Designing User-Centric Programming Aids for Kinesthetic Teaching of Collaborative Robots

Gopika Ajaykumar*, Maia Stiber, Chien-Ming Huang

Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218, USA

Abstract

Just as end-user programming has helped make computer programming accessible for a variety of users and settings, end-user robot programming has helped empower end-users without specialized knowledge or technical skills to customize robotic assistance that meets diverse environmental constraints and task requirements. While end-user robot programming methods such as kinesthetic teaching have introduced direct approaches to task demonstration that allow users to avoid working with traditional programming constructs, our formative study revealed that everyday people still have difficulties in specifying effective robot programs using these methods due to challenges in understanding robot kinematics and programming without situated context and assistive system feedback. These findings informed our development of *Demoshop*, an interactive robot programming tool that includes user-centric programming aids to help end-users author and edit task demonstrations. To evaluate the effectiveness of Demoshop, we conducted a user study comparing task performance and user experience associated with using Demoshop relative to a widely used commercial baseline interface. Results of our study indicate that users have greater task efficiency while authoring robot programs and maintain stronger mental models of the system when using Demoshop compared to the baseline interface. Our system implementation and study have implications for the further development of assistance in end-user robot programming.

Keywords: Human-Robot Interaction, End-User Development, End-User Robot Programming

1. Introduction

End-user programmers—programmers whose primary jobs do not involve coding professionally [1]—are an increasing population that leverages the growing power of computation by writing computer programs to meet their objectives

*Corresponding author

Email addresses: gopika@cs.jhu.edu (Gopika Ajaykumar), mstiber@jhu.edu (Maia Stiber), cmhuang@cs.jhu.edu (Chien-Ming Huang)

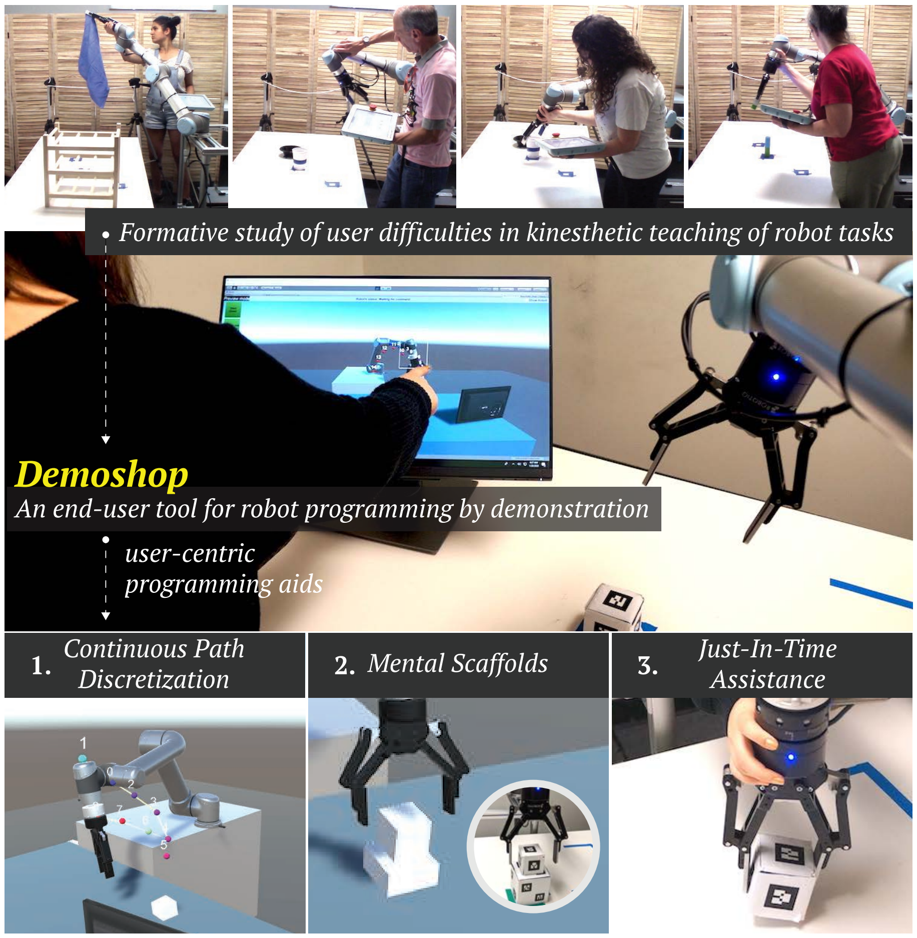
Formative study of user difficulties in kinesthetic teaching of robot tasks
Demoshop
An end-user tool for robot programming by demonstration
user-centric programming aids
1. Continuous Path Discretization
2. Mental Scaffolds
3. Just-In-Time Assistance

Figure 1: Based on user difficulties and needs that we observed in a formative study where users used kinesthetic teaching to program a robot, we developed Demoshop, an end-user robot programming tool that augments user capabilities in specifying motion demonstrations. Demoshop provides user-centric programming aids in the form of continuous path discretization, mental scaffolds, and just-in-time assistance.

regardless of their programming proficiency. Tools for end-user programming have become widely available and are commonly integrated into a diversity of platforms and application domains, such that most programs are now written by end-users and not professional software programmers [2]. To date, end-user computer programming has helped foster the development of innovative applications, the economic growth of individuals and small businesses, and the productivity of workplaces by making automation tools accessible for a wide range of employees [3, 4].

Similarly, *end-user robot programming* promises to democratize robotic assistance and foster economic growth by paving the way for collaborative robots to enter a variety of industries, including smaller companies for which robotic automation has traditionally been difficult to adopt because of high expenses or insufficient programming background among employees. In recent years, manufacturing firms in particular have embraced end-user robot programming. Collaborative robots from companies such as *Franka Emika* and *Universal Robots* can be programmed for a variety of manipulation tasks without coding and are becoming technical drivers for flexible, smart manufacturing. As robots become increasingly common in everyday environments, domains outside of manufacturing, such as education and outreach (e.g., [5]), medicine (e.g., [6]), and therapy (e.g., [7]), have started following suit in adopting end-user robot programming.

To increase accessibility, prior work on end-user robot programming has explored various programming interfaces and methods to reduce the high complexity involved in robot programming. In particular, visual programming is a common end-user robot programming method for simplifying logic specification when programming robot applications. For instance, *Lego Mindstorms*, Scratch [8], and CoBlox [9] utilize draggable block instructions to allow users to easily author the full structure and flow of a robot program. Due to the simplicity of its abstract nature, block-based visual programming has helped make robot programming accessible for a wide variety of people. However, this form of programming is constrained to work with tasks involving simple perceptual and action capabilities. As a result, block-based programming environments often lack scalability to more complex manipulation tasks.

To address this limitation, many robot programming interfaces, especially those used in manufacturing (e.g., *Universal Robots'* PolyScope and *Elephant Robotics*' ElephantOS), combine visual programming together with *kinesthetic teaching* (e.g., [10]), enabling users to specify more complex robot programs by physically guiding the robot to demonstrate a specific task. This hybrid approach allows for more complex robot manipulation, opening up the use of robot programming for more diverse functional tasks in real-world applications. Nonetheless, programming interfaces that incorporate kinesthetic teaching can still be challenging to use for end-users unfamiliar with robot kinematics, leading to unproductive demonstrations that require further editing and refinement. *How can we design user-centric demonstration aids to help end-users produce effective and efficient demonstrations?*

To answer this question, we first conducted a formative study to explore the barriers faced by users when programming a robot using a commercial end-user robot programming interface widely used today that combines visual programming and kinesthetic teaching. The study revealed that users experienced several difficulties in authoring and editing robot demonstrations and in receiving appropriate system feedback upon encountering challenges. We translated the user needs and challenges we observed into a set of design requirements that guided our development of *Demoshop*, an interactive robot programming tool that assists end-users in authoring and editing task demonstrations (Fig. 1).

Demoshop enables fine-grained, direct manipulation of a demonstrated trajectory via *continuous path discretization* and common tools for authoring and editing such as selection, insertion, and inspection. Demoshop provides *men-*

tal scaffolds throughout the programming process to help users understand the state of their program, the programming system, and the robot. In addition, Demoshop prototypes *just-in-time assistance* similar to the code completion and smart suggestion features that are commonly found in modern computer programming systems.

