve the final rule by saying "approve."
4) When the triggering conditions of a stored rule are met, the system *executes* the associated actions. During execution, visualizations display groceries being placed in zone two, including relevant highlights, robot trajectories, and current and future object positions. Users can stop the robot and redo the process at any time by saying "stop robot," or continue to create more rules.

To enable this workflow, our system takes inspiration from prior work in robot programming tools [20], [22], [24], augmented reality robotics interfaces [21], [88]–[90] and LLM research [8]. MARCER is composed of: (1) *Visual Interface*, (2) *Speech Processor*, (3) *Rule Generator*, (4) *Large Language Model*, (5) *Rule Monitor*, (6) *Scene Graph*, (7) *Object Tracker*, (8) *Planning Scene*, (9) *Action Dispatcher*, and (10) *Manipulation Node*. Fig. 3 illustrates these components and is detailed below.

A. Visual Interface and Speech Processor

To facilitate multimodal interaction, users can author robot programs using an AR visual interface (see Fig. 4) and voice

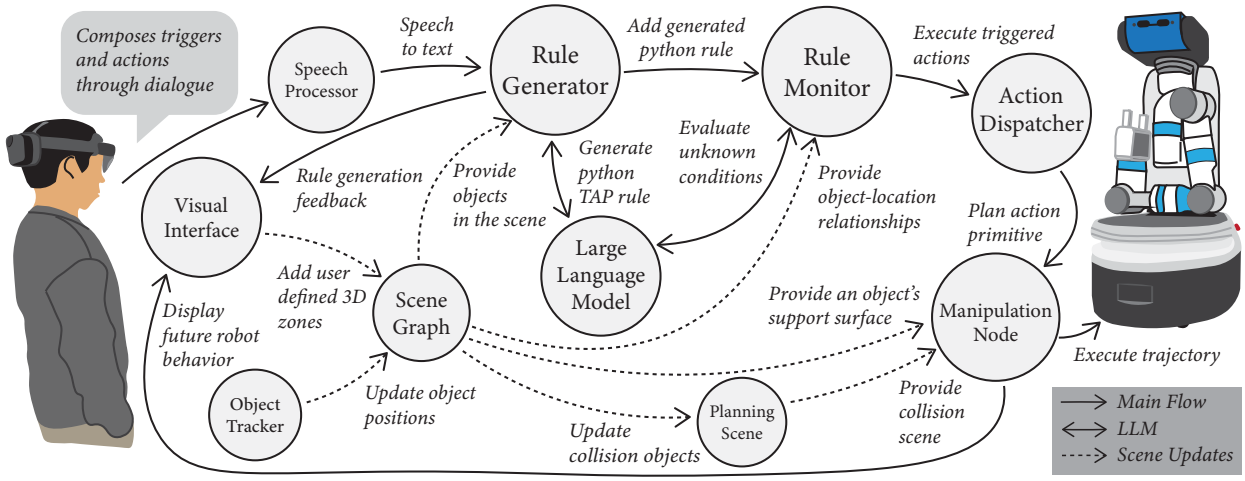


Fig. 3: After a spoken command, the Speech Processor converts it to text, which is sent to the Rule Generator. The Rule Generator collaborates with the LLM to create rules, displays feedback via the Visual Interface, and sends approved rules to the Rule Monitor. The Rule Monitor evaluates each condition, querying the LLM for unknown conditions as needed. The Scene Graph, updated by the Object Tracker, defines relationships between objects, surfaces, users, and zones, sharing this data system-wide. The Action Dispatcher works with the Manipulation Node to plan and execute the Rule Monitor's Action Plan.

commands. The Visual Interface allows users to create, resize, and move 3D zones that specify when to trigger an action or where to place objects. By grounding these zones in the real world, users can specify locations that require depth information such as shelves or object handover locations in space, overcoming the limitations of 2D interfaces that often provide a single top-down view of a workspace. This approach also enables us to overlay the robot's digital twin for visualization of planned motion trajectories. To minimize visual clutter, labels for surfaces, objects, and zones appear only when a user's gaze activates them. Additionally, a menu within the user's workspace displays text information to communicate program states (see Fig. 2). This includes user commands, rule feedback, a description of the generated TAP rule, and the current action being planned or executed. Also listed are active, inactive, and executing rules for user reference and management. Users can operate the visual interface entirely through voice commands, freeing the user's hands for tasks. We see this as a crucial requirement for scenarios that require robot assistance, where a user's hands may already be engaged. For example, the user may be washing dishes, unloading groceries, or folding laundry, while desiring the robot to put away cleaned dishes, grocery items, or folded clothes, analogous to human-human collaborative work. To navigate the interface, users can trigger buttons using voice commands mapped to specific interface menu keywords, such as "spawn zone" for creating a zone or "stop robot" to stop execution immediately in case of unexpected robot actions. To send instructions to the robot, the Speech Processor listens for the wake word, "Hey Fetch," sending the subsequent instruction to the Rule Generator.

B. Rule Generator and Large Language Model

The Rule Generator analyzes verbal commands by querying a LLM in three different stages: Trigger Detector, Function Generator, and Description Generator. Each stage uses prompts constructed from a reference dataset of 54 manually annotated examples containing a user command, its trigger condition, relevant environmental context, and a corresponding Python TAP function (see https://github.com/hri-ironlab/MARCER.git for the dataset and the Appendix for example prompts). While the LLM can operate independently, providing a reference dataset has been found to improve the quality of the generated output [8], [13].

