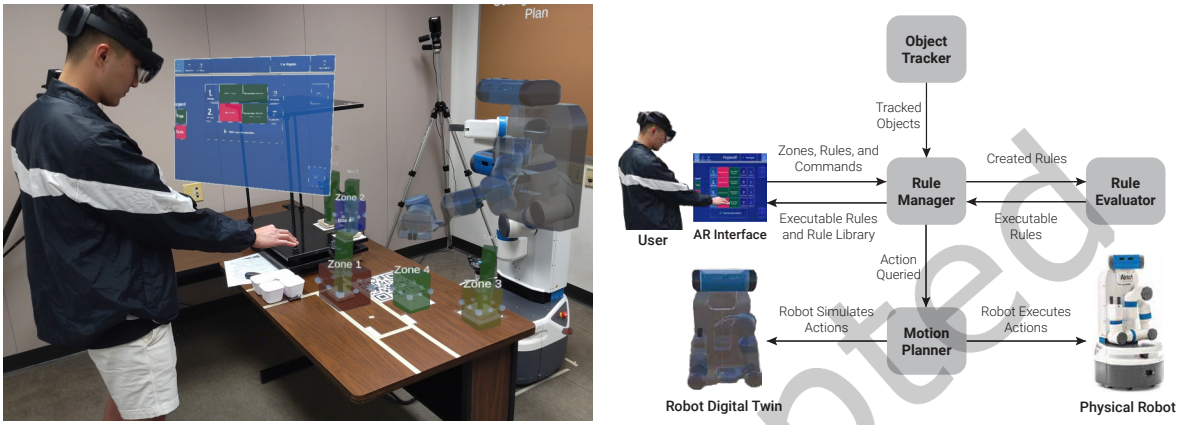
PRogramAR: Augmented Reality End-User Robot Programming

BRYCE IKEDA, University of North Carolina at Chapel Hill, USA
DANIEL SZAFIR, University of North Carolina at Chapel Hill, USA



(a) (b)

Fig. 1. We introduce PRogramAR, an augmented reality trigger-action programming (TAP) system that empowers non-expert users to program reactive robot behaviors. We describe the benefits of (a) combining augmented reality with TAP to ground program development in the context of its execution as implemented in (b) our system architecture.

The field of end-user robot programming seeks to develop methods that empower non-expert programmers to task and modify robot operations. In doing so, researchers may enhance robot flexibility and broaden the scope of robot deployments into the real world. We introduce *PRogramAR* (Programming Robots using Augmented Reality), a novel end-user robot programming system that combines the intuitive visual feedback of augmented reality (AR) with the simplistic and responsive paradigm of trigger-action programming (TAP) to facilitate human-robot collaboration. Through PRogramAR, users are able to rapidly author task rules and desired reactive robot behaviors, while specifying task constraints and observing program feedback contextualized directly in the real world. PRogramAR provides feedback by simulating the robot's intended behavior and providing instant evaluation of TAP rule executability to help end-users better understand and debug their programs during development. In a system validation, 17 end-users ranging from ages 18 to 83 used PRogramAR to program a robot to assist them in completing three collaborative tasks. Our results demonstrate how merging the benefits of AR and TAP using elements from prior robot programming research into a single novel system can successfully enhance the robot programming process for non-expert users.

CCS Concepts: • **Human-centered computing** → **Mixed / augmented reality**; *Usability testing*; • **Computer systems organization** → *Robotic autonomy*; *External interfaces for robotics.*

Authors' addresses: Bryce Ikeda, University of North Carolina at Chapel Hill, , Chapel Hill, North Carolina, USA, 27510, bikeda@cs.unc.edu; Daniel Szafir, University of North Carolina at Chapel Hill, , Chapel Hill, North Carolina, USA, 27510, dszafir@cs.unc.edu.

ACM 2573-9522/2024/1-ART
https://doi.org/10.1145/3640008

Additional Key Words and Phrases: End-User Robot Programming, Trigger-Action Programming (TAP), Augmented Reality (AR), Human-Robot Interaction (HRI), Human-Robot Collaboration (HRC)

1 INTRODUCTION

The proliferation of general computing technology necessitated the development of end-user tools, such as spreadsheets, for users who were not professional software developers. The increasing number of robot deployments (over 3 million robots operate in factories today [61]), creates a similar need for end-user robot programming. In pursuit of this goal, prior work in end-user robot programming has offered different programming paradigms (e.g., imperative [40], dataflow [33]) and representations (e.g., Hierarchical Finite State Machines [59], Behavior Trees[64]) to aid non-experts with programming robots. Recent work has begun extending these methods by using mixed reality technologies to improve an end-user's understanding of robot activities in 3D space [7, 15, 28, 31, 44, 49, 62, 68, 72]. However, these methods do not easily allow for programming complex reactive robot behaviors that are common in real-world robotics applications. Beyond the most common applications (e.g., manufacturing), we envision robots playing a valuable role in assisting individuals with everyday tasks. These tasks may include cleaning dishes or putting away groceries, where reactive robot behaviors are often necessary for coordinating interactions with humans. Such tasks require users define where objects are placed, triggers for actions, and multiple pick-and-place activities. For instance, when a user washes a dish and places it on a drying rack, a robot reacts by picking up the dish, drying it, and placing it in a user-defined location within a cabinet.

In pursuit of this vision, we combine the rich medium of AR with TAP. TAP, also known as event-driven programming, has gained popularity as a user-friendly approach for end-user programming, allowing users with no prior coding experience to develop successful reactive programs [77]. As a result, TAP has been adopted across a wide range of real-world domains, including project management, security systems, and smart hubs [69, 78]. In TAP, users define a set of circumstances known as *triggers* that initiate *actions* once the triggering conditions are met. In the context of dish washing, users can create a rule such as "**IF** a dish is on the drying rack, **THEN** dry the dish and place the dish in the cabinet." This rule prompts a robot to wait until it detects a dish on the drying rack, picking it up when detected, drying it, then placing it in a pre-defined cabinet location. The simplicity of TAP positions it to be an effective tool for non-expert users seeking to program robots for everyday tasks.

While prior work has explored the general notion of robot event-condition-action rules (e.g., [22, 83]), TAP has only recently been investigated for end-user robot programming [48, 52, 73]. In this context, Leonardi et al., 2019 [48] used TAP to enable non-expert users to craft reactive social robot behavior programs. Alternatively, Senft et al., 2021 [73] utilized TAP to enable non-expert users to program coordinated robot actions for human-robot collaboration (HRC) tasks. However, these existing TAP systems are constrained by a 2D screen development paradigm, which restricts users to defining programs and parameters in a manner disconnected from the actual operating environment of the robot. Such setups have been observed to diminish users' comprehension of the contextual aspects of their task [6, 37, 55]. Conversely, an ARHMD provides hands-free mobility, a wider field of view, and supports users in larger areas by allowing them to view the workspace from various perspectives. ARHMDs also enable more accurate depth estimation of virtual imagery, providing better blending of virtual and physical environments than 2D screens. In all previous studies focusing on TAP programming for robotics, users consistently expressed a desire for visual feedback and debugging support when building their trigger-action rules, as well as their own mental model of the system [48, 52, 73]. Our insight is to unlock the untapped synergies that exist between recent developments in AR and TAP, which we actualize in developing *PRogramAR* as a new end-user robotics programming system. To do this, we utilize an augmented-reality head-mounted display (ARHMD) to contextualize information directly in the user's scene and thus providing the following benefits: (1) Users can program the robot within the entire 3D workspace in which it operates, rather than being restricted to defining 2D zones on a tablet with a 2D field of view from the robot's camera as in Senft et al., 2021 [73], (2) Users can verify and monitor the correctness of their program by visualizing a simulated version of the robot's behavior

via the robot's 3D digital twin, (3) Integrating TAP within AR, rather than displaying it on a separate device (tablet or desktop), presents users with a cohesive and holistic system, reducing potential confusion and frustration from context switching across devices, and (4) Users can freely position the AR TAP interface without physically holding it, enabling them to effortlessly monitor both the physical workspace and TAP rules simultaneously (useful in debugging).

Contributions: We introduce PRogramAR, a system for supporting non-expert programmers with authoring reactive robot behaviors by adopting AR-based contextualization and simulation-based rule evaluation. By integrating known components—AR and TAP—we enhance the user's capability to coordinate actions effectively during collaborative tasks with a robot. To evaluate PRogramAR, we recruited 17 participants who used our system to author robot programs for three HRC tasks. In this study, HRC is used to describe a human-robot team working together towards a shared goal, based on terminology from prior work [7, 19, 73]. Overall, we contribute: (1) PRogramAR, a system for making reactive programming of robot manipulators easier for non-experts by combining AR and TAP, (2) A validation of the benefits of merging AR and TAP, with data collected from a diverse set of end-users with a wide age range, and (3) An open-source code release of PRogramAR generalizable to various robots and AR headsets to foster reproducibility and encourage future research and extensions by the community, which can be found at https://osf.io/gvxu5.