To assess the effectiveness of Demoshop in improving current kinesthetic teaching-based workflows, we conducted a user study in which participants were asked to author and edit demonstrations using either Demoshop or the programming interface used in our formative study. Our findings demonstrate the potential of introducing authoring and editing aids commonly found in computer programming into the end-user robot programming workflow.

The contributions of this paper include:

- *Design opportunities* for end-user robot programming systems that were identified through a formative study in which users used a programming system popularly used to program collaborative robots in manufacturing domains;
- *Demoshop*[1], an open-source interactive robot programming system that implements various user-centric programming aids, informed by the identified design opportunities, to reduce suboptimalities in task demonstrations provided by users;
- An *empirical understanding* of how Demoshop may improve the robot programming workflow and enhance user experience and task performance for users with or without technical training in robotics and programming.

2. Background and Related Work

2.1. Robot Programming

A key advantage of robotic systems lies in their versatility due to easy adaptation to different tasks without significant redesign to their hardware or control [11]. However, this advantage can only be fully utilized if the robot is easy for an end-user to program. Today, Robot Operating System (ROS), which consists of a variety of open-source software tools, libraries, and robot programming conventions, is the fundamental collaborative framework for robot programming [12]. While ROS provides flexible robot programming capabilities for a wide variety of robotic platforms and algorithmic implementations, the prerequisite knowledge, such as programming languages, kinematics, and computer vision, required to effectively program even simple tasks using ROS and its largely command line-based interaction style makes robot programming using this framework inaccessible to a layperson. This technical barrier partially limits the wide

[1]A system demo video is available at https://youtu.be/qvTMBZkvxwM. The Demoshop project source code is available at https://github.com/intuitivecomputing/demoshop.

adoption of robotic technology outside of research labs and industrial workplaces. Consequently, the question of how to enable effective end-user robot programming has been widely explored.

2.2. End-User Robot Programming

End-user robot programming aims to empower users to program a robot at the level of intricacy and learnability corresponding to their knowledge level. Prior research has explored several programming modalities to increase the accessibility of robot programming for non-programmers [13], including visual programming (e.g., [14, 15, 10, 16, 17, 18, 19, 20, 21, 22, 23]), natural language-based programming (e.g., [24, 25, 26, 27]), programming in augmented (e.g., [28, 29, 30, 31]) or mixed (e.g., [32, 33, 34]) reality, and tangible programming (e.g., [35, 36]).

Among the various end-user robot programming methods, a direct method for specifying robot skills without manually specifying a program is to *demonstrate* the task skill under consideration. Demonstrations may be provided through kinesthetic teaching, where users directly maneuver the robot to demonstrate an intended action (e.g., [37, 38, 39]), video (e.g., [40, 41, 42]), or remote operation of a robot (e.g., [43, 44]). Demonstrations of robot motion paths may be specified as continuous trajectories or as a series of *waypoints* that the robot should traverse.

To increase the applicability of robot demonstrations, research on *Programming by Demonstration* (PbD) has investigated methods to generalize demonstrations to various situations (e.g., [45, 46]), enable robots to learn from multiple demonstrations of the same task (e.g., [47]), and teach robots to learn what not to do [48]. However, by and large, PbD is only as effective as the user-specified demonstrations themselves. Therefore, this work explores authoring and editing tools that seek to help end-users produce effective and efficient kinesthetic demonstrations that can later be used in subsequent learning algorithms or end-user robot programming tools.

2.3. Robot Demonstration Enhancement

To enhance the quality of robot skills based on end-user demonstrations, previous research has investigated computational methods to enable effective robot learning even when demonstrations are sub-optimal (e.g., [49, 50, 51]) and has explored active learning paradigms through which robots can inquire on how to improve their programs (e.g., [52, 53, 54, 55, 56]). Another avenue for handling sub-optimal demonstrations is to enable users themselves to annotate, edit, and refine their demonstrations. For example, prior work has explored how user annotation of task constraints (e.g., [57]) and waypoint reference frames (e.g., [10, 38]) can effectively improve demonstration quality and generalizability. To allow end-users to enhance program quality effectively, prior work has enabled editing of kinesthetic demonstrations through Graphical User Interfaces (GUIs) (e.g., [58, 59, 60]) or mixed reality (e.g., [32]).

Previous research on direct end-user editing and refinement has mostly focused on the modification of individual waypoints in robot motion path demonstrations and program adaptation to different task objects. This work extends prior research by supporting direct editing of waypoints while allowing users to use a straightforward method of demonstration through continuous path recording. This hybrid use of continuous demonstration and discretized waypoint modification leverages their respective strengths, as we describe in the next section. Furthermore, this work adds to previous work on end-user robot programming by integrating proactive assistance commonly found in computer programming interfaces, such as smart suggestions and syntax checking, together with features unique to robot programming, such as contextual visualization, into the robot programming workflow to *assist* end-users in minimizing and removing demonstration suboptimalities.

3. Difficulties in End-User Robot Programming

To understand user needs in end-user robot programming, we conducted a formative study to identify the diversity of experiences and challenges users encounter when working with one of the most widely used end-user robot programming interfaces today [61].

3.1. Formative Study

We instructed eight participants (2 males, 6 females), with ages ranging from 19 to 57 ($M = 31.13, SD = 16.22$), to program a UR5 6-DOF robotic arm to complete four tasks in an hour-long formative study. Participants came from various backgrounds and disciplines. Four participants had advanced degrees. Half of the recruited participants had significant programming experience (3.5 or greater on a 5-point Likert scale, with 1 being having no experience and 5 being having a lot of experience), and the other half had no programming experience. Though participants reported having a lot of experience with technology ($M = 4.38, SD = 0.74$), they had less experience with robots specifically ($M = 1.91, SD = 1.16$) and with programming them ($M = 1.75, SD = 0.89$), reported on 5-point Likert scales with 1 being having no experience and 5 being having a lot of experience.

Participants authored programs for four tasks, each of which highlighted different aspects of the physical manipulation of the robot. All tasks were completed in the same tabletop setting, with locations of task objects marked and preserved across study instances. Participants were required to use kinesthetic teaching to program the robot and were free to use either continuous trajectories, waypoints, or a combination of the two during the programming process. After the experimenter received the participant's informed consent, they provided the participant with a verbal tutorial and demonstration on how to use the PolyScope interface on the teach pendant device that comes with the robot (Fig. 2). The participant was free to ask any questions during the tutorial and during the subsequent practice task in which they were asked to program the UR5 to pick and place a block between pre-specified locations.

After completing the practice task, the participants completed four programming tasks using the programming interface (see Fig. 1 for examples). During this time, the experimenter remained in the room to monitor participant safety. The first task involved building a tower out of three blocks. This task emphasized both the trajectory and precision of the robot's movement. The second task was to program the robot to pick a towel hanging off a rack and to place it on a "clothesline" rope. This task also highlighted trajectory and orientation but did not require high precision. The third task required the participant to program the robot to pour a cup of paperclips into a bowl. This task involved precise control of trajectory, orientation, and speed. Finally, the fourth task involved programming the robot to pick up a cup and place it on a rack, which required the user to maneuver the robot's end effector through several rotations while avoiding collisions with the rack.

After the participant finished programming the robot, the experimenter set the task objects back at their original positions and executed the robot program for the participant. For each task, the participant was free to correct or edit the program as many times as they wanted until they were satisfied, or they could move on to the next task. The participant was stopped 45 minutes into the study regardless of their task progress to complete an open-ended interview and demographics questionnaire. The study was video-recorded, and participants received $10 USD for completion of the study.

3.2. Findings

We identified common themes and concepts that emerged from the study participants' experiences. These findings informed the features we chose to include in our demonstration tool. We summarize our key findings below.

3.2.1. Difficulties with Authoring Robot Motions

We found that there was a tradeoff between authoring robot motion demonstrations using waypoints and authoring using continuous trajectory recordings, which was in line with the findings from prior work [62]. Most participants authored their demonstrations using continuous trajectory recordings because they found that this approach gave them more control over the robot's motion and required less cognitive effort compared to specifying individual waypoints. However, participants often experienced fatigue or had trouble manipulating the 6-DOF arm, together with difficulties in navigating the largely monochrome, disjointed programming interface (Fig. 2), which resulted in jerky motion or unnecessary pauses in demonstrations recorded using continuous trajectory recording. For example, a participant noted, *"Physically delivering [the robot] was harder than I thought it would be. . . wrestling it into that final position was harder than I thought it would be"* (P2). Another participant concluded that *"[continuous trajectory recording], although easy, seems very inadequate for like an application"* (P4).