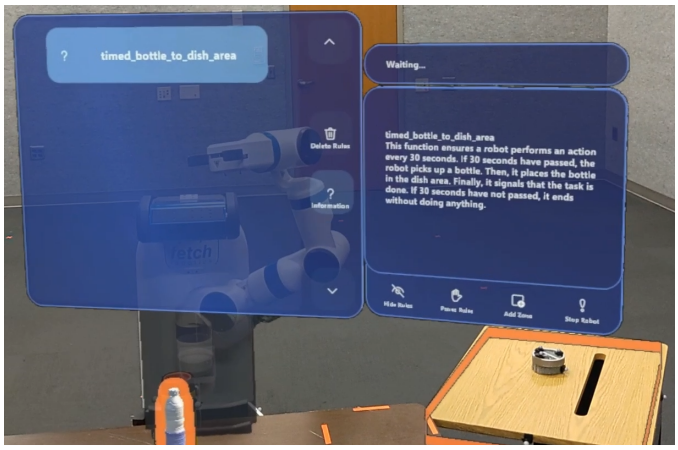


Fig. 4: The visual interface can be placed anywhere in AR. Active rules appear on the left panel, with executing rules highlighted. Users can also view descriptions of each rule. Objects and locations are highlighted orange during execution.

Trigger Detector: In the first stage, the Trigger Detector queries the LLM with a prompt containing both the user's spoken command and a set of example command-trigger pairs to identify the intended trigger type. Currently,

MARCER supports four types of triggers, a *location trigger* of the kind "If an object/person is in `[location]`, then `[desired_action]`" (e.g., "If a dish is on the table, put it in the sink" or "When I get back, if I enter Zone 1, hand me food from the shelf"), a *timed trigger* of the format "If `[desired_time]` has passed, `[desired_action]`" (e.g., "In 10 minutes, pour me a glass of water" or "In 1 hour, put the food on the table"), a *timed location trigger* of the format "If an object/person has been in `[location]` for `[desired_time]`, `[desired_action]`" (e.g. "Whenever any object has been in Zone 1 for 5 minutes, move it to the middle shelf" or "If the glass has been sitting on the table for 30 seconds, pour some water in it"), and a *null trigger* for handling immediately executable commands, such as "move the mustard from the table to the shelf." MARCER does require each command to begin with the keywords "Hey Fetch" followed by a conditional statement or a direct command. However, rather than forcing users to adhere to a strict conditional format, MARCER seeks to match what was said to one of our four trigger types. This provides users with the freedom to verbally specify their commands however they like. After inferring the trigger type, this information is sent to the Function Generator.

Function Generator: The Function Generator queries the LLM to generate a Python function that represents the user's command as a trigger-action rule. By directly producing and executing Python code at runtime, MARCER eliminates the need to translate task descriptions into executable code. To enable this, the query utilizes a prompt constructed with examples tailored to the specific trigger type from the dataset. In addition, the Scene Graph (see §III-E) provides relevant names and locations of objects and surfaces within the scene. This information is added to the prompt along with any corrective feedback the user may have provided to fix prior generated rules. To evaluate triggers (e.g., "If a dish is on the table, put it in the sink" has the trigger "dish is on the table", or "Put all food on the table" should execute for all objects of type "food") and dynamically changing environments, we designed a primitive function `check_condition` that can be included in the generated function. When executed by the Rule Monitor (see §III-C), if it returns true, the associated actions are passed to the Action Dispatcher (see §III-D). In case of an unknown condition, the function queries the LLM for a truth value.

Description Generator: The Description Generator queries the LLM to summarize the output Python function in plain language that non-programmers can understand. Users can then review this summary and decide if it matches their expectations. If not, they can provide verbal feedback for recomputing the plan. MARCER incorporates each round of feedback until the user is satisfied with the generated function output, which, once approved, is forwarded to the Rule Monitor.

C. Rule Monitor

MARCER currently supports the *If-Then* TAP rule paradigm, with potential for future expansion. An If-Then rule triggers an action once its **If** condition is met. For example, "**If** an object is in [Zone 1], **then** [move all objects from Zone 1 to the shelf]" or "If a dish is on the dish area for [10 minutes], move it to the bottom shelf", which MARCER formulates as "**If** a dish is on the [dish area] and [10 minutes] have passed, **then** [move it to the bottom shelf]". The *Location Triggers* are based on "surfaces," like tables or shelves in the environment, and user-defined "zones" that can exist anywhere in 3D space (for example, a user might create a midair zone for object handovers). To evaluate location conditionals, the Rule Monitor queries the Scene Graph to check whether objects satisfy an "is on" or "is in" relationship. For *Timed Triggers*, the Rule Monitor stores a new timestamp when a rule is initially added to the monitoring list, or when an object moves into a time-restricted location. During each conditional check, if the elapsed time with respect to the initial stored timestamp exceeds the specified threshold in the rule, the Rule Monitor triggers the associated actions. Users can create, edit, or delete rules when the robot is ready for commands, during rule setup, or while providing verbal feedback.

D. Action Dispatcher

When a rule trigger evaluates to true, the Action Dispatcher executes its associated actions by coordinating with the Manipulation Node (§III-F). This includes managing the planning and execution of actions or activating fallback measures when necessary. For example, if execution fails because sensed joint positions deviate from the motion plan's tolerance, the system replans from the current robot state and retries the action. The Action Dispatcher continues to broadcast state updates and feedback, enabling other components to monitor the progress of manipulation tasks. Once all actions for a TAP rule succeed, the next triggered rule can be executed.

E. Scene Graph and Object Tracker

The Scene Graph and Object Tracker are used to compute the positions and relationships of objects, surfaces, and zones in the scene. These object poses, along with zone poses and dimensions from the visual interface, are fed into the Scene Graph. Scene graphs are commonly utilized in robotics for creating a shared 3D world model to be queried by different components in a robotic system. Our Scene Graph component currently consists of *nodes* representing objects, zones, and surfaces, each storing *attributes* such as position, orientation, and size. After each update loop, the Scene Graph computes its *edges*, representing relationships between nodes. Currently supported are the "is on" relationships between an object and a surface and the "is in" relationships between an object and a zone. This relationship is utilized by the Rule Generator for building TAP rules relevant to the scene, Rule Monitor for checking conditions, and the Manipulation Node to calculate allowable surface collisions during pick-and-place operations.