2 RELATED WORK

Our work on PRogramAR is inspired by past research in robot programming, trigger-action programming, and augmented reality.

2.1 Robot Programming

Prior research has investigated a variety of methods for robot programming. One prominent approach is that of skill demonstration (i.e., learning from demonstration / LFD), in which users define robot actions through kinesthetic teaching, teleoperation, or passive observation (see [10, 71] for relevant surveys). One advantage of LFD systems is that programming is directly embedded in the robot's operational context (i.e., how the robot moves in the real 3D environment); however, LFD can be difficult to generalize to new environments and is often used to teach a robot primitive motions, rather than to build coordination mechanisms that enable collaborative human-robot tasks through reactive robot programs. Methods for program specification present an alternative approach, where interfaces let users define and parameterize desired robot actions with varying degrees of abstraction. For example, research has explored visual robot programming tools where users allocate task execution through flow diagrams, behavioral trees, or block-based programming interfaces [8, 35, 38, 40, 70]. However, in order to use such systems, end-users are often required to know fundamental programming concepts (e.g., variables, conditionals, loops, etc.), which may limit system applicability. These systems also adhere to a traditional programming approach, requiring users to specify the whole robot program before execution. Moreover, the feedback provided by these systems is typically visualized on a 2D screen, thereby disconnecting it from the context of program execution where the physical robot moves through 3D space. Likewise teach pendants, currently the industry standard for end-user programming of repetitive robots tasks, can be complex and difficult to use for individuals that are not trained professionals [45, 66]. Therefore, we have designed a new program specification system that eliminates the need for end-users to possess prior programming knowledge, and enables non-traditional programming workflows that cater to users of all backgrounds. Furthermore, we leverage AR to provide immersive, visual feedback in the real operational space, directly connecting program development with program execution.

2.2 Trigger-Action Programming

Trigger-Action Programming, which forms the foundation of popular tools such as If-This-Than-That (IFTTT), Zapier, and SmartThings, has been successful in engaging users at all levels of programming proficiency [13, 21, 23, 32]. Research has investigated various ways to refine this programming process by improving support for user mental model formation when designing their trigger-action sets [39], understanding common bugs that occur in a TAP program [11, 63], and explaining TAP behavior in an understandable manner [84, 85]. More recently, researchers have begun to apply TAP to end-user robot programming. For instance, a study by Leonardi et al., 2019 [48] found that TAP can be an effective method for end-users to personalize social behaviors for humanoid robots. Further research was conducted by Manca et al., 2019 [52], who developed a visual analytics tool to understand the rules created by end-users. However, these systems focused solely on creating verbal responses to triggers without considering scenarios where a user may want to coordinate physical tasks with a robot. Senft et al., 2021 [73] built on this work in their Situated Live Programming (SLP) system, which provides a TAP interface for physical task coordination between humans and robots. In SLP, users can define regions in their workspace as zones containing objects and positions relevant to TAP rules. SLP also incorporates *live programming*, which provides users with the flexibility to program the initial robot actions, then gradually construct the complete program. With each step, users can define new trigger-action pairs based on the current state of the environment, facilitating incremental robot program development [73]. In contrast, traditional programming techniques require users to specify the whole program before execution. Although promising, SLP uses a top-down camera attached to the robot's end-effector to visualize the scene on a tablet. This configuration prohibits users from specifying zones outside the robot's 2D camera view. In addition, the tablet interface poses challenges in debugging TAP rules as users may find it difficult to simultaneously monitor both the virtual TAP scene and the physical actions performed by the robot in the real world. Notably, a common message shared by users in prior work was that these systems lacked feedback, such as the ability to observe rules in action before they were run on the actual robot or support for rule executability. To address these drawbacks, we leveraged mixed reality technology to provide intuitive visual feedback during programming in the form of a novel TAP system.

2.3 Augmented Reality Robot Programming

Augmented Reality (AR) has gained popularity in robotics due to its ability to provide contextual information *in situ* within a user's environment, potentially improving situational awareness, system usability, and overall user interactions (see [1, 50, 75, 76, 81] for recent surveys of mixed reality robotics). Prior research has explored various forms of AR, including 2D overlay displays, projection-based displays, and AR tablets. 2D overlay displays present a fixed view of the robot's workspace on a 2D computer screen, over which contextual information can be drawn [2, 73]. Projection-based displays directly project 2D visualizations into the user's workspace, often incorporating interaction mechanisms such as gesture tracking, smart touch tables, or programming wands [3, 30, 31, 53]. Tablets offer mobility by overlaying AR content over the tablet camera feed, enabling users to monitor the scene from any viewpoint [17, 26, 44, 47]. Utilizing the benefits of AR, which stem from overlaying virtual information onto the real-world, each of these viewing modalities have enhanced the robot programming process for users. However, 2D overlays have a limited field of view, inhibit the user's ability to communicate depth parameters, and visual feedback is viewed separate from the real-world. Projection-based displays are difficult to transfer to new environments and restrict mobility, while tablets occupy the user's hands and constrain the interface to within the small size of the screen.

Therefore, Augmented reality head-mounted displays (ARHMDs) have been used to alleviate these issues. ARHMDs free user's hands of hardware, promote unrestricted mobility and interaction throughout the whole workspace, and provide information directly contextualized in the user's real world. As a result, ARHMDs have been used to help industrial workers define robot trajectories and action primitives [18, 25, 68]. ARHMDs have

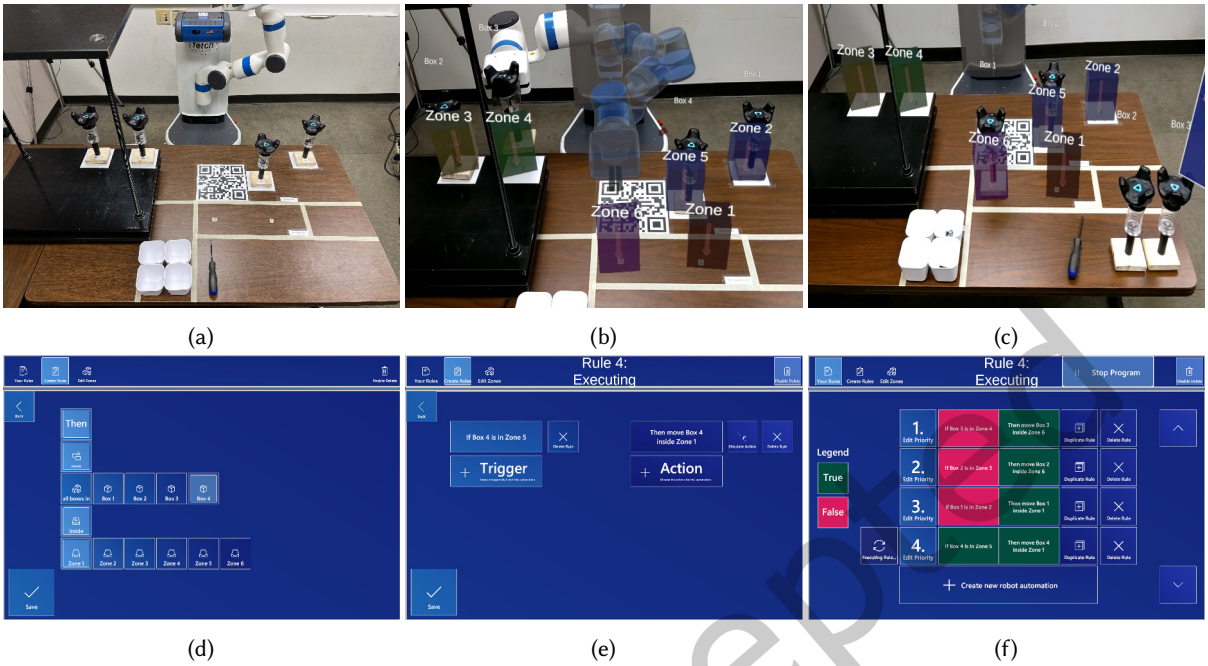


Fig. 2. An example workflow for Task 2. Participants (a) started with all objects in the robot's workspace, (b) then created zones for each box and each place position. The goal was to (c) move the boxes to the exchange area so the participant could assemble the parts inside the boxes. Participants used the AR Interface (d-f) to create various TAP rules to be run on the robot.

also been used to communicate low-level robot sensor information to experts [19, 58] and to facilitate debugging robot programs for expert roboticists [42]. However, such systems have been designed for specific professional workers, rather than for non-expert end-users. In other studies, robot motion intent was communicated to users via the *Robot Digital Twin*, but do not offer an easy way to author reactive robot actions for human-robot collaboration tasks [72, 80]. Our approach in developing PRogramAR is inspired by that of Kragic et al., 2018 [46], Bambusek et al., 2019 [7], and Gadre et al., 2019 [29], who use AR to assist users in performing collaborative tasks with robots, to which we add the lens of trigger-action programming for defining reactive robot behaviors.