On the other hand, participants who authored their demonstration using waypoints developed programs with smoother, more efficient movements because their programs did not include any of the jerkiness that accompanied

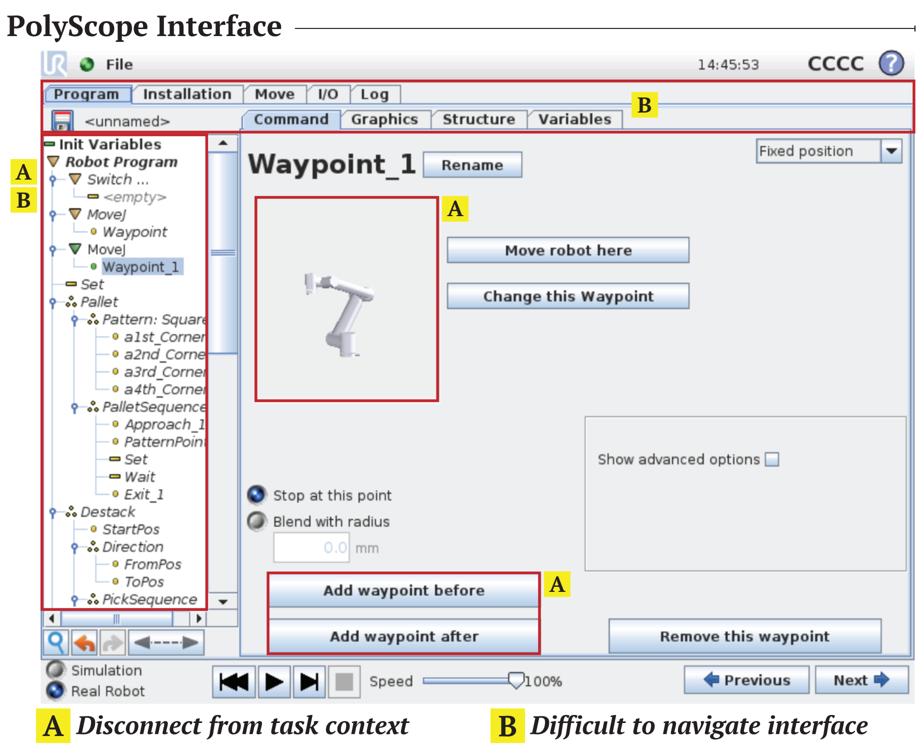

Figure 2: Our formative study revealed that users face various difficulties in navigating the highly partitioned end-user robot programming interface *PolyScope* without environmental and task context.

physically dragging the robot manipulator. Furthermore, participants programming using waypoints were able to more efficiently modify their programs by changing individual waypoints instead of overwriting entire continuous trajectories. However, several participants had difficulty forming appropriate mental models for commands involving waypoints, leading to collisions or incorrect task outcomes, since the motions between waypoints are determined by the robot's motion planner and not by the user. This revealed a tradeoff between the user-friendliness of continuous trajectory recording, especially for users without technical backgrounds, and the efficiency of manual waypoint specification, which allows for more optimal demonstrations at the expense of user control.

3.2.2. Difficulties with Demonstration Editing

Participants mentioned how difficult it was to edit their demonstration without having a detailed visualization of their demonstration. One participant suggested path visualization as a desired system feature: *"If you already have the table set up like this, say if it's a job situation . . . you can visualize the path [the robot is] taking in a picture or something"* (P1). Participants also found that the processes of checking their work before running their demonstration and moving the robot to a particular point in the program where editing may be required were unintuitive. As a result, we found that more participants corrected their program by completely starting over with the programming process rather than modifying the suboptimal or erroneous commands in the program. One participant noted, *"I don't really think I corrected, if you remember I just said 'oh I have to delete it' and just start again. Uhm, probably just a tiny bit more knowledge would have me go 'oh I can just go back a step to fix it"'* (P5), which suggests that the process of editing without restarting the demonstration may not be beginner-friendly.

3.2.3. Difficulties with System Response

The demonstration authoring and editing process was made more difficult for users by lack of system feedback when their demonstration was not working correctly and lack of system assistance during programming and editing. Participants mentioned that it would have been helpful if the system could give them more guidance on what actions and commands would be useful for the demonstration that they were programming or if it could warn them when they were about to use a programming command incorrectly (e.g., when they forgot to instantiate a path command with a recording or inadvertently nested commands). A participant suggested that *"color would be nice, or you know if there was a way to say 'well these are the steps you're gonna be doing' . . . maybe if they were color-coded, or something, cause this is very monotonous and monochrome"* (P2).

Overall, our formative study highlighted the key barriers, such as incorrect mental models of demonstrations and sub-optimal movements introduced into continuous trajectory recording, that prevent users who may not necessarily have technical backgrounds or experience with a particular programming interface from developing effective demonstrations.

3.3. Design Features for End-User Robot Programming Systems

Based on our observations and participant feedback, we identified several design features for end-user robot programming systems to reduce some of the pain points involved in the end-user robot programming workflow commonly used to program collaborative robots today. Below, we list the primary design objectives we derived based on observations from our formative study:

3.3.1. Continuous Path Discretization

Users should be able to easily demonstrate continuous motion trajectories while still taking advantage of the smooth robot motions and ease of editing associated with waypoints. Robot programming systems should therefore represent a user's continuous demonstration while still allowing them to access and edit their demonstration at a waypoint-level resolution.

3.3.2. Mental Scaffolds

End-user robot programming systems should include scaffolds that support the formation and maintenance of users' mental models of their programs, the

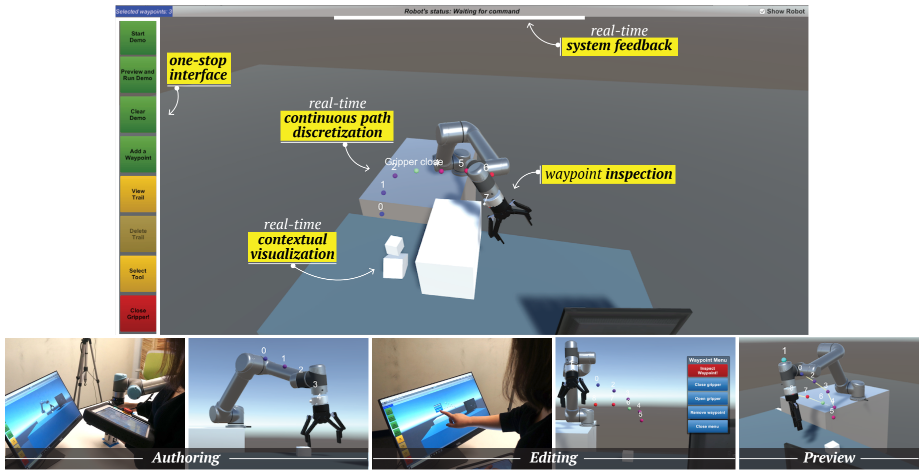

Figure 3: Demoshop provides users with a one-stop interface through which they can use continuous path discretization to develop and modify programs, mental scaffolds to understand the state of their program, and just-in-time assistance to facilitate their programming process.

robot, and the programming interface. End-user robot programming interfaces should center around the 3-D visualization of the user's demonstration contextualized within the environment in which the demonstration was created, which is essential in helping users develop and maintain accurate mental models of the robot programs they create [58]. We found this design goal to be especially critical for waypoint-based demonstrations because the participants in our formative study found waypoints to be confusing without visualization of their spatial locations and surrounding environmental context. Furthermore, end-user robot programming systems should help prevent interface usage errors and offer undisruptive feedback while the user is creating, modifying, debugging, or running a demonstration.

3.3.3. Just-In-Time Assistance

End-user robot programming systems should provide support throughout the programming process, from authoring to execution, similar to the assistance provided in most computer programming systems (e.g., integrated development environments). For example, the system should automatically add relevant annotations, such as gripper actions, to waypoints, thus reducing the extent of post-demonstration editing required of the user, analogous to how a modern computer program editor may automatically insert a closing curly brace.