F. Manipulation Node and Planning Scene

MARCER creates functions composed of known primitive actions that, when combined, can accomplish a variety of household tasks. Currently, MARCER supports six primitive actions: pick, place, pour, wipe surface, wave, and dance. To

execute these actions, the Action Dispatcher sends the current action to the Manipulation Node, which then coordinates with the Planning Scene. Upon receiving a manipulation plan request, the Planning Scene takes a snapshot of the world objects, surfaces, and robot pose by querying the Scene Graph. From this snapshot, the Manipulation Node computes a collision-free motion plan, which is displayed via the Visual Interface. The robot executes the plan through its joint controller.

IV. System Implementation

Robot Platform and Object Tracking: MARCER currently works with the Fetch robot, a 7 degree-of-freedom mobile manipulator [32]. Within the Manipulation Node, MARCER utilizes the MoveIt! Task Constructor (MTC) to plan and execute manipulation actions [91]. MARCER's motion planning time averaged 7.62 seconds/plan (SD = 1.50 sec.). One benefit to MTC, is its compatibility with over 150 robot platforms using MoveIt! Thus, MARCER can be adapted to other robots by swapping the MoveIt! configuration profiles provided by other platforms. To track objects within the scene, we use a Vicon motion capture system as robust perception was not the focus on this work; future systems could leverage the robot and ARHMD sensors to replace external tracking. By placing reflective markers on the robot and scene objects, we can track the position and orientation of objects in the scene relative to the robot. This information is passed to the Scene Graph to be read by other components. Currently, our setup (depicted in Fig. 5a), includes four surfaces, the *middle shelf*, *bottom shelf*, *dish area*, and *table* and a combination of common household items including food, drinkware, and a sponge.

AR Interface: For our visual interface, we use the Microsoft HoloLens 2 ARHMD. We use the Unity game engine and the Mixed Reality Toolkit to build and stream holograms to the ARHMD [92]. To align the holograms and the real world in the AR camera space, we follow a two-step process. First, we match the origin of a fiducial marker tracked by the AR headset [93] with a corresponding point in the Vicon tracking space. Then, the positions of the tracked objects and the robot are transformed into the fiducial marker's coordinate system, aligning their virtual representations in the HoloLens. With this approach, we are also able to translate the position and orientation of virtual zones to the real world.

Speech Processing and Large Language Model: To enable natural language input, our speech processor utilizes the Windows Keyword Recognition Subsystem, PyAudio, and the Whisper speech recognition model [94]. To navigate the AR interface, users can trigger buttons using voice commands mapped to specific keywords. To recognize commands to send to the Rule Generator, the Speech Processor listens for the key phrase, "Hey Fetch." Once recognized, the instruction following the key phrase is sent to the Rule Generator. We implement our Large Language Model using the `GPT-4o` [57], [95] model offered by the OpenAI API [96].

Hardware and Communication: Our system operates on two desktops equipped with an NVIDIA RTX 3080 GPU. One of the machines has a Windows 11 operating system, and runs the Vicon tracking software and our Unity application. Vicon tracking data is transmitted to the second desktop via a Python socket. The second desktop has a Ubuntu 20.04 operating system, and hosts the back-end of our programming system on Robot Operating System (ROS), a framework designed for developing robot software. The ROS TCP Endpoint and Connector [97] connect Unity to ROS, enabling accurate visualizations of the robot and object states.

V. System Evaluation

We conducted an IRB-approved study to explore how users interact with MARCER. Participants completed three robot-assisted tasks modeled after everyday scenarios: (1) Kitchen Cleanup, (2) Item Storing, and (3) Object Handover. These tasks involve defining object placements, triggers for actions, and creating combinations of actions, and as such we believe MARCER may also apply to more specialized tasks such as object assembly [21], [22], [98] or chemistry experiments [24], which require similar specifications.

Task 1. Kitchen Cleanup: (20 minute cap) Participants programmed the robot to clean the kitchen area (see Fig. 5b). To be considered clean, objects needed to remain in the dish area for 15 seconds (mimicking being rinsed), then be placed on the bottom shelf. Users were told to build a repetitive rule for this task to handle multiple dishes. Next, users programmed the robot to clean the bottle, pour the contents of the glass into the cup, and have the robot clean the glass. After the user moved the cup to the side table and the main table was empty, the robot needed to wipe the main table with the sponge.

Task 2. Item Storing: (20 minute cap) Participants programmed the robot to move the objects from the table to the shelves. While the glass and cup could be placed anywhere on the bottom shelf, participants needed to define a food storage area by placing a zone in the outlined space on the middle shelf. See Fig. 5c for the initial setup of the scene.

Task 3. Object Handover: (10 minute cap) Participants programmed the robot to trigger a handover when an object was held in a particular zone in 3D space. Once programmed, the user held the object in the zone to complete the handover.