3 SYSTEM DESIGN

PRogramAR is designed to make programming of robot manipulators easier by adopting AR-based contextualization and simulation-based rule evaluation in combination with the benefits of TAP. Our system, which draws on prototype designs and findings from prior research projects (e.g., [9, 15, 48, 72, 73, 80]), is composed of seven components: (1) *AR Interface*, (2) *Rule Manager*, (3) *Object Tracker*, (4) *Rule Evaluator*, (5) *Motion Planner*, (6) *Physical Robot*, and (6) *Robot Digital Twin* (Figure 1b). An example of a full workflow from our system validation (§4) is depicted in Figure 2. In the following sub-sections, we describe each component of our system design.

3.1 AR Interface

Users interact with PRogramAR through an *AR Interface* embedded directly within the human-robot working environment (Figure 1a). This interface helps users create rules that are defined by triggers and paired actions

that dictate when and how a robot should perform a task. To ground TAP rules in the real world, users create, move, resize, and delete 3D zones within the real environment to indicate regions relevant to triggers or actions (once created, each zone has a different preset color and a unique zone number displayed above it for ease of reference). By utilizing 3D zones, users gain complete expressibility as they can communicate depth information in 3D spaces such as shelves. This is in contrast to prior work that relied on 2D zones, which limited users' ability to specify depth information. With our interface, PRogramAR supports both traditional programming processes, where users define their full program before execution, and *live programming*. In *live programming*, users can program TAP rules while the robot is planning (regardless of planning time) or executing actions, although edits may require re-planning.

3.2 Trigger-Action Programming Rules

PRogramAR currently supports combining triggers and actions into two types of TAP rules that support a user's mental model of a program: *If-Then* rules, as previously supported by Senft et al., 2021 [73], and our own addition of *While-Do* rules [39]. Triggers are parameterized by *objects*, recognized items known to the system (tracked by the *Object Tracker*, with tracking described more in §3.5), *conditions* (e.g., "is", "in"), and *zones*. In our system, the currently supported triggers include when, (1) objects are *in* a *zone* (e.g., Box 1 *is in* Zone 2), or (2) objects are *not in* a *zone* (e.g., Box 2 is *is not in* Zone 3). Actions are parameterized by a *robot action* (e.g., "move"), *objects*, and *zones*. Due to the limited capabilities of our robot, the only supported action is moving an object from a *zone* or its current location, to another *zone* (e.g., *move* Box 2 *inside* Zone 3). When the defined trigger is true, an *If-Then* rule executes its actions once before moving to the next rule. A *While-Do* rule performs its associated actions continuously as long as the condition is true before moving to the next rule. For instance, in a scenario where multiple objects need to be transferred from one zone to another, an *If-Then* rule might move only one object before moving to the next rule. In contrast, a *While-Do* rule might move all the objects before proceeding to execute the next rule. To summarize, the current rules, triggers, and actions supported by PRogramAR are as follows:

- Rules: *If-Then* and *While-Do*
- Trigger: Objects present within a zone
- Trigger: Objects absent within a zone
- Action: Moving objects from one zone to inside another zone
- Action: Moving objects from any location to inside a zone

The simplest rule would consist of a single trigger, containing a single object-zone pair, and a single action, also with a single object-zone pair. For example, a rule could be "If [Box 1] is [in] [Zone 1], then [move] [Box 1] inside [Zone 2]." Users can also specify rules of arbitrary complexity by using additional conditions connected by *AND* or *OR* logical operators in the rule triggers. Each trigger can also have multiple actions connected by *AND* operators. Similar to SLP [73], PRogramAR allows users to define, edit, and delete TAP rules at any time (prior, during, or post robot execution). This creates a *live programming* environment to aid with debugging and progressively building reactive robot programs at runtime. In contrast to SLP [73], which prompts users to fix rule priority conflicts, PRogramAR executes TAP rules in a user-specified order that can be adjusted as needed (Fig. 2f). Moreover, PRogramAR leverages AR to make use of all three dimensions of the user's workspace. This enables users to specify programs for placing a box on a shelf or, in future real world scenarios, moving a plate from a dish rack into a cabinet. Such programs are challenging to express in SLP [73], which relies on a top-down view, as it lacks depth perception, making it difficult to communicate spatial relationships accurately.

3.3 Rule Manager and Evaluator

As users create rules, they are maintained in a library within a *Rule Manager*, which communicates with other system components to manage rule options, available zones, and tracked objects while pushing updates to the AR interface. One key feature that goes beyond prior TAP systems such as SLP [73] or Leonardi et al., 2019 [48], is the *Rule Evaluator*. This component continuously checks whether the conditions of a rule are satisfied by the current state of the world. The *Rule Manager* then pushes updates to the AR Interface to reflect the status of each rule. The purpose of this feedback is to assist users with debugging their created rules by explicitly indicating whether a rule should or should not be executed. If the output of the *Rule Evaluator* conflicts with a user's expectations, then they may need to either edit their rules, or validate the current state of the world or virtual zones. Triggers and actions that evaluate to true, and are therefore in the queue to be executed, are colored green. Rules that evaluate to false, and are therefore not going to execute, are colored red (Fig. 2f). Red/green hues were chosen from a color blind accessible pallet to provide a level of contrast easily differentiable by all users [60].

3.4 Motion Planner and Robot Simulation

PRogramAR leverages the MoveIt! Task Constructor (MTC), an open-source software for robot manipulator action planning that is compatible with over 150 robot platforms using MoveIt [36]. While our current implementation targets the Fetch robot, a mobile robot with a 7-degree of freedom manipulator, other developers interested in utilizing or expanding PRogramAR can easily adapt it to their own MTC-compatible robot by replacing the MTC bindings specific to Fetch (e.g., updating the platform configuration in the launch file) [82]. When users choose to simulate or execute their program, the *Rule Manager* sends an *action query* to the *Motion Planner*. The MTC framework facilitates the planning of robot manipulator actions by solving individual sub-tasks and connecting them into a complete action plan [36]. For instance, a pick-and-place task can be divided into stages such as robot approach, grasp pose, and lift direction. Each stage is solved using a motion planning framework such as OpenRAVE or MoveIt!, and subsequently connected sequentially to generate the full motion plan [20, 24].

If the *Motion Planner* successfully generates a feasible motion plan for the programmed action, users have the option to simulate individual rules using the *Robot Digital Twin*. The digital twin is overlaid on top of the physical robot and demonstrates simulated motion plans contextualized in the real world. Once users are satisfied the simulation, they may execute their program on the physical robot. During this time, a digital twin continues to mirror the robot's actions at twice the robot's speed, enabling users to preview the robot's behavior both before and during execution (Fig. 2b). At times, the *Motion Planner* may be unable to return a valid trajectory due to unreachable zones or objects, or obstacles that could cause collisions. In such cases, the system notifies users with an error message projected above the robot's head stating, "Error: Zone too far or pick/place position too close to other boxes." Users can then debug their program accordingly. This approach is motivated by prior work emphasizing the importance of error detection and prevention in enhancing programming success [35, 39].