4. Demoshop: An End-User Tool for Robot Programming by Demonstration

4.1. System Overview

In light of the design goals we uncovered from our observations of user experiences in end-user robot programming, we implemented each of our design features in the form of various programming aids (i.e., continuous path discretization, mental scaffolds, and just-in-time assistance) in Demoshop (Fig. 3). Our current implementation uses the *Universal Robots* UR5 robot manipulator, a popular 6-DOF collaborative robot, as its programming platform. Demoshop's GUI is implemented using the Unity game engine, which includes a physics engine and offers portability to augmented and mixed reality, which are emerging mediums for end-user robot programming (e.g., [28, 32, 33, 63, 30, 31, 60]). We use the MoveIt! framework for motion planning and object tracking, and we integrated our software using Unity and ROS through ROS#. Next, we describe how Demoshop implements easy authoring and editing capabilities through continuous path discretization, mental scaffolds, and just-in-time assistance.

4.2. Continuous Path Discretization

In addition to enabling authoring by manual waypoint specification, which is the common approach to motion demonstration authoring implemented in many end-user robot programming systems today, Demoshop allows users to author continuous demonstrations, which the system automatically parses into a set of waypoints in the robot's joint space, rather than mandating that they manually specify individual waypoints in a path themselves. When users begin demonstrating a continuous path for the robot, Demoshop discretizes the path into a set of waypoints separated by a fixed 10-centimeter interval, which we chose empirically as it worked well for tabletop pick-and-place tasks.

Our discretization process naturally filters out the extraneous information that we found to be common in users' continuous trajectory demonstrations during our formative study. A user's continuous trajectory recording may contain noise in the form of small jitters introduced in the course of maneuvering the robot or pauses from when the user switched their attention from moving the robot to navigating the programming interface. The discretization process abstracts this noisy representation so that only the essential demonstration signal is preserved as waypoints. This approach also provides a balance between execution flexibility (i.e., collision checking) and representation of a user's program structure [57].

4.3. Mental Scaffolds

Demoshop emphasizes contextual visualization, feedback, and visibility at all times to support users in developing accurate mental models of the robot, task environment, programming system, and program state.

4.3.1. Contextual Visualization

Demoshop's interface centers on the 3-D visualization of the robot, the user-specified demonstration, and the surrounding environment. Users are able to choose their desired viewpoint within the 3-D view by using common finger swipe gestures and visualize the real-time movement of the physical robot at all times. They can also toggle the robot visualization on and off or drag windows away to reduce the amount of visual clutter on the screen as needed, which can allow easier viewing of demonstration viewpoints. Sequential information about demonstrations are conveyed by numeric temporal indices above waypoints together with "cold-hot" gradient coloring, with older waypoints being bluer and newer waypoints being redder. If applicable, waypoint gripper actions can be viewed by hovering over or pressing on waypoints. Within the visualization, users may preview waypoints by clicking on them to view their associated robot configuration and may preview the entire demonstration completely in simulation without committing to running the program on the real robot. The visualization shows the surrounding environment and tracks and displays task objects in real time. In our current implementation, we display the environmental context using hard-coded game objects specific to our study environment and assume that the study environment remains static during program execution. For the purposes of our study, we simplify object detection and tracking by tagging all task objects with AR tags. We use a modified version of the ROS package ar_track_alvar for AR tracking.

4.3.2. Feedback and Visibility

For users to have a safe and effective robot programming experience, it is critical that they form and preserve an accurate understanding of what the system is doing. We assist with this process by increasing the visibility of several aspects of the system and providing system feedback when applicable. For any robot movement displayed on Demoshop, the system indicates to the user whether the motion shown on the screen is a simulated preview or whether it reflects the motion of the physical robot in the real world. To further increase awareness, any buttons that will result in the immediate movement of the physical robot are visually distinguished using warning indicators, which include color as well as verbal symbols for color-blind populations. Similarly, the user is always required to view a preview of the robot's motion and to then confirm whether they want to run the demonstration on the physical robot before executing a demonstration. The physical robot's current status is displayed at all times above the demonstration so that users have feedback on the physical robot's actions and on the reason for any errors (i.e., the robot is unable to execute a movement or there is no feasible plan for the demonstration due to collision).

4.4. Just-In-Time Assistance

Demoshop offers users various forms of just-in-time assistance to prevent system misuse and to assist in making the PbD process easier on the user. We highlight examples of how our system offers just-in-time assistance.

4.4.1. Demonstration Authoring Assistance

Since we designed our path discretization process to discretize at uniform spatial intervals, waypoints essential to a user's demonstration may not make it into the final demonstration. One way users can ensure that a waypoint is included in the demonstration is by using Demoshop's manual waypoint specification feature. However, this approach still places the burden of program specification on the user. Therefore, our system tries to aid the user by automatically adding key waypoints from the user's kinesthetic demonstration into the discretized representation. In particular, our system attempts to identify key waypoints by inferring the user's low-level motion intent. We implement this low-level intent recognition with simple checks on whether the user has rested the robot at a specific joint configuration for longer than a three-second period, which tends to occur during critical transition points in the robot's motion path, or if they have opened or closed the robot gripper at any joint configuration during the demonstration. In these cases, the system considers the joint configuration a key waypoint and automatically adds it into the demonstration for the user, while filtering out any automatically generated waypoints that are too close to the key waypoint. If the key waypoint corresponds to a point in the demonstration where the user opened or closed the robot's gripper, the system automatically annotates the waypoint with the respective gripper action during the demonstration so that users do not have to manually specify the gripper action using Demoshop's waypoint editing tools after the demonstration. Demoshop's auto-annotation feature enables end-users to specify gripper actions in the course of a kinesthetic demonstration, rather than specifying gripper actions separately post-demonstration, which is the convention for teach pendant interfaces such as PolyScope.

The system also tries to predict a user's low-level motion intent when they manually add a waypoint. When users manually add waypoints using Demoshop, they must specify the temporal index at which they want to add the waypoint using an input text field. We implement low-level motion intent recognition by having the system predict the temporal index at which to insert the manually specified waypoint based on the index of the waypoint closest to it in 3-D space. As in the case of continuous path discretization, the system offers automatic gripper action annotation for manually specified waypoints.

4.4.2. Error Prevention Assistance

Demoshop takes several measures to prevent execution or interface usage errors. As prior work has suggested that novice programmers may find it difficult to appropriately initialize programs [64] and that a common error in end-user robot programming of manipulation tasks is failing to consider gripper initialization [65], Demoshop performs gripper initialization checks for users. Specifically, if the user is about to run a demonstration that may require the gripper to be open and the gripper is currently closed, the system warns the user and offers assistance in opening the gripper. If the user had turned the robot visualization off earlier in the programming process (e.g., to view the waypoints more easily on the 2-D interface) but then attempts to preview the robot's configuration at a particular waypoint or preview the resulting motion from their program without turning the visualization back on, the system automatically toggles the robot visualization on for them. At all times, the system makes use of bubble effects and shadows on buttons to afford clickability and disables buttons completely when they should not be used.

Together with the key functionalities of continuous path discretization, mental scaffolds, and just-in-time assistance, Demoshop provides common features such as selection tools, editing menus, and waypoint inspection capabilities to enable easy program modification. Its one-stop interface uses color to organize commands rather than relying on dividers or tabs, which can cause disruption to the PbD process (Fig. 2), while also providing verbal alternatives for any information conveyed through color. Our current implementation is programmed to work specifically with the UR5 manipulator, but its integration with ROS enables it to be easily adapted to work with different robots or sensors.