A. Procedure

Participants first read and signed a consent form. An examiner then guided them through a 30 minute training script, which included creating a direct command, a one-time rule that executed and deleted itself, and a repetitive rule that continued to check for execution. The examiner explained the visualizations, how to provide verbal feedback, how to delete rules, and how to view active rules and their descriptions, answering any questions along the way. After successfully completing the tutorials, participants read over the first task, Kitchen Cleanup, while the experimenters set up the scene. The task began when participants made their first interaction with the interface. Once the task was completed or time ran out, the task was ended. This procedure was repeated for Task 2 and Task 3. After Task 3, participants completed a survey that included the System Usability Scale (SUS) [99],

(a) Workspace (b) Task 1: Kitchen Cleanup (c) Task 2: Item Storing (d) Task 3: Object Handover

Fig. 5: The robot can move (a) the Scene Objects between the "Middle Shelf", "Bottom Shelf", "Dish Area", and "Table." In Task 1 (b), users programmed the robot to clean the dishes by moving them to the "Dish Area" for 15 seconds, placing them on the bottom shelf, and wiping the table. In Task 2 (c), users defined a virtual zone on the left side of the "Middle Shelf" and instructed the robot to move food to that zone and place a glass and cup anywhere on the bottom shelf. In Task 3 (d), users placed a virtual zone in 3D space for triggering a handover, prompting the robot to grab the object from the user's hand.

demographic information, and subjective feedback, followed by a semi-structured interview. The training script, task sheets, survey, and interview guide are included in our git repository.

B. Measures

We collected a series of measurements to characterize our system using data collected from participants. These include task completion time, total time spent programming, the number of times users created a one-time command or repetitive rule, gave verbal feedback, or rejected rules, and types of errors encountered. We categorized the errors into five types: **T**: Vicon tracking error, **R**: Robot manipulation error, **L**: LLM logic error/hallucination, **H**: Headset error, and **M**: Microphone error. Please see the Appendix for more details.

C. Participants

For this study we recruited 15 participants (9 male, 6 female) with ages ranging from 18–50 (M = 25.6, SD = 7.67). Thirteen (86.6%) participants reported having three or more years of programming experience and eight (53.3%) participants reported owning an IoT device. On a single item 1–7 scale (7 = most familiar), participants reported their average familiarity with robotics as 3.2 (SD = 2.11), VR/AR technologies as 4.1 (SD = 1.81), and TAP as 5.8 (SD = 1.42).

D. Findings

Table I summarizes the objective data from our system evaluation. Ten (66.67%) participants completed all three tasks, while two (13.33%) did not finish the first task, and three (20%) did not finish the second within the time limits. The two who failed Task 1 commanded the robot to clean the glass with a sponge instead of following the instructions of leaving it in the dish area for 15 seconds. Although valid in the real world, this method exceeded the robot's capabilities, preventing it from planning and executing the action. In Task 2, two participants placed an object on the table in a position the robot could not reach, leading it to continuously fail to plan a trajectory to pick it up. The third participant created a rule that moved all food objects, even those already on the shelf, to the food storage area. This left no space for the final object, and the participant did not debug the issue in time.

The participants' programming time for all tasks (M = 5:36, SD = 4:07) including the LLM plan generation time (M = 4.69 sec./query, SD = 2.41 sec.) contributed to just 27% of the total experiment time (M = 20:37, SD = 8:21). Surveys showed that users found creating conditional rules straightforward, with both location-based triggers (M = 6.4, SD = 1.30) and time-based triggers (M = 6.4, SD = 0.91) rated to be easy to create. Users often opted to set rules using the triggers, as pointed out by **P11:** *"The autonomous repetitive task feature made it very easy. For the second scenario it was quite easy to automate the entire process without needing to re-command"* and **P14:** *"Having repeated rules made it easier so I did not have to say multiple commands"*. When asked about the rule editing mechanism, **P15** noted that *"[giving feedback] definitely helped a lot when the task given in the first place was wrong and it made changing statements easier. It also saved time in a way."* Along the same lines, **P9** mentioned that *"The feedback system helped a lot because I made a lot of mistakes and I wanted to edit things."*

Users also agreed that the interaction with the robot was fluent (M = 5.13, SD = 1.73) and became more fluent over time (M = 6.07, SD = 1.10), as noted by **P6:** *"At first I felt adversarial to the robot, but by the end we were a team."* User comments such as **P4:** *"seeing what the robot was gonna do before it actually performed the action made the whole thing really friendly,"* **P1:** *"I think like I didn't really struggle with making the rules because I just said it how I thought I would naturally say it in general, and the response it gave me was pretty intuitive and matched what I was looking for,"* and **P8:** *"Having the robot explain the thought process and actions it was going to take before actually taking them [made it easier]"* point to MARCER's integration of the AR previews, unconstrained speech commands, and natural language plan description being particularly useful. The system received a SUS score of 70.83 (SD = 16.39), indicating "above average" usability [100].

VI. Discussion

MARCER demonstrates the design and development of a state-of-the-art robot programming interface that provides

TABLE I: Metrics obtained include task completion time and whether they did not finish (DNF) in the allotted time, total time spent programming, number of rules created for the task, number of times feedback was provided, number of times a rule was rejected, and the System Usability Score. Errors encountered include **T**: Vicon Tracking error, **R**: Robot manipulation error, **L**: LLM logic error/hallucination, **H**: Headset error, and **M**: Microphone error (see the Appendix for more details).