3.5 AR Apparatus and Tracking

PRogramAR makes use of an ARHMD to present the main AR Interface and visualizations to users. While our current implementation uses the HoloLens 2, PRogramAR is built on top of the OpenXR application protocol interface (API), which allows any compatible mixed reality device, such as the MagicLeap or Meta Quest, to run our application. PRogramAR relies on having a single fiducial marker placed in the workspace for aligning the coordinate frames of the virtual environment with the real world. The *Object Tracker* currently relies on external markers (in our validation we used four Vive trackers with lighthouse base stations); in the future, this might be performed directly using visual processing of the camera feeds from the ARHMD and/or robot. We derived the translation and rotation matrices necessary for aligning and calibrating our various coordinate systems (ARHMD,

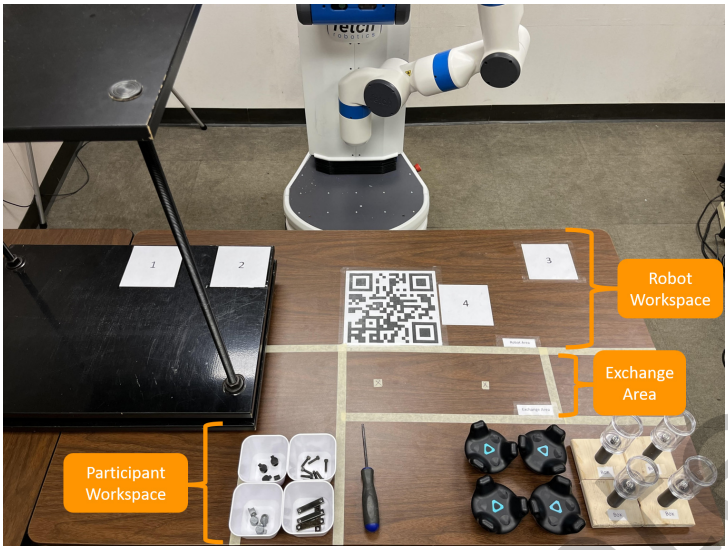


Fig. 3. The task workspace was divided into three areas, the middle of which both the participant and the robot could work in. The shelf on the left provided a 3D component to the workspace, which 2D interfaces cannot account for.

object tracking, and robot) as described by Peer et al., 2018 [65] such that user actions and program specifications could be accurately mapped into robot plans and AR visual feedback was appropriately displayed.

4 SYSTEM VALIDATION

To evaluate PRogramAR, we designed and conducted a validation study in which participants programmed three collaborative tasks with a Fetch robot.

4.1 Environment

Participants used PRogramAR in a controlled laboratory setting using a $1.2m \times 0.75m$ table as the shared workspace. On one side of the table there was a $.58m \times 0.4m \times .11m$ shelf, which the robot could be programmed to place objects on. Unlike previous TAP systems that were limited to a 2D top-down view of the workspace, we demonstrate the benefits of an ARHMD by requiring 3D placements of objects on the shelf on the left side of the robot's workspace (Figure 3). This poses a challenge for 2D interfaces when communicating depth information for TAP rules, especially when the top-down camera view may be occluded by the roof of the shelf. To align with real-world safety standards for human-robot shared workspaces [54, 79], the task space was divided into three areas: (1) *Robot workspace*, where only the robot was allowed to work during execution, (2) *Exchange area*, where both the participant and the robot could enter while working, and (3) *Participant workspace*, where only the participant could work (Figure 3).

4.2 Programming Tasks

For this study, participants programmed a Fetch robot to perform three tasks that grew in complexity and therefore increased potential for working in parallel towards a shared goal: (1) Kitting, (2) Assembly-A, and (3) Assembly-B. These tasks were inspired by prior work [73] and real world use cases [54] and each had a time cap for completion. While kitting and assembly do resemble manufacturing tasks, we believe the requirement

Table 1. A summary of quantitative results from our study.

Measure	Result
# users who completed Task 1	11 (68.75%)
# users who completed Task 2	10 (62.5%)
# users who completed Task 3	11 (68.75%)
Mean completion time for Task 1	$16m\ 4s$ (SD = $2m\ 39s$)
Mean completion time for Task 2	$12m\ 40s$ (SD = $2m\ 16s$)
Mean completion time for Task 3	$18m\ 15s$ (SD = $3m\ 35s$)
# users who worked in parallel with the robot during Task 1	0 (0%)
# users who worked in parallel with the robot during Task 2	9 (56.25%)
# users who worked in parallel with the robot during Task 3	14 (87.5%)
Mean user age	26.88 (SD = 9.14)
# users with no programming experience	8 (50%)
# users owning IoT device(s)	7 (43.75%)
Mean prior experience using TAP (1-7)	3 (SD = 2.09)
Mean prior experience using AR (1-7)	2.81 (SD = 2.16)
Mean prior experience using robots (1-7)	2.19 (SD = 1.88)
Mean SUS score	77.81 (SD = 11.79)

of defining where objects are placed, triggers for performing actions, and multiple pick and place activities are transferable to other service applications (putting away dishes/groceries or tidying rooms) where similar specifications are necessary.

Task 1. Kitting: (20 minute cap) Participants put two screws, one metal bar, one grey round fastener and one black round fastener, into four different boxes. Participants then programmed the robot to move the boxes from the *exchange area* to particular locations within the *robot workspace* (numbered white squares in Figure 3).

Task 2. Assembly-A: (15 minute cap) Participants programmed the robot to move boxes from the *robot workspace* into the *exchange area*. Once received, participants assembled four items from pieces in each container, placing them into white bins located nearby.

Task 3. Assembly-B: (25 minute cap) Participants programmed the robot to move boxes from the *robot workspace* into the *exchange area*. Once received, the participant assembled the pieces from each container, put the assembled object back in the container and programmed the robot to move the boxes from the *exchange area* to one of the four initial positions in the *robot workspace*.

To accomplish these tasks, the robot must be programmed to transfer objects to and from different zones and object locations within the workspace. Participants were tasked with assembling objects that were chosen to be intentionally difficult to handle, with the aim of increasing the probability of success for participants who collaborated in parallel, rather than sequentially, with the robot. The collaborative behavior of working in parallel towards a shared goal was identified when a participant actively engaged in their own physical tasks while the robot simultaneously executed its own user-programmed tasks.

4.3 Participants & Procedure

For this study, approved by our university IRB, we recruited a total of 20 participants from our local community through our university's online research recruitment platform. Since PRogramAR is intended for applications beyond manufacturing (e.g., services in the home), we recruited participants of all age ranges and experience levels. Three participants had technical difficulties and were unable to continue the study (e.g., failures with

objects trackers or Wi-Fi connectivity). One participant, aged 83, lacked the dexterity to manually assemble our task objects (screws and fasteners). Instead, this participant performed a modified version of our study, thus this participant's data is analyzed separately (see §5). As a result, our primary data set includes 16 participants (4 male, 10 female, 1 other and 1 prefer not to say), summarized in Table 1. The average age of the participants was 26.88 years (SD = 9.14) across a range of 18–58. Eight participants (50%) reported having no computer programming experience, three (18.75%) reported 1 year or less, and five (31.25%) reported 3 years or more. Seven (43.75%) of the participants indicated they own an IoT device, such as a smart hub, and participants' average familiarity with trigger-action programming was 3.00 (SD = 2.09) using a single item with a seven-point scale. Prior to the study, participants reported having little previous experience working with robots (M = 2.18, SD = 1.88) or using virtual or augmented reality technology (M = 2.81, SD = 2.16 on seven-point scales) on a seven-point range. Our sample of participants represents a broad distribution of age groups with different levels of experience, which reflects many of our target end-users.

Each participant's session consisted of six phases: (1) Introduction, (2) Kitting, (3) Assembly-A, (4) Assembly-B, and (5) Conclusion. (1) Participants were given time to read and sign a consent form. Then the researcher explained what they would be doing and showed the participant a 5 minute tutorial video explaining how to use PRogramAR to program the robot. Then, the HoloLens was calibrated for the participant. Finally, participants were asked if they had any questions before beginning the first task. (2) Participants began the first task and a timer was started once they acknowledged they could see the interface in the scene. For this task, they were given 20 minutes and were allowed to ask clarifying questions on how the interface worked, but not how to complete the task. Once the robot correctly placed the final object, the task was completed and the timer was stopped. (3) For the Assembly-A task, participants were given 15 minutes and were allowed to reuse the rules and the zones created from the first task. The timer was started once the participant verbally confirmed they could see the interface in the scene. Once the robot correctly placed the final object, the task was completed and the timer was stopped. (4) For the Assembly-B task, participants were give 25 minutes and were allowed to reuse the rules and the zones created from the first and second task. The timer was started once the participant verbally confirmed they could see the interface in the scene. Once the robot correctly placed the final object, the task was completed and the timer was stopped. (5) Following the final task, the researcher conducted a verbal interview with participants to understand their experience using PRogramAR. Then participants completed a questionnaire to gather demographics, and to assess the perceived usability of our system via the System Usability Scale (SUS) [14]. Finally, the researcher debriefed participants by explaining the goal of the study and compensated them with a $15 gift card.