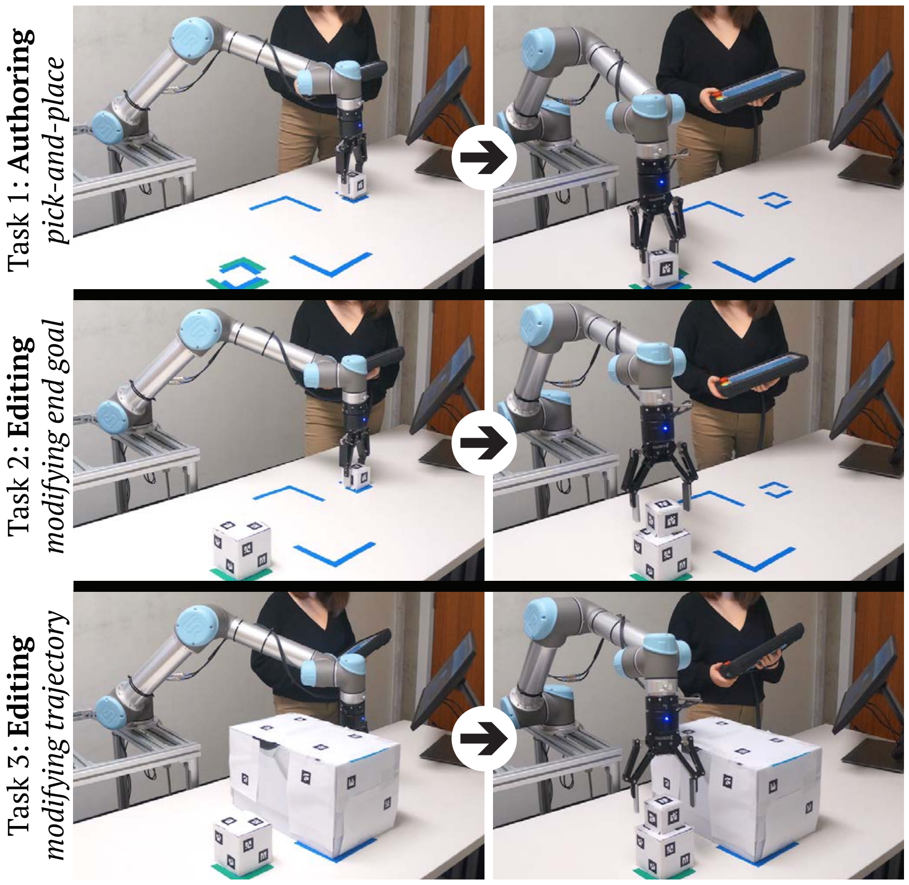

Figure 4: Our user evaluation consists of three pick-and-place tasks. The first task focused on authoring, while the second and third tasks required the user to edit their existing program to meet new task requirements.

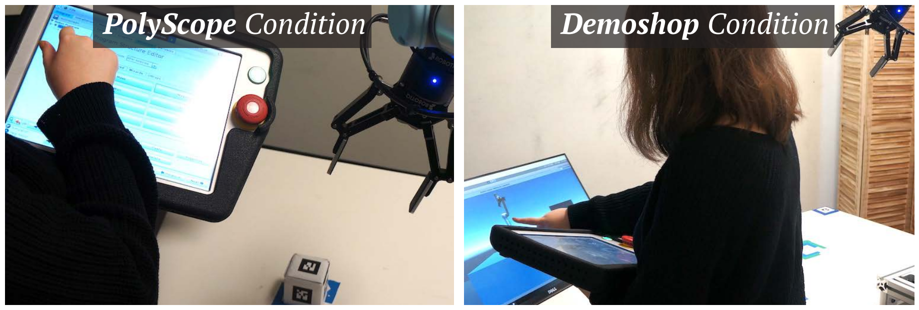

Figure 5: We conducted a between-subjects study in which users were randomly assigned to either a control condition in which they used *Universal Robots*' PolyScope or the experimental condition in which they used Demoshop.

5. Evaluation

Our user evaluation sought to assess users' learnability, usability, and task performance when using Demoshop. Our primary hypothesis was that Demoshop would have a higher learnability and a higher usability and would better aid users in achieving higher task performance when authoring and editing their demonstrations compared to the baseline programming interface (PolyScope) used in our formative study.

5.1. Experimental Tasks and Study Design

We contextualized our evaluation in common pick-and-place tasks that are a core part of various manipulation tasks that collaborative robots are envisioned to assist people with. In particular, we designed three experimental tasks; each of them focused on different aspects of robot program authoring and editing (Fig. 4). For the first experimental task, participants had to author a demonstration for the robot to pick up a block and to place it at a specified location. For the second task, participants had to edit their demonstration from the first task so that the robot placed the block on top of a base object at the same end location as before. This task emphasized editing the end goal of the demonstration. The third task required the participant to edit their program from the second task to avoid having the robot collide with a large box that was placed as an obstacle. This task emphasized editing intermediate waypoints in the robot demonstration.

We varied the interface used by participants for the experimental tasks, and we designed a between-subjects study in which participants were randomly assigned to one of the two conditions (Fig. 5):

- *PolyScope.* In this condition, participants used the PolyScope interface developed by the manufacturer of the robot that was used in our formative study, which is part of the teach pendant that users used to move the robot. Users were limited to adding, previewing, and executing waypoints; path recording; and gripper actions for this study. We provided users with a tutorial video demonstrating how to use these features and structure, preview, and move to specific parts in a program. This baseline condition represents a commonly used method for programming collaborative robots.

- *Demoshop.* In this condition, participants used Demoshop to complete the authoring and editing tasks. In this study, participants interacted with Demoshop through a touchscreen computer. We provided users with a video demonstrating relevant system features.

For both of our study conditions, users had to use the teach pendant that came with the robot to move the robot while authoring a demonstration. Users could move the robot by using the freedrive button on the teach pendant to physically drag the robot or by using the movement commands available on the teach pendant to move the robot in a desired direction.

5.2. Dependent Measures

We used a combination of objective and subjective measures to evaluate users' learnability, usability, and task performance when using Demoshop.

5.2.1. Objective Measures

We measured *practice time* to approximate learnability [66] and used *task progress* and *task time* to assess aspects of task performance.

Practice Time (Seconds). We defined practice time as the length of time participants spent on the practice pick-and-place task after viewing the tutorial video and on trying out the various system features before indicating that they felt ready to move on to the first experimental task.

Task Progress. We estimated task progress by the number of experimental tasks for which the participant indicated to the experimenter that they were ready to move on to the next task. We note that the participant's indication did not guarantee the success, nor the quality, of their demonstrated program.

Task Time (Seconds). Task time was measured as the interval from when the experimenter handed off the task to the participant to when the participant indicated that they were ready for the next task, excluding time spent running the demonstration on the robot. This metric indicated task efficiency.

5.2.2. Subjective Measures

To understand participants' perceptions of and experience with the programming system, we relied on the System Usability Scale (SUS) [67, 68]; a custom scale assessing the participant's mental model of the system; and individual questionnaire items.

System Usability Scale (0–100). The SUS is an established scale for assessing usability comprised of ten questionnaire items probing various aspects of usability, such as user confidence and perceived system integration. A score of 70 or higher is suggested to indicate acceptable, good usability [67].

Mental Model of System (1–5). This scale consisted of six items (Cronbach's α=0.84) that examined the participant's understanding of the robot's capabilities and the effect of their edits, as well as the participant's perceptions of their ability in identifying erroneous aspects of their programs.

In addition, we included individual items on 1-to-5 rating scales regarding the participant's perceptions of adopting the programming system for use in the household and in industry.

5.3. Study Procedure

After obtaining informed consent from the participant, the experimenter provided the participant with a tutorial video for the interface corresponding to their experimental condition. For both conditions, the participant could pause, rewind, or replay the tutorial videos before proceeding to the practice task. During the practice task, the participant was free to ask the experimenter questions about the programming process and view the tutorial video as many times as they desired. Once the participant indicated to the experimenter that they felt comfortable with programming and operating the robot on their own using the interface, they moved on to the experimental tasks. The participant was stopped 50 minutes into the study regardless of their progress. At this point, they were asked to fill out a post-study questionnaire about their experience programming the robot using the interface. After completion of the questionnaire, the experimenter conducted a short open-ended interview to obtain further comments about the user's experience. The study was approximately one hour in length, and participants were compensated with $10 USD.

5.4. Participants

We recruited 16 participants (10 males, 6 females) from the surrounding community. Participants' ages ranged from 18 to 64 ($M = 26.94, SD = 12.80$). Participants reported being moderately experienced with robots ($M = 2.94, SD = 1.53$), highly experienced with technology ($M = 4.19, SD = 0.91$), and experienced with programming ($M = 3.63, SD = 1.09$), but less experienced with programming robots ($M = 2.13, SD = 1.45$) on 1-to-5 rating scales (5 being lots of experience). Participants came from a diverse array of educational backgrounds and industries, including fields outside of engineering such as academic administration, education, psychology, and film.