	Task 1: Kitchen Cleanup					Task 2: Item Storage					Task 3: Object Handover						
PID	Total Time	Prog. Time	# Rules	# Feed.	# Rej.	Total Time	Prog. Time	# Rules	# Feed.	# Rej.	Total Time	Prog. Time	# Rules	# Feed.	# Rej.	Errors	SUS
P1	7:58	1:22	3	0	0	9:05	2:48	2	1	0	2:51	0:16	1	0	0	T, R, L	72.5
P2	11:27	2:51	5	0	0	14:52	4:25	2	3	0	2:33	0:26	1	0	0	-	70
P3	9:34	1:41	4	0	0	DNF	2:47	8	0	0	1:55	0:17	1	0	0	T, L	87.5
P4	9:50	1:34	4	0	0	8:44	0:50	2	0	0	2:22	0:22	1	0	0	H	70
P5	8:37	1:59	3	0	0	DNF	5:05	2	2	3	5:02	2:33	1	0	2	T, L, R	40
P6	9:47	0:57	3	1	0	6:46	0:43	4	0	0	1:54	0:14	1	0	0	-	82.5
P7	15:43	5:20	4	0	0	15:45	2:49	4	0	2	4:14	2:06	1	1	1	T, L	80
P8	9:57	2:05	4	3	1	9:41	0:37	2	0	0	2:13	0:13	1	0	0	-	92.5
P9	7:44	2:01	3	0	0	DNF	4:19	8	0	4	3:01	0:39	1	0	1	L, R	67.5
P10	DNF	7:55	8	1	0	9:24	1:01	3	0	0	2:33	0:44	1	1	0	L, R	35
P11	10:35	1:51	4	2	0	10:27	0:36	2	0	0	2:15	0:15	1	0	0	-	70
P12	10:34	1:00	4	0	0	18:26	2:08	5	0	0	2:35	0:23	1	0	0	L, R	60
P13	8:36	1:32	4	0	0	13:42	2:32	5	0	1	2:54	0:20	1	0	0	L, R	67.5
P14	8:17	1:07	4	0	0	9:49	0:52	2	0	0	2:18	0:20	1	0	0	H	77.5
P15	DNF	3:56	7	1	0	11:56	3:11	5	1	0	3:20	0:29	1	0	0	L, M	90
MEAN	9:54	2:29	4.27	0.53	0.07	11:33	2:19	3.73	0.47	0.67	2:48	0:38	1	0.13	0.27		70.83

cohesive, multimodal user interaction. From participant interactions, one striking observation was the diverse ways in which participants approached their programming tasks. Users were able to verbally direct the robot in flexible, natural ways, without being forced to adhere to the strict formatting and syntax required by traditional programming systems. These commands ranged from statements like **P8**, "The user will hand you a can of pringles in zone 1, then move it to the lower shelf," to **P10** "Once an object is in dish area for 15 seconds move it to the bottom shelf" and **P11**, "If you see an object on the table move it to zone 1." Our evaluation found that participants were generally effective using MARCER to complete the three tasks. Across all trials, only two (4.44%) tasks were failed due to invalid LLM generated plans that participants could not recover from. However, we observed that MARCER fell short in communicating system failures effectively. Therefore, future systems should not only communicate their capabilities, but also why specific actions fail. Still, MARCER's high success rate provides strong support for the power of combining the simple and intuitive structure of TAP with the translation capabilities of LLMs, and we argue that future systems should **empower users to communicate programs their own way**.

A. Limitations and Future Work

World State Estimation: One limitation of MARCER is its reliance on a Vicon motion capture system for object tracking. One way to address this is to integrate a perception pipeline that combines data from the robot and ARHMD sensors to continuously estimate a model of the real world while incorporating advanced Vision-Language Models (VLMs).

TAP Expressions: Further extensions might also utilize more of the expressive power of TAP for robotics. For example, Huang and Cakmak [101], highlight nine trigger-action programming pairs that match a user's mental model. These pairs, such as *While-Do*, *As-Long-As-Do*, and *If-When-Then*, could extend our set and improve user interaction.

Finally, future work can conduct further evaluations, including against other robot programming systems and with larger and more representative samples of novice users to better understand trade-offs and produce more generalizable results.

VII. Conclusion

This paper introduces MARCER, a multimodal system for composing and refining robot trigger-action rules for everyday tasks. MARCER integrates natural language processing with an Augmented Reality Head-Mounted Display (ARHMD) to provide hands-free interaction and visual feedback within the context of robot activities. We demonstrate how MARCER enables users to set up various home-assistance tasks and characterize performance in a system evaluation. By combining the expressive power of trigger-action programming, natural language verbal input, and augmented reality visual feedback, we pave the way for seamless integration of general-purpose robot assistants in the home.

VIII. Acknowledgments

This work was supported by NSF Award #2222953.

REFERENCES

[1] Abderraouf Maoudj Abdelfetah Hentout, Mustapha Aouache and Isma Akli. Human–robot interaction in industrial collaborative robotics: a literature review of the decade 2008–2017. *Advanced Robotics*, 33(15-16):764–799, 2019.

[2] Eloise Matheson, Riccardo Minto, Emanuele G. G. Zampieri, Maurizio Faccio, and Giulio Rosati. Human–robot collaboration in manufacturing applications: A review. *Robotics*, 8(4), 2019.

[3] S. Robla-Gómez, Victor M. Becerra, J. R. Llata, E. González-Sarabia, C. Torre-Ferrero, and J. Pérez-Oria. Working together: A review on safe human-robot collaboration in industrial environments. *IEEE Access*, 5:26754–26773, 2017.

[4] M. Scopelliti, M. V. Giuliani, A. M. D'Amico, and F. Fornara. If i had a robot at home... peoples' representation of domestic robots. In Simeon Keates, John Clarkson, Patrick Langdon, and Peter Robinson, editors, *Designing a More Inclusive World*, pages 257–266, London, 2004. Springer London.

[5] Joe Saunders, Dag Sverre Syrdal, Kheng Lee Koay, Nathan Burke, and Kerstin Dautenhahn. "teach me–show me"—end-user personalization of a smart home and companion robot. *IEEE Transactions on Human-Machine Systems*, 46(1):27–40, 2016.

[6] Garrett Wilson, Christopher Pereyda, Nisha Raghunath, Gabriel de la Cruz, Shivam Goel, Sepehr Nesaei, Bryan Minor, Maureen Schmitter-Edgecombe, Matthew E. Taylor, and Diane J. Cook. Robot-enabled support of daily activities in smart home environments. *Cognitive Systems Research*, 54:258–272, 2019.

[7] Abhinav Gupta, Adithyavairavan Murali, Dhiraj Prakashchand Gandhi, and Lerrel Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[8] Maitrey Gramopadhye and Daniel Szafir. Generating executable action plans with environmentally-aware language models. in 2023 ieee. In *RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3568–3575.