4.4 Analysis Method

To gather participants' feedback regarding their experience with PRogramAR, we conducted and recorded a semi-structured verbal interview with a pre-defined list of questions focused on the participants' interaction with the robot, the effectiveness of the programming tool, and their overall impressions of PRogramAR. We chose semi-structured interviews because it offers a balance between structure and the flexibility to follow-up on unanticipated and interesting responses [27]. To transcribe the interview recordings, we used an intelligent verbatim approach, aligning spoken data with written conventions while preserving the intended meaning and structure of the original speech [56]. Our goal was to convey the key points and ideas in the conversation, rather than how it was said, and to improve readability. Following the transcription of our data, we applied thematic analysis, a method for identifying, organizing, and reporting patterns within a data set, enabling us to systematically summarize key features of verbal feedback gathered from participants. Thematic analysis is performed in six steps: (1) familiarization with the data, (2) generating initial codes, (3) searching for themes, (4) reviewing potential themes, (5) defining and naming themes, (6) producing the report [12]. Our analysis revealed

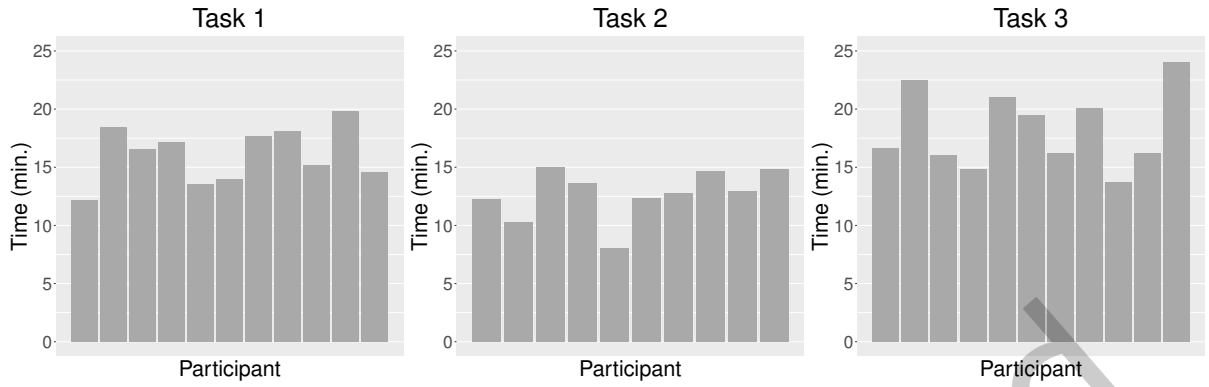


Fig. 4. The time it took for participants to complete each task. The programs created in Task 1 could be reused for Task 2, resulting in quicker completion times for Task 2. Task 3 took the longest time because it required more steps. Five participants in Task 1, six in Task 2, and five in Task 3 were unable to complete the tasks within the allotted time. Their times are not shown within the graphs.

three themes, discussed in §5: (1) User-friendly robot programming, (2) Supporting different levels of expression, and (3) Supporting users through *in-situ* contextualization.

5 RESULTS

Following the completion of the study, participants gave our system an average SUS rating of 77.81 (SD = 11.79) resulting in what is an above average rating. A number of participants struggled to complete specific tasks within the designated time constraints. We believe these instances are largely due to participants, many of whom had no programming, AR, or robotics experience, having a relatively short learning time (5 minutes) to become familiar with the many novel aspects of our system. Overall, participants averaged 16 minutes, 4 seconds (SD = 2 minutes, 39 seconds) to complete the first task, 12 minutes, 40 seconds (SD = 2 minutes, 16 seconds) to complete the second task and 18 minutes, 15 seconds (SD = 3 minutes, 35 seconds) to complete the third, and we believe with higher time caps all participants would have eventually completed all tasks. In general, Task 1 took longer than Task 2 because participants needed extra time to understand the interface and the AR interactions. Participants could also re-purpose TAP rules created in Task 1 for Task 2, which saved time. As expected, Task 3 took the longest amount of time because it involved moving objects between the *participant workspace* and the *robot workspace* twice, rather than once (see Figure 4 for more details).

Over the course of the tasks participants became comfortable creating multiple rules at once and working in parallel with the robot. The number of participants who performed their tasks in parallel with the robot were zero (0%) in Task 1, nine (56.25%) in Task 2, and fourteen (87.5%) in Task 3. While this increase was affected by the task design (i.e., Task 3 was designed to generate more opportunities for parallel task execution), participants stated they became more comfortable using PRogramAR and progressively let the robot perform tasks simultaneously with them:

P10: *"It was good once I got used to it. I think if you were in this environment, working this way . . . it will seem comfortable, to me once you do it a few times . . . By the third task, I was like, well, I'm going to be doing something while I'm having it do something."*

This emphasizes that with time to become comfortable, PRogramAR allowed users to manage their own tasks simultaneously with the robot's tasks. Below, we discuss other advantages and disadvantages of PRogramAR reported by participants grouped by the themes that emerged from our analysis.

5.1 User-friendly robot programming

Of the sixteen participants, thirteen (81.25%) commented on their positive experience using PRogramAR. These participants in particular appreciated its simplicity especially for non-experts, as reflected in the following comment:

P8: *"I don't have to be a genius to be able to do this. It's not super confusing . . . I was actually surprised about that."*

Moreover, seven (43.75%) participants reported that using TAP within PRogramAR was less intimidating than typical computer programming, while three (37.5%) participants without a computer science background perceived that TAP was comparable to their current work applications and therefore felt familiar.

P11: *"I wasn't really thinking about the coding element that much, which I think is good, probably in the sense of being user friendly. I don't think normal coding is super user friendly."*
P16: *"I've done this type of if-then work in other database management . . . that's why I feel like I got used to the rule language pretty quickly."*

This feedback highlights that TAP is perceived as user-friendly when applied in an AR environment and reinforces the notion that TAP can lower the barrier to entry for robot programming as found in prior work [67]. When participants were asked if they believed there was a group of users who would have a hard time learning this interface, four (25%) said that it would be their grandparents. On the contrary, one elderly participant (P7, age 83) who completed a modified version of our study, due to difficulties manually assembling task objects, reported a positive experience learning and using PRogramAR. During the first two tasks, this participant was provided additional guidance from the researcher, who helped them develop successful program for the first two tasks. For the third task, due to the participant having difficulty putting objects together, they were instructed to put them to the side without assembling them, and to continue with the task as usual. For this modified task, the participant was able to successfully build their program in the allotted time, without further guidance from the researcher. Including this participant's response, our SUS score rises to 79.12 (STD = 12.57). This experience suggests that users of all ages can adopt this technology if well-designed guidance is provided during the initial familiarization phase. This participant provided the following feedback on PRogramAR:

P7: *"It was a simple interface to learn and just took a little to get used to . . . I have a 97 year old friend who refused to use a computer . . . he could have picked up on this I'm sure."*

To further enhance the programming experience for users, color-coded TAP rules were added to the *Rule Manager* (Fig. 2f). This feature indicated to users whether a rule could be executed given the current state of the world. Our feedback revealed that four (25%) participants found this feature to be particularly helpful for verifying the planned execution of created rules, e.g.,:

P15: *"I liked that it showed over here which rule is being executed and the highlighted true false condition, like when it was off or when a box was in a zone it was highlighted true and the if condition column was highlighted green"*
P16: *"I realized that the conditions weren't true because the box wasn't in that zone. So that helped me move the zone back where the box was."*

This indicates that the color-coded TAP rule feedback served as an effective visual cue, enabling users to make informed adjustments to their program. In summary, our user responses reinforce previous findings that the TAP paradigm can be adopted by non-expert users, and that proper feedback regarding TAP rules can enhance the programming process [48, 73, 77]. Furthermore, it is encouraging that these ideas hold when providing TAP in AR to people of varying ages and backgrounds.

5.2 Supporting different levels of expression

As discussed by Senft et al., 2021 [73], our work similarly highlights that TAP supports different levels of expression. When it came to the final task, participants were able to apply their own unique strategies. Of the eleven participants that successfully completed the third task, seven (63.36%) set up multiple zones for each object and placement position for their rules. All but one participant utilized a *live programming* approach throughout this task, continuously building and editing blocks of rules (Figure 2). The remaining four (36.36%) preferred to use a smaller number of zones that they moved around the workspace as the task progressed, also utilizing a *live programming* approach. Across all three tasks, we observed a small number of participants adopt a traditional programming process, defining all the necessary rules for completing the task before execution. Specifically, one (6.25%) participant in Task 1, three (18.75%) participants in Task 2, and one (6.25%) participant in all three tasks used a traditional programming approach. Although participants had the freedom to choose their strategies, it did not guarantee their approach would be effective or successful. One challenge five (31.25%) participants encountered was with keeping track of multiple low-level rules. This was because participants often implemented too many zones and *If-Then* rules, for example:

P9: *"When I was adding the extra conditions and extra actions, I couldn't remember which ones I had already added . . . there's just a lot going on."*

Seven (53.84%) participants recognized that there might have been a more optimal approach than their initial implementation. Participants often attributed this difficulty to the time limit that created a sense of pressure to quickly incorporate multiple *If-Then* rules, which was the perceived easiest path.