6. Results

In our data analysis, we first performed normality tests on the obtained data for our measures to test whether or not they were from a normal distribution. If the data followed a normal distribution, we used a student's t-test for analysis; otherwise, we used a Mann-Whitney U test. Our data analysis used two-tailed tests, and we considered $p < .05$ as a significant effect. We followed Cohen's guidelines on effect size and considered $0.1 \leq r < 0.3$ or $d = 0.2$ a small effect size, $0.30 \leq r < 0.50$ or $d = 0.5$ a medium effect size, and $r \geq 0.50$ or $d = 0.8$ a large effect size [69]. Fig. 6 summarizes our main results.

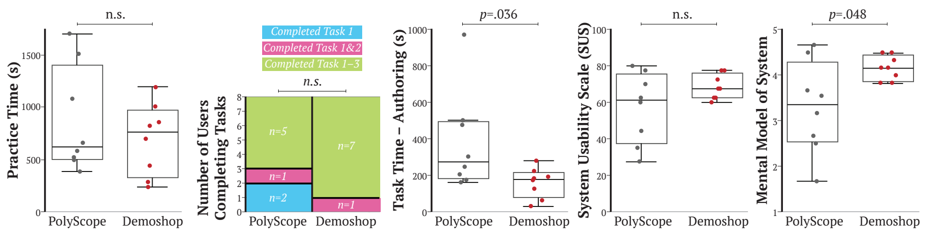

Figure 6: Box and whisker plots of data from our objective measures of learnability, task progress, and task efficiency and data from our subjective measures of usability. The top and bottom of each box represent the first and third quartiles, and the line inside each box is the statistical median of the data. The length of the box is defined as the interquartile range (IQR). The ends of the whiskers are the first quartile minus 1.5 IQR and the third quartile plus 1.5 IQR. For subjective measures, a higher value close to 5 indicates higher perceived usability.

6.1. Objective Measures

Our data showed no significant difference in practice time between participants using the PolyScope ($M = 871.71, SD = 504.25$) and Demoshop ($M = 700.76, SD = 344.75$) interfaces, $t(14) = 0.79, p = .442, d = 0.40$. Although we did not find significant difference in task progress between the two conditions, $X^2(2) = 2.33, p = .311$, there were two participants using the PolyScope interface who only completed the first experimental task (authoring) within the allotted time. Since the first experimental task was the only one that all participants completed, our analysis of task time only focused on the authoring task. A Mann-Whitney U test indicated that users spent significantly longer on the first experimental task in the PolyScope condition ($Mdn = 278.44$) compared to the Demoshop condition ($Mdn = 182.77$), $U = 12.0, p = .036, r = 0.53$.

6.2. Subjective Measures

Our data indicated that there was no significant difference in participants' usability ratings for Demoshop ($Mdn = 67.50$) compared to PolyScope ($Mdn = 61.25$) as measured by SUS, $U = 41.5, p = .328, r = 0.25$. Participants reported having a significantly stronger mental model of the system using Demoshop ($M = 4.17, SD = 0.27$) compared to PolyScope ($M = 3.30, SD = 1.02$), $t(7.96) = 2.34, p = .048, d = 1.17$. Moreover, participants thought that Demoshop was more likely to be adopted in the household ($M = 3.38, SD = 1.06$) than PolyScope ($M = 2.13, SD = 1.13$), $t(13.95) = 2.29, p = .038, d = 1.14$. However, there was no significant difference in participants' opinions on whether the two interfaces (Polyscope: $Mdn = 5$; Demoshop: $Mdn = 5$) would be likely to be adopted in industry, $U = 40.5, p = .382, r = 0.29$.

7. Discussion

In this paper, we presented Demoshop, which supports users in programming robot motion demonstrations via kinesthetic teaching. Based on the user needs we observed from a formative study, we implemented *continuous path discretization* to allow for flexible customization of demonstrations, *mental scaffolds* to help users develop correct mental models of their programs, and *just-in-time assistance* to enhance user experience of the programming process. Results of our user evaluation indicate that users have greater task efficiency while authoring robot programs and maintain stronger mental models of the system when using Demoshop compared to PolyScope.

While our results suggest that Demoshop can help improve task efficiency, task time varies notably among participants. In particular, we have observed that task time may be affected by differing techniques for authoring the demonstration and differing expectations on what constitutes task success among the participants. For example, some participants physically dragged the robot, while others used the movement control on the teach pendant to author the robot's movement. Moreover, our results do not directly reflect task efficiency for editing and debugging. Nevertheless, we found that users were able to navigate and understand how to use Demoshop quickly, making the PbD process more streamlined. Below, we discuss additional findings, implications for end-user robot programming, and the limitations and future directions of our work.

7.1. Improvements for Usability

Overall, post-study interviews revealed that most participants found Demoshop to be straightforward, easy, and intuitive to use during demonstration authoring and editing. One participant recounted, "*I think [the programming process] is pretty straightforward. I don't have any background in Computer Science, but using the system was easier than I thought when I got used to it*" (P10). Participants also found that Demoshop made the editing process to be particularly intuitive, with one participant stating that *"I just think the ability to . . . edit waypoints individually was pretty good cause that like allowed me to really quickly like sub in waypoints, keep existing . . . the idea that I didn't have to . . . reset the waypoints, like, all of them simultaneously . . . I was able to pinpoint which waypoints I wanted to do and that was really nice"* (P11). On the other hand, our results suggested that users did not find Demoshop to be significantly more usable than PolyScope. The average SUS score for Demoshop was below 70, which is a suggested score for acceptable usability [67, 68]. Furthermore, users had comparable learnability and task progress for both systems, indicating that there are still areas that could be improved to make Demoshop more user-friendly. Several participants pointed out that the instability of the real-time tracking of task objects (boxes and blocks) made the contextual visualization less useful, and even caused the system to seem deceptive, so we have extended our system to include more robust object tracking to provide a more reliable contextual visualization.

7.2. Advantages and Applications of Path Discretization

In this work, we adopted path discretization as a system feature to strike a balance between the advantages of continuous trajectory demonstration and manual waypoint specification that we observed in our formative study. One participant mentioned that they liked that continuous path discretization allowed them to use continuous path recording for authoring while still providing them with access to individual waypoints during editing: "*If we are setting up a path from scratch, then definitely [continuous path specification] is better, but if I already have a path and just have to slightly change it, I think adding a waypoint is more convenient*" (P9).

Path discretization has additional advantages and use cases that can further augment end-user robot programming. Discretizing a continuous motion demonstration into a set of essential motion waypoints provides a concise abstraction of a user task, which may be ideal for robot learning algorithms due to its succinctness [70]. In addition, a discretized representation of a continuous demonstration allows the user to provide fine-grained annotation on various portions of the provided demonstration. For example, a user may specify which waypoints may be problematic for a demonstrated task, providing the robot with the opportunity to substitute the erroneous waypoints. In this way, even a partially flawed demonstration can be used by the robot in the learning process.

Discretization can also be harnessed to create generalized pattern-based representations of demonstrations. A discretized representation of a task may be used as a template for the robot, providing the general shape and phases that comprise a particular action. These templates may be used to automatically structure new robot programs to minimize the cognitive burden of initial program demonstration via kinesthetic teaching. In this paper, we focused on making the process of authoring and editing more seamless and intuitive but did not explore how we can help users generalize their demonstration to new scenarios (e.g., [46, 71]), nor did we consider specification of program logic for complex tasks. We plan to explore how we can integrate generalizability with programming aids, such as program templates, to help users develop more interactive, adaptable programs.

7.3. Recognition of Demonstration Intent

Demoshop currently offers programming assistance that predicts the user's low-level motion intent by considering the spatial and temporal configuration of waypoints. We are currently exploring when and how to offer assistance to users in avoiding sub-optimalities (e.g., unnecessary sub-trajectories and joint configurations) and errors (e.g., collisions) through the estimation of higher-level demonstration intent.