[9] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[10] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-e: An embodied multimodal language model. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8469–8488. PMLR, 23–29 Jul 2023.

[11] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[12] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[13] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR, 2022.

[14] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023.

[15] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):1–26, 2024.

[16] Dhruv Batra, Angel X Chang, Sonia Chernova, Andrew J Davison, Jia Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, et al. Rearrangement: A challenge for embodied ai. *arXiv preprint arXiv:2011.01975*, 2020.

[17] David Porfirio, Allison Sauppé, Aws Albarghouthi, and Bilge Mutlu. Authoring and verifying human-robot interactions. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, page 75–86, New York, NY, USA, 2018. Association for Computing Machinery.

[18] Enrique Coronado, Fulvio Mastrogiovanni, Bipin Indurkhya, and Gentiane Venture. Visual programming environments for end-user development of intelligent and social robots, a systematic review. *Journal of Computer Languages*, 58:100970, 2020.

[19] Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L Littman. Practical trigger-action programming in the smart home. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 803–812, 2014.

[20] Nicola Leonardi, Marco Manca, Fabio Paternò, and Carmen Santoro. Trigger-action programming for personalising humanoid robot behaviour. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–13, New York, NY, USA, 2019. Association for Computing Machinery.

[21] Bryce Ikeda and Daniel Szafir. Programar: Augmented reality end-user robot programming. *J. Hum.-Robot Interact.*, 13(1), mar 2024.

[22] Emmanuel Senft, Michael Hagenow, Robert Radwin, Michael Zinn, Michael Gleicher, and Bilge Mutlu. Situated live programming for human-robot collaboration. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 613–625, 2021.

[23] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

[24] Ulas Berk Karli, Juo-Tung Chen, Victor Nikhil Antony, and Chien-Ming Huang. Alchemist: Llm-aided end-user development of robot applications. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, pages 361–370, 2024.

[25] Peter Pirolli and Stuart Card. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of international conference on intelligence analysis*, volume 5, pages 2–4. McLean, VA, USA, 2005.

[26] Bryce Ikeda and Daniel Szafir. Advancing the design of visual debugging tools for roboticists. In *Proceedings of the 2022 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '22, page 195–204. IEEE Press, 2022.

[27] Michael Walker, Thao Phung, Tathagata Chakraborti, Tom Williams, and Daniel Szafir. Virtual, augmented, and mixed reality for human-robot interaction: A survey and virtual design element taxonomy. *ACM Transactions on Human-Robot Interaction*, 12(4):1–39, 2023.

[28] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, 21(6):34–47, 2001.

[29] Hooman Hedayati, Michael Walker, and Daniel Szafir. Improving collocated robot teleoperation with augmented reality. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '18, page 78–86, New York, NY, USA, 2018. Association for Computing Machinery.

[30] John P McIntire, Paul R Havig, and Eric E Geiselman. What is 3d good for? a review of human performance on stereoscopic 3d displays. In *Head-and Helmet-Mounted Displays XVII; and Display Technologies and Applications for Defense, Security, and Avionics VI*, volume 8383, page 83830X. International Society for Optics and Photonics, 2012.

[31] Eric Rosen, David Whitney, Elizabeth Phillips, Gary Chien, James Tompkin, George Konidaris, and Stefanie Tellex. Communicating robot arm motion intent through mixed reality head-mounted displays. In Nancy M. Amato, Greg Hager, Shawna Thomas, and Miguel Torres-Torriti, editors, *Robotics Research*, pages 301–316, Cham, 2020. Springer International Publishing.

[32] Melonee Wise, Michael Ferguson, Derek King, Eric Diehr, and David Dymesich. Fetch and freight: Standard platforms for service robot applications. In *Workshop on autonomous mobile service robots*, pages 1–6, 2016.

[33] Sara Beschi, Daniela Fogli, and Fabio Tampalini. Capirci: a multi-modal system for collaborative robot programming. In *End-User Development: 7th International Symposium, IS-EUD 2019, Hatfield, UK, July 10–12, 2019, Proceedings 7*, pages 51–66. Springer, 2019.

[34] Justin Huang and Maya Cakmak. Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 453–462, 2017.

[35] Justin Huang, Tessa Lau, and Maya Cakmak. Design and evaluation of a rapid programming system for service robots. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 295–302. IEEE, 2016.

[36] David Porfirio, Mark Roberts, and Laura M. Hiatt. Goal-oriented end-user programming of robots. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '24, page 582–591, New York, NY, USA, 2024. Association for Computing Machinery.

[37] Kelleher R Guerin, Colin Lea, Chris Paxton, and Gregory D Hager. A framework for end-user instruction of a robot assistant for manufacturing. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 6167–6174. IEEE, 2015.

[38] Chris Paxton, Felix Jonathan, Andrew Hundt, Bilge Mutlu, and Gregory D Hager. Evaluating methods for end-user creation of robot task plans. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6086–6092. IEEE, 2018.

[39] Emilia I Barakova, Jan CC Gillesen, Bibi EBM Huskens, and Tino Lourens. End-user programming architecture facilitates the uptake of robots in social therapies. *Robotics and Autonomous Systems*, 61(7):704–713, 2013.

[40] Floris Erich, Masakazu Hirokawa, and Kenji Suzuki. A visual environment for reactive robot programming of macro-level behaviors. In *Social Robotics: 9th International Conference, ICSR 2017, Tsukuba, Japan, November 22-24, 2017, Proceedings 9*, pages 577–586. Springer, 2017.

[41] Manuel García-Herranz, Pablo A. Haya, and Xavier Alamán. Towards a ubiquitous end-user programming system for smart spaces. *J. Univers. Comput. Sci.*, 16:1633–1649, 2010.