P5: *"I think if I had more time, I would have experimented . . . For the third test, I started to look at the other command, While-Do, to see what that meant. If I had more time to think of a more finessed trigger command, just but because the If-Then was familiar, and I knew it could still execute the task at hand."*

Therefore, although participants were given the flexibility to generate TAP rules how they wanted, future research should explore ways to assist users by automatically generating high-quality rules that participants may modify after its creation. This becomes particularly crucial as the complexity of real-world scenarios increases with a larger number of available rules, objects, and zones.

5.3 Supporting users through *in-situ* contextualization

A major characteristic of AR is its ability to merge virtual and physical worlds. Previous research has examined this property in order to improve robot programming using ARHMDs [7, 16, 18]. One of the benefits of incorporating an ARHMD into robot programming is users are able to engage with the AR interface from anywhere in their workspace, unencumbered by a physical monitor. This benefit was recognized by two (12.5%) participants who stated:

P16: *"I like that no matter where I was sitting, I could kind of engage the interface."*
P17: *"I like that if you feel like using this at home, you wouldn't have a bunch of, you might have a monitor, but it would just be like this and be pretty simple."*

Despite this positive feedback, one participant mentioned an alternate viewpoint. They would have preferred using a computer, since it was what they were used to, indicating that some users may be resistant to adopting technologies that are perceived as disruptive to existing workflows. Another motivation for incorporating an ARHMD was to enable users to visualize the entire workspace in which they were operating, rather than being limited to a single camera's point of view. During our study, two (12.5%) participants mentioned they appreciated this feature, for example:

P2: *"I like that you can see what's actually happening. I used VR glasses . . . and that was one of the ones where it's a video game where you can't really see your surroundings. So it was cool how I have an idea about where you are, but also seeing this other things."*

Furthermore, multiple participants successfully completed each task by utilizing PRogramAR to define 3D location parameters on the shelf, which is difficult to do in prior work that utilized 2D screens. Another crucial feature that was provided by PRogramAR was the simulation component, which displayed motion plans using the *Robot Digital Twin*. This feature has been shown to improve safety and control in robotics applications when provided via an ARHMD [4, 37, 72]. Seven (43.75%) participants mentioned using the simulation to build confidence in their program, with comments including:

P5: *"Once I was able to have the If-Then statements do the simulation and make sure that it did the task as I intended, I felt pretty confident."*
P20: *"I think it was better when I simulated it because at least I knew there was a pass . . . it made me feel better about it, doing it the way that it was supposed to and also just making sure that whatever I put as the rule was correct."*

Another participant envisioned the benefit of using the simulation tool when programming more complex tasks for service robots deployed in the home.

P13: *"For even more complicated tasks, like household chores or something robots were capable of, that would be really helpful to see like exactly what's going to happen once you program some set path or set of actions"*

However, similar to prior observations (e.g., [16]), as participants became familiar with PRogramAR and the capabilities of the robot, eleven (68.75%) participants no longer found it necessary to rely on the simulation tool after the first task. One limitation of our simulation caused by the motion planner, was if a goal was located too far for the robot to reach or if occluding objects prevented the robot from grabbing an object, no motion plan would be generated. Therefore, instead of providing a simulated motion plan, error messages describing these edge cases would be displayed above the robot. Consequently, four (25%) participants struggled with the final task because they had difficulty deciphering the meaning of the error messages.

P6: *"I wanted to know what that error meant about the zone. There was no guidance on the panel . . . at one point I realized that I had accidentally moved one of my zones away from where I thought it was. I didn't realize that I had done that."*

Therefore, future research should continue to explore methods of integrating error feedback that naturally guides non-expert users towards identifying the source of program errors. Overall, participant feedback provided promising evidence regarding our motivation to integrate TAP into an ARHMD environment. For instance, participants derived confidence from using an ARHMD to verify their program using the simulation tool for the first task. However, more work is necessary to further understand the effectiveness of our system in the context of non-expert robot programming.

6 DISCUSSION

In this study, we aimed to explore how the benefits of AR may enhance the process of programming reactive robot behaviors using trigger-action programming. First, our work confirms prior findings demonstrating that TAP is a user-friendly approach for enabling non-experts to programming robots. We observed PRogramAR accommodate varying levels of expressivity, with some participants working in parallel with the robot, while others worked sequentially, and most of the participants instinctively applied a *live programming* approach. It is also encouraging to see that these findings hold when providing TAP in AR to people of varying ages (e.g., 18–83) and backgrounds, as multiple participants were able to complete the tasks within the given time frame using our system. Second, the inclusion of AR visualizations, particularly the 3D digital twin simulation, provided multiple users with confidence in their program execution. Participants also appreciated the ability to freely position the *AR Interface* anywhere in the scene and how AR allowed them to work within the entire human-robot workspace. We believe that by combining AR and TAP, our design of PRogramAR provides a starting point for future AR systems to build upon in order to provide richer expressions and visual debugging capabilities to users, thereby continuing to improve the end-user robot programming experience for all.

6.1 Limitations and Future Work

Although our validation of PRogramAR shows promise, limitations and future challenges remain. For example, the tasks programmed by users in our study were abstract (pick-and-place, generic assembly, etc.) and limited by the object tracking and manipulation capabilities of our system. Future work should examine more realistic and complicated tasks that initially motivated our work, such as tidying rooms or putting away dishes. To enable the specification of TAP rules for more real-world tasks, our system needs to incorporate: (1) a more advanced object detection and tracking algorithm such as Yolo [43], (2) precise low-level manipulation capabilities that include generalizable object grasping techniques (e.g., Contact-GraspNet [74]) and actions such as twisting, opening, or deictic gestures (e.g., learning from demonstration [71]), and (3) increased TAP rule expressibility by including more object states such as dirty or clean dishes, social behaviors [48], and trigger-action rules (e.g., As-long-as-Do, If-When-Then [39]). However, to utilize these capabilities, PRogramAR will need to assist users with defining more complex robot programs. One way to do this is through virtual kinesthetic teaching, where users directly manipulate a robot's digital twin to fine-tune action plans. Researchers could also utilize the AR headset's egocentric camera to record users physically demonstrating tasks. This demonstration could then be translated into parameters for robot action programs. Developing such a system also unlocks opportunities to leverage AR's advantages in larger spaces, where users will need to create and keep track of more rules and virtual objects. For example, the *Robot Digital Twin* could demonstrate complex tasks like cooking and depict changes in object states (e.g., before and after food is cut or cooked) in a larger kitchen environment. In lengthy tasks, a simulation tool could condense robot actions into a shorter timeframe, facilitating quick debugging by users. Studies within larger environments, in which the workspace cannot be covered by a single viewpoint, may lead to greater insight into the benefits of AR over a 2D interface. However, these capabilities currently pose as challenges for future AR robot programming research, as they are not yet available in existing AR systems. Another limitation of our study was the long planning times of our mobile manipulator. This was a limitation of the solver, that would cause the robot to collide with objects if planning was done too quickly. To address this issue, future systems could continuously compute and store motion plans to be used when an action is triggered. In addition, reducing the planning time for future systems will help introduce more aspects of *live programming*, that could allow users to initiate re-planning of robot trajectories quicker. Also problematic was the task times that constrained the users. This constructed an artificial limitation that prevented some participants from fully exploring the interface and creating desired rule sets. Therefore, future work might increase the task time or incorporate state of the art language models such as GPT-4, that could quickly provide applicable rules for

participants to reduce their mental load [34, 41]. For example, participants could describe high-level goals using natural language, which could then be input into a large-language model. Then, the model could generate the robot program rules and provide users with a subset of task relevant trigger and action parameters for program customization. This approach would provide users with code templates, eliminating the need to build rules from scratch. Moreover, this could reduce the cognitive load for participants as they work with larger rule sets, a consequence of more complex tasks, environments, and advanced robotics systems. Finally, future work should investigate how to continue to incorporate error feedback that seamlessly guides non-expert users to the source of program bugs. For example, PRogramAR could be improved by providing a simulated motion plan that highlights problematic collisions that may occur, rather than describing errors in plain language [5, 51, 57].

7 CONCLUSION

In this work we present PRogramAR, a novel augmented reality trigger-action robot programming system that empowers non-expert users to create reactive robot behaviors for collaborative tasks. In the development of PRogramAR, we integrated concepts from various domains of robot programming and augmented reality interface research into a unified and comprehensive system. Specifically, PRogramAR introduces a unique combination of trigger-action programming, offering a high-level abstraction of robot programming concepts, and augmented reality feedback that is contextualized directly within the user's environment to facilitate the construction of accurate mental models of the programmed behavior. In our system validation, individuals of different ages and levels of experience successfully developed and deployed programs that enabled them to work in collaboration with the robot towards a shared goal. Moreover, the feedback received from participants support the advantages of merging augmented reality and trigger-action programming in the context of robot programming. We look forward to further explorations of this work in pursuit of a universally user-friendly robot programming system.