Our initial exploration has identified a behavioral signature—a gap between the amount of effort exerted by the user and the resulting change in the motion demonstration (Fig. 7). From our exploratory study, we found that this signature emerges when the user is having trouble moving the robot and expends a sizable amount of energy in attempting to maneuver it to a specific joint configuration (e.g., trying to align the robot gripper with the target object precisely). Our future work will investigate how to recognize this behavioral signature and how the system may automatically intervene by moving the robot to a user's intended position, which may be particularly essential for end-users

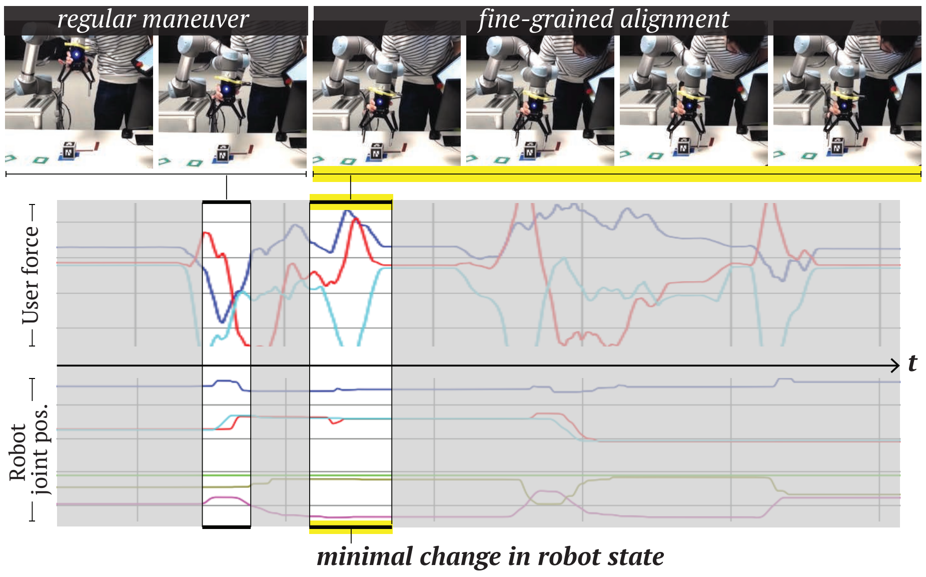

Figure 7: Our initial exploration of sensor data from users' kinesthetic demonstrations has revealed a common behavioral signature that indicates a gulf between user effort and robot motion outcomes.

unfamiliar with the robot, end-users with physical impairments preventing certain movements, or end-users that need to perform more intricate maneuvers. In addition to automatic solutions, we will add features for the end-user to manually override or correct system assistance, as done by previous works that use automatic system assistance in end-user robot programming (e.g., [72, 73, 74]). We have currently implemented a method of authoring assistance in Demoshop that autocompletes object grasping during pick-and-place tasks; effectively, this assistance snaps the gripper to the target object of interest. We will continue to incorporate intelligent aids that draw from human behavioral cues and intent to provide personalized assistance that goes beyond the level of program syntax.

7.4. Error Prevention in End-User Robot Programming

While end-user programming makes programming accessible for people with wide-ranging backgrounds, it introduces potential pitfalls in the form of extensive errors in software created by end-users [75], which can have especially grave consequences for applications in robotics that may have interactions with the physical world and people. Lack of programming background may make end-users more susceptible to relying on workarounds when writing their code, potentially worsened by incorrect evaluations of their own programming ability [76]. Therefore, it is critical to enable end-user programmers to effectively discover sources of failure within their code while minimizing system interference in their programming workflow [77]. Demoshop currently includes mental scaffolds and just-in-time assistance that automatically help prevent system usage and robot execution errors. To further prevent a diversity of errors transparently, we would like to build off previous works on verification of end-user robot programs [78] by incorporating verification modules into Demoshop that can help end-users pinpoint errors in their code and suggest possible fixes to the user (e.g., alternate motion paths).

7.5. Limitations and Future Work

We plan to conduct additional evaluations that better represent the scenarios in which Demoshop may be used. Although we found no significant correlations between participants' programming experience and our study results, participants in our evaluation did have an above average prior experience with programming, including participants from non-STEM fields, who had an average programming experience of 3.6 on a 5-point Likert scale ($SD = 1.14$, 5 being lots of experience). Since Demoshop is meant to be used for end-users from diverse backgrounds, we will focus on participant recruitment from a wider variety of sources. Furthermore, while users with and without technical background both implied that Demoshop was straightforward and easy to use, participants found that Demoshop seemed more likely to make its way into a household environment than PolyScope but not necessarily more likely to be found in industry. This finding suggests that Demoshop may be more approachable for the average user but may be limited in its applicability to more advanced applications such as those found in manufacturing. In this work, our user evaluation focused on tabletop pick-and-place tasks that involved simple robot trajectories and object interactions. We would like to further develop Demoshop's sensing capabilities and evaluate Demoshop for tasks involving more dexterous and precise manipulation in the future to understand how the findings from our current study apply in more complex task scenarios.

Although participants perceived Demoshop as being accessible for home use, we acknowledge that our current implementation requires extensions, such as collection of real-time sensory feedback throughout the programming process and support of more accessible non-industrial robot platforms that are easier to maneuver, before it can be deployed in homes. We emphasize that Demoshop is being continuously developed, and we have made all of the code for Demoshop's implementation openly available for developers together with instructions on how to extend the system's sensory module and modify it to work with different manipulators. As we continue to develop Demoshop, we hope to include more advanced computer vision capabilities so that Demoshop can support manipulation robust to the potential occlusion and clutter found in non-industrial environments. We would also like to further empower end-users in not only performing robot motion specification, but also specification of environmental context (e.g., [10, 79]), so that end-users do not need to rely on expert developers to tag their objects and set up their task environment within Demoshop.

In this work, we used PolyScope as the basis for our formative study and system design and as the baseline for our user evaluation. We chose the PolyScope interface as the focus of our observation as it exemplifies the end-user robot programming workflows adopted in industrial environments today and may therefore serve as a tool in understanding user experiences with current commercial end-user robot programming systems. However, we would like to note that PolyScope does not represent the current state-of-the-art in end-user robot programming and does not include any sensing capabilities, which, as shown in our formative study, limits its usability. In future extensions of Demoshop, we would like to evaluate Demoshop against additional state-of-the-art baselines from academia and industry (e.g., [16]) to obtain a better understanding of Demoshop's relative performance against more advanced programming systems.

Finally, we found that users believed that they had better mental models of the system and their demonstration using Demoshop compared to PolyScope, but in reality still needed comparable practice times and experienced similar task progress between the two interfaces. This finding reveals a discrepancy between how users think they are performing and how they are actually performing when using Demoshop. Our current implementation prototyped just-in-time assistance focused on low-level robot manipulation and error prevention mechanisms specific to the Demoshop interface. By offering more intelligent programming aids and adaptivity to users' level of expertise and to dynamic task environments, we aim to reduce the gap between user experiences and performance outcomes.

8. Conclusion

End-user robot programming by demonstration parallels end-user computer programming in its potential to propel the adoption of automation and programming tools in a variety of workplaces without requiring extensive programming training or expertise. However, developing kinesthetic demonstrations presents its own unique set of challenges, requiring that users have an understanding of robot kinematics and the situated context of motion paths and waypoints. Therefore, it is critical to introduce authoring and editing aids into the kinesthetic teaching workflow to minimize the cognitive burden required of users in specifying optimal robot programs. In this paper, we describe a formative study exploring user experiences in kinesthetic teaching and present a set of design objectives based on our observations for end-user robot programming systems to reduce user difficulties in end-user robot programming. Based on these design guidelines, we implement programming aids in the form of *continuous path discretization*, *mental scaffolds*, and *just-in-time assistance* in the Demoshop interface and demonstrated their effectiveness in improving authoring efficiency and user understanding in end-user robot programming. We plan to continue our exploration of user-centered programming assistance by investigating data-driven approaches towards understanding user programming intents, opportunities for system-level assistance, and user needs. By applying intelligent assistance to end-user robot programming, we aim to progress towards the democratization of robotic assistance for everyday users.

Acknowledgment

The authors would like to thank Annie Mao and Yeping Wang for their help with this work. This work was supported in part by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1746891, the Nursing and Engineering Joint Fellowship from the Johns Hopkins University, and the National Science Foundation award #1840088.