[42] Anind K. Dey, Timothy Sohn, Sara Streng, and Justin Kodama. icap: Interactive prototyping of context-aware applications. In Kenneth P. Fishkin, Bernt Schiele, Paddy Nixon, and Aaron Quigley, editors, *Pervasive Computing*, pages 254–271, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[43] Yngve Dahl and Reidar-Martin Svendsen. End-user composition interfaces for smart environments: A preliminary study of usability factors. In Aaron Marcus, editor, *Design, User Experience, and Usability. Theory, Methods, Tools and Practice*, pages 118–127, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[44] Julia Brich, Marcel Walch, Michael Rietzler, Michael Weber, and Florian Schaub. Exploring end user programming needs in home automation. *ACM Trans. Comput.-Hum. Interact.*, 24(2), apr 2017.

[45] Tianyi Wang, Xun Qian, Fengming He, Xiyun Hu, Yuanzhi Cao, and Karthik Ramani. Gesturar: An authoring system for creating freehand interactive augmented reality applications. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, UIST '21, page 552–567, New York, NY, USA, 2021. Association for Computing Machinery.

[46] Tianyi Wang, Xun Qian, Fengming He, Xiyun Hu, Ke Huo, Yuanzhi Cao, and Karthik Ramani. Capturar: An augmented reality tool for authoring human-involved context-aware applications. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pages 328–341, 2020.

[47] Joe Davison, Joshua Feldman, and Alexander Rush. Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1173–1178, Hong Kong, China, November 2019. Association for Computational Linguistics.

[48] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *CoRR*, abs/1911.12543, 2019.

[49] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. Language Models as Knowledge Bases? *ArXiv*, abs/1909.01066, 2019.

[50] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[51] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*, 2019.

[52] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. 2019.

[53] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

[54] Belinda Z. Li, Maxwell Nye, and Jacob Andreas. Implicit Representations of Meaning in Neural Language Models. In *ACL*, 2021.

[55] Adam Roberts, Colin Raffel, and Noam M. Shazeer. How Much Knowledge Can You Pack into the Parameters of a Language Model? *ArXiv*, abs/2002.08910, 2020.

[56] Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems*, 32(2):604–624, 2020.

[57] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[58] Alexandros Rotsidis, Andreas Theodorou, Joanna J Bryson, and Robert H Wortham. Improving robot transparency: An investigation with mobile augmented reality. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1–8. IEEE, 2019.

[59] Sunao Hashimoto, Akihiko Ishida, Masahiko Inami, and Takeo Igarashi. Touchme: An augmented reality based remote robot manipulation. In *The 21st International Conference on Artificial Reality and Telexistence, Proceedings of ICAT2011*, volume 2, 2011.

[60] Hyoungnyoun Kim, Jun-Sik Kim, Kwanghyun Ryu, Seyoung Cheon, Yonghwan Oh, and Ji-Hyung Park. Task-oriented teleoperation through natural 3d user interaction. In *2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 335–338. IEEE, 2014.

[61] E Schmidt, S Schweizer, and D Hendrich. Augmented reality robot operation interface with google tango, 2018.

[62] Linfeng Chen, Akiyuki Ebi, Kazuki Takashima, Kazuyuki Fujita, and Yoshifumi Kitamura. Pinpointfly: An egocentric position-pointing drone interface using mobile ar. In *SIGGRAPH Asia 2019 Emerging Technologies*, pages 34–35. 2019.

[63] Jared Alan Frank, Sai Prasanth Krishnamoorthy, and Vikram Kapila. Toward mobile mixed-reality interaction with multi-robot systems. *IEEE Robotics and Automation Letters*, 2(4):1901–1908, 2017.

[64] Jens Lambrecht and Jörg Krüger. Spatial programming for industrial robots based on gestures and augmented reality. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 466–472. IEEE, 2012.

[65] Seungho Chae, Hyocheol Ro, Yoonsik Yang, and Tack-Don Han. A pervasive assistive robot system including projection-camera technology for older adults. In *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 83–84, 2018.

[66] Ramsundar Kalpagam Ganesan, Yash K Rathore, Heather M Ross, and Heni Ben Amor. Better teaming through visual cues: how projecting imagery in a workspace can improve human-robot collaboration. *IEEE Robotics & Automation Magazine*, 25(2):59–71, 2018.

[67] Fabrizio Lamberti, Davide Calandra, Federica Bazzano, Filippo G Prattico, and Davide M Destefanis. Robotquest: A robotic game based on projected mixed reality and proximity interaction. In *2018 IEEE Games, Entertainment, Media Conference (GEM)*, pages 1–9. IEEE, 2018.

[68] Nhan Tran, Trevor Grant, Thao Phung, Leanne Hirshfield, Christopher Wickens, and Tom Williams. Get this!? mixed reality improves robot communication regardless of mental workload. In *Companion of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, pages 412–416, 2021.

[69] Stephanie Arevalo Arboleda, Franziska Rücker, Tim Dierks, and Jens Gerken. Assisting manipulation and grasping in robot teleoperation with augmented reality visual cues. In *Proceedings of the 2021 CHI conference on human factors in computing systems*, pages 1–14, 2021.

[70] Yi-Shiuan Tung, Matthew B. Luebbers, Alessandro Roncone, and Bradley Hayes. Workspace optimization techniques to improve prediction of human motion during human-robot collaboration. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '24. ACM, March 2024.

[71] Michael Walker, Hooman Hedayati, Jennifer Lee, and Daniel Szafir. Communicating robot motion intent with augmented reality. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '18, page 316–324, New York, NY, USA, 2018. Association for Computing Machinery.

[72] Michael E. Walker, Hooman Hedayati, and Daniel Szafir. Robot teleoperation with augmented reality virtual surrogates. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 202–210, 2019.