REFERENCES

[1] Gopika Ajaykumar, Maureen Steele, and Chien-Ming Huang. 2021. A survey on end-user robot programming. *ACM Computing Surveys (CSUR)* 54, 8 (2021), 1–36.

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

[3] Rasmus S Andersen, Simon Bøgh, Thomas B Moeslund, and Ole Madsen. 2016. Task space HRI for cooperative mobile robots in fit-out operations inside ship superstructures. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 880–887.

[4] Pasquale Arpaia, Carmela Bravaccio, Giuseppina Corrado, Luigi Duraccio, Nicola Moccaldi, and Silvia Rossi. 2020. Robotic autism rehabilitation by wearable brain-computer interface and augmented reality. In *2020 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*. IEEE, 1–6.

[5] Giancarlo Avalle, Francesco De Pace, Claudio Fornaro, Federico Manuri, and Andrea Sanna. 2019. An Augmented Reality System to Support Fault Visualization in Industrial Robotic Tasks. *IEEE Access* PP (09 2019), 1–1. https://doi.org/10.1109/ACCESS.2019.2940887

[6] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. 2001. Recent advances in augmented reality. *IEEE Computer Graphics and Applications* 21, 6 (2001), 34–47. https://doi.org/10.1109/38.963459

[7] Daniel Bambuŝek, Zdeněk Materna, Michal Kapinus, Vítězslav Beran, and Pavel Smrž. 2019. Combining Interactive Spatial Augmented Reality with Head-Mounted Display for End-User Collaborative Robot Programming. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. 1–8. https://doi.org/10.1109/RO-MAN46459.2019.8956315

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

[9] Sara Beschi, Daniela Fogli, and Fabio Tampalini. 2019. CAPIRCI: a multi-modal system for collaborative robot programming. In *International Symposium on End User Development*. Springer, 51–66.

[10] Geoffrey Biggs and Bruce MacDonald. 2003. A survey of robot programming systems. In *Proceedings of the Australasian conference on robotics and automation*, Vol. 1. 1–3.

[11] Will Brackenbury, Abhimanyu Deora, Jillian Ritchey, Jason Vallee, Weijia He, Guan Wang, Michael L Littman, and Blase Ur. 2019. How users interpret bugs in trigger-action programming. In *Proceedings of the 2019 CHI conference on human factors in computing systems*. 1–12.

[12] Virginia Braun and Victoria Clarke. 2012. Thematic analysis. (2012).
[13] Julia Brich, Marcel Walch, Michael Rietzler, Michael Weber, and Florian Schaub. 2017. Exploring end user programming needs in home automation. *ACM Transactions on Computer-Human Interaction (TOCHI)* 24, 2 (2017), 1–35.
[14] John Brooke et al. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.
[15] Connor Brooks and Daniel Szafir. 2020. Visualization of Intended Assistance for Acceptance of Shared Control. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 11425–11430. https://doi.org/10.1109/IROS45743.2020.9340964
[16] Yuanzhi Cao, Tianyi Wang, Xun Qian, Pawan S Rao, Manav Wadhawan, Ke Huo, and Karthik Ramani. 2019. 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*. 521–534.
[17] Sonia Mary Chacko and Vikram Kapila. 2019. An augmented reality interface for human-robot interaction in unconstrained environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 3222–3228.
[18] Wesley P Chan, Maram Sakr, Camilo Perez Quintero, Elizabeth Croft, and HF Machiel Van der Loos. 2020. Towards a multimodal system combining augmented reality and electromyography for robot trajectory programming and execution. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 419–424.
[19] Andre Cleaver, Faizan Muhammad, Amel Hassan, Elaine Short, and Jivko Sinapov. 2020. SENSAR: A visual tool for intelligent robots for collaborative human-robot interaction. *arXiv preprint arXiv:2011.04515* (2020).
[20] David Coleman, Ioan Sucan, Sachin Chitta, and Nikolaus Correll. 2014. Reducing the barrier to entry of complex robotic software: a moveit! case study. *arXiv preprint arXiv:1404.3785* (2014).
[21] Yngve Dahl and Reidar-Martin Svendsen. 2011. End-user composition interfaces for smart environments: A preliminary study of usability factors. In *International Conference of Design, User Experience, and Usability*. Springer, 118–127.
[22] Michael De Rosa, Seth Goldstein, Peter Lee, Padmanabhan Pillai, and Jason Campbell. 2008. Programming modular robots with locally distributed predicates. In *IEEE International Conference on Robotics and Automation*. 3156–3162.
[23] Anind K Dey, Timothy Sohn, Sara Streng, and Justin Kodama. 2006. iCAP: Interactive prototyping of context-aware applications. In *International conference on pervasive computing*. Springer, 254–271.
[24] Rosen Diankov. 2010. Automated construction of robotic manipulation programs. (2010).
[25] Huy Dinh, Quilong Yuan, Iastrebov Vietcheslav, and Gerald Seet. 2017. Augmented reality interface for taping robot. In *2017 18th International Conference on Advanced Robotics (ICAR)*. 275–280. https://doi.org/10.1109/ICAR.2017.8023530
[26] Jared A Frank, Matthew Moorhead, and Vikram Kapila. 2017. Mobile mixed-reality interfaces that enhance human–robot interaction in shared spaces. *Frontiers in Robotics and AI* 4 (2017), 20.
[27] Fiona Fylan. 2005. Semi-structured interviewing. *A handbook of research methods for clinical and health psychology* 5, 2 (2005), 65–78.
[28] Samir Yitzhak Gadre, Eric Rosen, Gary Chien, Elizabeth Phillips, Stefanie Tellex, and George Konidaris. 2019. End-user robot programming using mixed reality. In *IEEE International conference on robotics and automation (ICRA)*. 2707–2713.
[29] Samir Yitzhak Gadre, Eric Rosen, Gary Chien, Elizabeth Phillips, Stefanie Tellex, and George Konidaris. 2019. End-User Robot Programming Using Mixed Reality. In *2019 International Conference on Robotics and Automation (ICRA)*. 2707–2713. https://doi.org/10.1109/ICRA.2019.8793988
[30] Ramsundar Kalpagam Ganesan, Yash K Rathore, Heather M Ross, and Heni Ben Amor. 2018. Better teaming through visual cues: how projecting imagery in a workspace can improve human-robot collaboration. *IEEE Robotics & Automation Magazine* 25, 2 (2018), 59–71.
[31] Yuxiang Gao and Chien-Ming Huang. 2019. PATI: a projection-based augmented table-top interface for robot programming. In *Proceedings of the ACM International Conference on Intelligent User Interfaces (IUI)*. 345–355.
[32] Manuel García-Herranz del Olmo, Pablo A Haya, and Xavier Alamán. 2010. Towards a ubiquitous end-user programming system for smart spaces. *Journal of Universal Computer Science* (2010).
[33] Dylan F Glas, Takayuki Kanda, and Hiroshi Ishiguro. 2016. Human-robot interaction design using interaction composer eight years of lessons learned. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 303–310.
[34] Maitrey Gramopadhye and Daniel Szafir. 2023. Generating Executable Action Plans with Environmentally-Aware Language Models. arXiv:2210.04964 [cs.RO]
[35] Kelleher R Guerin, Colin Lea, Chris Paxton, and Gregory D Hager. 2015. A framework for end-user instruction of a robot assistant for manufacturing. In *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 6167–6174.
[36] Michael Görner, Robert Haschke, Helge Ritter, and Jianwei Zhang. 2019. MoveIt! Task Constructor for Task-Level Motion Planning. In *2019 International Conference on Robotics and Automation (ICRA)*. 190–196. https://doi.org/10.1109/ICRA.2019.8793898
[37] Hooman Hedayati, Michael Walker, and Daniel Szafir. 2018. Improving Collocated Robot Teleoperation with Augmented Reality. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction* (Chicago, IL, USA) *(HRI '18)*. Association for Computing Machinery, New York, NY, USA, 78–86. https://doi.org/10.1145/3171221.3171251
[38] Gaoping Huang, Pawan S Rao, Meng-Han Wu, Xun Qian, Shimon Y Nof, Karthik Ramani, and Alexander J Quinn. 2020. Vipo: Spatial-visual programming with functions for robot-IoT workflows. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[39] Justin Huang and Maya Cakmak. 2015. Supporting mental model accuracy in trigger-action programming. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 215–225.
[40] Justin Huang, Tessa Lau, and Maya Cakmak. 2016. Design and evaluation of a rapid programming system for service robots. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 295–302.
[41] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*. PMLR, 9118–9147.
[42] Bryce Ikeda and Daniel Szafir. 2022. Advancing the Design of Visual Debugging Tools for Roboticists. In *Proceedings of the 2022 ACM/IEEE International Conference on Human-Robot Interaction* (Sapporo, Hokkaido, Japan) *(HRI '22)*. IEEE Press, 195–204.
[43] Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, and Bo Ma. 2022. A Review of Yolo algorithm developments. *Procedia Computer Science* 199 (2022), 1066–1073.
[44] Michal Kapinus, Vítězslav Beran, Zdeněk Materna, and Daniel Bambušek. 2019. Spatially situated end-user robot programming in augmented reality. In *IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. 1–8.
[45] Michal Kapinus, Zdeněk Materna, Daniel Bambušek, and Vitězslav Beran. 2020. End-User Robot Programming Case Study: Augmented Reality vs. Teach Pendant. In *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*. 281–283.
[46] Danica Kragic, Joakim Gustafson, Hakan Karaoguz, Patric Jensfelt, and Robert Krug. 2018. Interactive, Collaborative Robots: Challenges and Opportunities. In *IJCAI*. 18–25.
[47] Jens Lambrecht and Jörg Krüger. 2012. Spatial programming for industrial robots based on gestures and Augmented Reality. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 466–472. https://doi.org/10.1109/IROS.2012.6385900
[48] Nicola Leonardi, Marco Manca, Fabio Paternò, and Carmen Santoro. 2019. Trigger-Action Programming for Personalising Humanoid Robot Behaviour. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300675
[49] Matthew B Luebbers, Connor Brooks, Carl L Mueller, Daniel Szafir, and Bradley Hayes. 2021. Arc-lfd: Using augmented reality for interactive long-term robot skill maintenance via constrained learning from demonstration. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 3794–3800.
[50] Zhanat Makhataeva and Huseyin Atakan Varol. 2020. Augmented reality for robotics: A review. *Robotics* 9, 2 (2020), 21.
[51] Ivo Malỳ, David Sedláček, and Paulo Leitao. 2016. Augmented reality experiments with industrial robot in industry 4.0 environment. In *2016 IEEE 14th international conference on industrial informatics (INDIN)*. IEEE, 176–181.
[52] Marco Manca, Fabio Paternò, and Carmen Santoro. 2019. Analyzing trigger-action programming for personalization of robot behaviour in iot environments. In *International Symposium on End User Development*. 100–114.
[53] Zdeněk Materna, Michal Kapinus, Vítězslav Beran, Pavel Smrž, and Pavel Zemčík. 2018. Interactive spatial augmented reality in collaborative robot programming: User experience evaluation. In *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 80–87.
[54] Eloise Matheson, Riccardo Minto, Emanuele GG Zampieri, Maurizio Faccio, and Giulio Rosati. 2019. Human–robot collaboration in manufacturing applications: A review. *Robotics* 8, 4 (2019), 100.
[55] John P McIntire, Paul R Havig, and Eric E Geiselman. 2012. 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*, Vol. 8383. International Society for Optics and Photonics, 83830X.
[56] Caitlin McMullin. 2021. Transcription and qualitative methods: Implications for third sector research. *VOLUNTAS: International journal of voluntary and nonprofit organizations* (2021), 1–14.
[57] Dimitris Mourtzis, Vasilios Zogopoulos, and E Vlachou. 2017. Augmented reality application to support remote maintenance as a service in the robotics industry. *Procedia Cirp* 63 (2017), 46–51.
[58] F. Muhammad, A. Hassan, A. Cleaver, and J. Sinapov. 2019. Creating a Shared Reality with Robots. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 614–615. https://doi.org/10.1109/HRI.2019.8673191
[59] Hai Nguyen, Matei Ciocarlie, Kaijen Hsiao, and Charles C Kemp. 2013. Ros commander (rosco): Behavior creation for home robots. In *2013 IEEE International Conference on Robotics and Automation*. IEEE, 467–474.
[60] David Nichols. 2022. *Coloring for Colorblindness*. https://davidmathlogic.com/colorblind/
[61] International Federation of Robotics. 2021. *IFR presents world robotics 2021 reports*. https://ifr.org/ifr-press-releases/news/robot-sales-rise-again
[62] Soh-Khim Ong, AWW Yew, Naresh Kumar Thanigaivel, and Andrew YC Nee. 2020. Augmented reality-assisted robot programming system for industrial applications. *Robotics and Computer-Integrated Manufacturing* 61 (2020), 101820.
[63] Mitali Palekar, Earlence Fernandes, and Franziska Roesner. 2019. Analysis of the susceptibility of smart home programming interfaces to end user error. In *2019 IEEE security and privacy workshops (SPW)*. IEEE, 138–143.
[64] Chris Paxton, Andrew Hundt, Felix Jonathan, Kelleher Guerin, and Gregory D Hager. 2017. CoSTAR: Instructing collaborative robots with behavior trees and vision. In *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 564–571.