[73] Eric Rosen, Nishanth Kumar, Nakul Gopalan, Daniel Ullman, George Konidaris, and Stefanie Tellex. Building plannable representations with mixed reality. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11146–11153. IEEE, 2020.

[74] Yuanzhi Cao, Tianyi Wang, Xun Qian, Pawan S Rao, Manav Wadhawan, Ke Huo, and Karthik Ramani. Ghostar: A time-space editor for embodied authoring of human-robot collaborative task with augmented reality. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, pages 521–534, 2019.

[75] Zhanat Makhataeva and Huseyin Atakan Varol. Augmented reality for robotics: A review. *Robotics*, 9(2):21, 2020.

[76] Daniel Szafir. Mediating human-robot interactions with virtual, augmented, and mixed reality. In *International Conference on Human-Computer Interaction*, pages 124–149. Springer, 2019.

[77] Ryo Suzuki, Adnan Karim, Tian Xia, Hooman Hedayati, and Nicolai Marquardt. Augmented reality and robotics: A survey and taxonomy for ar-enhanced human-robot interaction and robotic interfaces. In *CHI Conference on Human Factors in Computing Systems*, pages 1–33, 2022.

[78] Gopika Ajaykumar, Maureen Steele, and Chien-Ming Huang. A survey on end-user robot programming. *ACM Comput. Surv.*, 54(8), oct 2021.

[79] Yuanzhi Cao, Tianyi Wang, Xun Qian, Pawan S. Rao, Manav Wadhawan, Ke Huo, and Karthik Ramani. Ghostar: A time-space editor for embodied authoring of human-robot collaborative task with augmented reality. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19, page 521–534, New York, NY, USA, 2019. Association for Computing Machinery.

[80] Philipp Mittendorfer, Eiichi Yoshida, and Gordon Cheng. Realizing whole-body tactile interactions with a self-organizing, multi-modal artificial skin on a humanoid robot. *Advanced Robotics*, 29:51 – 67, 2015.

[81] Luka Peternel, Nikolaos G. Tsagarakis, and Arash Ajoudani. Towards multi-modal intention interfaces for human-robot co-manipulation. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2663–2669, 2016.

[82] Matteo Fumagalli, Serena Ivaldi, Marco Randazzo, Lorenzo Natale, Giorgio Metta, Giulio Sandini, and Francesco Nori. Force feedback exploiting tactile and proximal force/torque sensing. *Autonomous Robots*, 33:381 – 398, 2012.

[83] Serena Ivaldi, Sébastien Lefort, Jan Peters, Mohamed Chetouani, Joëlle Provasi, and Elisabetta Zibetti. Towards engagement models that consider individual factors in hri: On the relation of extroversion and negative attitude towards robots to gaze and speech during a human–robot assembly task. *International Journal of Social Robotics*, 9:63–86, 2015.

[84] Soshi Iba, Christiaan JJ Paredis, and Pradeep K Khosla. Interactive multimodal robot programming. *The international journal of robotics research*, 24(1):83–104, 2005.

[85] Raúl Marín, Pedro J Sanz, Patricio Nebot, and Raul Wirz. A multimodal interface to control a robot arm via the web: a case study on remote programming. *IEEE Transactions on Industrial Electronics*, 52(6):1506–1520, 2005.

[86] Batu Akan, Afshin Ameri, Baran Cürüklü, and Lars Asplund. Intuitive industrial robot programming through incremental multimodal language and augmented reality. In *2011 IEEE International Conference on Robotics and Automation*, pages 3934–3939. IEEE, 2011.

[87] Baichuan Huang, Deniz Bayazit, Daniel Ullman, Nakul Gopalan, and Stefanie Tellex. Flight, camera, action! using natural language and mixed reality to control a drone. In *2019 International conference on robotics and automation (ICRA)*, pages 6949–6956. IEEE, 2019.

[88] Connor Brooks and Daniel Szafir. Visualization of intended assistance for acceptance of shared control. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11425–11430, 2020.

[89] Eric Rosen, David Whitney, Elizabeth Phillips, Gary Chien, James Tompkin, George Konidaris, and Stefanie Tellex. *Communicating Robot Arm Motion Intent Through Mixed Reality Head-Mounted Displays*, pages 301–316. 01 2020.

[90] Michael Walker, Hooman Hedayati, Jennifer Lee, and Daniel Szafir. Communicating robot motion intent with augmented reality. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 316–324, 2018.

[91] Michael Görner, Robert Haschke, Helge Ritter, and Jianwei Zhang. Moveit! task constructor for task-level motion planning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 190–196, 2019.

[92] MixedRealityToolkit. Mixedrealitytoolkit-unity, 2023.

[93] microsoft. Mixedreality-qrcode-sample, 2021.

[94] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pages 28492–28518. PMLR, 2023.

[95] OpenAI. Hello GPT-4o. 2024.

[96] Greg Brockman, Peter Welinder, Mira Murati, and OpenAI. Openai: Openai api. 2020.

[97] Laurie Cheers, Devin M., Amanda, Hamid Younesy, Shuo Diao, Peter Smith, Stephan Hasler, and Tiffany Yau. ROS TCP Endpoint. https://github.com/Unity-Technologies/ROS-TCP-Endpoint.

[98] Eloise Matheson, Riccardo Minto, Emanuele GG Zampieri, Maurizio Faccio, and Giulio Rosati. Human–robot collaboration in manufacturing applications: A review. *Robotics*, 8(4):100, 2019.

[99] John Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[100] J. Sauro. *A Practical Guide to the System Usability Scale: Background, Benchmarks & Best Practices*. Measuring Usability LLC, 2011.

[101] Justin Huang and Maya Cakmak. Supporting mental model accuracy in trigger-action programming. In *Proceedings of the 2015 acm international joint conference on pervasive and ubiquitous computing*, pages 215–225, 2015.