[65] Alex Peer, Peter Ullich, and Kevin Ponto. 2018. Vive Tracking Alignment and Correction Made Easy. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 653–654. https://doi.org/10.1109/VR.2018.8446435

[66] Alexander Perzylo, Nikhil Somani, Stefan Profanter, Ingmar Kessler, Markus Rickert, and Alois Knoll. 2016. Intuitive instruction of industrial robots: Semantic process descriptions for small lot production. In *2016 ieee/rsj international conference on intelligent robots and systems (iros)*. IEEE, 2293–2300.

[67] Andrew Petersen, Michelle Craig, Jennifer Campbell, and Anya Tafliovich. 2016. Revisiting why students drop CS1. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*. 71–80.

[68] Camilo Perez Quintero, Sarah Li, Matthew KXJ Pan, Wesley P Chan, HF Machiel Van der Loos, and Elizabeth Croft. 2018. Robot programming through augmented trajectories in augmented reality. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1838–1844.

[69] Amir Rahmati, Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. 2017. IFTTT vs. Zapier: A comparative study of trigger-action programming frameworks. *arXiv preprint arXiv:1709.02788* (2017).

[70] Muhammet Ramoğlu, Çağlar Genç, and Kerem Rızvanoğlu. 2017. Programming a robotic toy with a block coding application: A usability study with non-programmer adults. In *International Conference of Design, User Experience, and Usability*. Springer, 652–666.

[71] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. 2020. Recent advances in robot learning from demonstration. *Annual review of control, robotics, and autonomous systems* 3 (2020), 297–330.

[72] Eric Rosen, David Whitney, Elizabeth Phillips, Gary Chien, James Tompkin, George Konidaris, and Stefanie Tellex. 2020. *Communicating Robot Arm Motion Intent Through Mixed Reality Head-Mounted Displays*. 301–316. https://doi.org/10.1007/978-3-030-28619-4_26

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

[74] Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. 2021. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 13438–13444.

[75] Ryo Suzuki, Adnan Karim, Tian Xia, Hooman Hedayati, and Nicolai Marquardt. 2022. 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*. 1–33.

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

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

[78] Blase Ur, Melwyn Pak Yong Ho, Stephen Brawner, Jiyun Lee, Sarah Mennicken, Noah Picard, Diane Schulze, and Michael L Littman. 2016. Trigger-action programming in the wild: An analysis of 200,000 ifttt recipes. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 3227–3231.

[79] ALES Vysocky and PETR Novak. 2016. Human-Robot collaboration in industry. *MM Science Journal* 9, 2 (2016), 903–906.

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

[81] Michael Walker, Thao Phung, Tathagata Chakraborti, Tom Williams, and Daniel Szafir. 2022. Virtual, augmented, and mixed reality for human-robot interaction: A survey and virtual design element taxonomy. *arXiv preprint arXiv:2202.11249* (2022).

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

[83] Hui Zhang and Michael J Boyles. 2013. Visual exploration and analysis of human-robot interaction rules. In *Visualization and Data Analysis 2013*, Vol. 8654. 154–167.

[84] Lefan Zhang, Weijia He, Jesse Martinez, Noah Brackenbury, Shan Lu, and Blase Ur. 2019. AutoTap: synthesizing and repairing trigger-action programs using LTL properties. In *2019 IEEE/ACM 41st international conference on software engineering (ICSE)*. IEEE, 281–291.

[85] Lefan Zhang, Weijia He, Olivia Morkved, Valerie Zhao, Michael L Littman, Shan Lu, and Blase Ur. 2020. Trace2tap: Synthesizing trigger-action programs from traces of behavior. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 3 (2020), 1–26.