References

[1] B. A. Myers, A. J. Ko, and M. M. Burnett, "Invited research overview: end-user programming," in *CHI'06 extended abstracts on Human factors in computing systems*, 2006, pp. 75–80.

[2] A. J. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrance, H. Lieberman, B. Myers *et al.*, "The state of the art in end-user software engineering," *ACM Computing Surveys (CSUR)*, vol. 43, no. 3, pp. 1–44, 2011.

[3] C. Scaffidi, "Potential financial motivations for end-user programming," in *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2016, pp. 180–184.

[4] V. Wulf and M. Jarke, "The economics of end-user development," *Communications of the ACM*, vol. 47, no. 9, pp. 41–42, 2004.

[5] V. Paramasivam, J. Huang, S. Elliott, and M. Cakmak, "Computer science outreach with end-user robot-programming tools," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 2017, pp. 447–452.

[6] C. Datta, H. Y. Yang, P. Tiwari, I. H. Kuo, and B. A. MacDonald, "End user programming to enable closed-loop medication management using a healthcare robot," *Social Science*, 2011.

[7] E. I. Barakova, J. C. Gillesen, B. E. Huskens, and T. Lourens, "End-user programming architecture facilitates the uptake of robots in social therapies," *Robotics and Autonomous Systems*, vol. 61, no. 7, pp. 704–713, 2013.

[8] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The scratch programming language and environment," *ACM Transactions on Computing Education (TOCE)*, vol. 10, no. 4, pp. 1–15, 2010.

[9] D. Weintrop, A. Afzal, J. Salac, P. Francis, B. Li, D. C. Shepherd, and D. Franklin, "Evaluating coblox: A comparative study of robotics programming environments for adult novices," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–12.

[10] J. Huang and M. Cakmak, "Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts," in *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI.* IEEE, 2017, pp. 453–462.

[11] T. Lozano-Perez, "Robot programming," *Proceedings of the IEEE*, vol. 71, no. 7, pp. 821–841, 1983.

[12] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," *In ICRA workshop on open source software*, vol. 3, 2009.

[13] G. Ajaykumar, M. Steele, and C.-M. Huang, "A survey on end-user robot programming," *ACM Computing Surveys (CSUR)*, vol. 54, no. 8, 2021.

[14] S. Alexandrova, Z. Tatlock, and M. Cakmak, "Roboflow: A flow-based visual programming language for mobile manipulation tasks," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 5537–5544.

[15] D. F. Glas, T. Kanda, and H. Ishiguro, "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, 2016, pp. 303–310.

[16] C. Paxton, A. Hundt, F. Jonathan, K. Guerin, and G. D. Hager, "Costar: Instructing collaborative robots with behavior trees and vision," in *2017 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2017, pp. 564–571.

[17] D. Shepherd, P. Francis, D. Weintrop, D. Franklin, B. Li, and A. Afzal, "An ide for easy programming of simple robotics tasks," in *IEEE 18th International Working Conference on Source Code Analysis and Manipulation*, September 2018, pp. 209–214.

[18] E. Coronado, F. Mastrogiovanni, B. Indurkhya, and G. Venture, "Visual programming environments for end-user development of intelligent and social robots, a systematic review," *Journal of Computer Languages*, vol. 58, p. 100970, 2020.

[19] E. Coronado, D. Deuff, P. Carreno-Medrano, L. Tian, D. Kulić, S. Sumartojo, F. Mastrogiovanni, and G. Venture, "Towards a modular and distributed end-user development framework for human-robot interaction," *IEEE Access*, vol. 9, pp. 12 675–12 692, 2021.

[20] M. Racca, V. Kyrki, and M. Cakmak, "Interactive tuning of robot program parameters via expected divergence maximization," in *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, 2020, pp. 629–638.

[21] G. Huang, P. S. Rao, M.-H. Wu, X. Qian, S. Y. Nof, K. Ramani, and A. J. Quinn, "Vipo: Spatial-visual programming with functions for robot-iot workflows," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–13.

[22] A. Schoen, C. Henrichs, M. Strohkirch, and B. Mutlu, "Authr: A task authoring environment for human-robot teams," in *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, 2020, pp. 1194–1208.

[23] S. Beschi, D. Fogli, and F. Tampalini, "Capirci: a multi-modal system for collaborative robot programming," in *International Symposium on End User Development*. Springer, 2019, pp. 51–66.

[24] N. Buchina, S. Kamel, and E. Barakova, "Design and evaluation of an end-user friendly tool for robot programming," in *2016 25th IEEE International Symposium on Robot and Human Interactive Communication*, August 2016, pp. 185–191.

[25] S. Lauria, G. Bugmann, T. Kyriacou, and E. Klein, "Mobile robot programming using natural language," *Robotics and Autonomous Systems*, vol. 38, no. 3-4, pp. 171–181, 2002.

[26] M. Stenmark and P. Nugues, "Natural language programming of industrial robots." in *ISR*. IEEE, 2013, pp. 1–5.

[27] N. G. Buchina, P. Sterkenburg, T. Lourens, and E. I. Barakova, "Natural language interface for programming sensory-enabled scenarios for human-robot interaction," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2019, pp. 1–8.

[28] Y. Cao, Z. Xu, F. Li, W. Zhong, K. Huo, and K. Ramani, "V. ra: An in-situ visual authoring system for robot-iot task planning with augmented reality," in *Proceedings of the 2019 on Designing Interactive Systems Conference*, 2019, pp. 1059–1070.

[29] W. P. Chan, M. Sakr, C. P. Quintero, E. Croft, and H. M. Van der Loos, "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, 2020, pp. 419–424.

[30] S. Ong, A. Yew, N. Thanigaivel, and A. Nee, "Augmented reality-assisted robot programming system for industrial applications," *Robotics and Computer-Integrated Manufacturing*, vol. 61, p. 101820, 2020.

[31] M. Kapinus, V. Beran, Z. Materna, and D. Bambušek, "Spatially situated end-user robot programming in augmented reality," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2019, pp. 1–8.

[32] S. Y. Gadre, E. Rosen, G. Chien, E. Phillips, S. Tellex, and G. Konidaris, "End-user robot programming using mixed reality," in *2019 International conference on robotics and automation (ICRA).* IEEE, 2019, pp. 2707–2713.

[33] Y. Gao and C.-M. Huang, "Pati: A projection-based augmented table-top interface for robot programming," in *Proceedings of the 24th International Conference on Intelligent User Interfaces.* Marina del Ray, California: ACM, 2019, pp. 345–355.

[34] M. Ostanin, S. Mikhel, A. Evlampiev, V. Skvortsova, and A. Klimchik, "Human-robot interaction for robotic manipulator programming in mixed reality," in *2020 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2020, pp. 2805–2811.

[35] Y. S. Sefidgar, P. Agarwal, and M. Cakmak, "Situated tangible robot programming," in *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI.* IEEE, 2017, pp. 473–482.

[36] A. Kubota, E. I. Peterson, V. Rajendren, H. Kress-Gazit, and L. D. Riek, "Jessie: Synthesizing social robot behaviors for personalized neurorehabilitation and beyond," in *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, 2020, pp. 121–130.

[37] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz, "Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective," in *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction.* Association for Computing Machinery, 2012, p. 391–398.

[38] M. Stenmark, M. Haage, and E. A. Topp, "Simplified programming of re-usable skills on a safe industrial robot: Prototype and evaluation," in *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, 2017, pp. 463–472.

[39] Y. S. Liang, D. Pellier, H. Fiorino, and S. Pesty, "iropro: An interactive robot programming framework," *International Journal of Social Robotics*, pp. 1–15, 2021.

[40] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," in *Conference on Robot Learning.* PMLR, 2017, pp. 357–368.

[41] Y. Wang, G. Ajaykumar, and C.-M. Huang, "See what i see: Enabling user-centric robotic assistance using first-person demonstrations," in *Proceedings of the 15th annual ACM/IEEE international conference on Human-Robot Interaction.* ACM, 2020.

[42] T. Yu, C. Finn, S. Dasari, A. Xie, T. Zhang, P. Abbeel, and S. Levine, "One-shot imitation from observing humans via domain-adaptive meta-learning," in *Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*, H. Kress-Gazit, S. S. Srinivasa, T. Howard, and N. Atanasov, Eds., 2018. [Online]. Available: http://www.roboticsproceedings.org/rss14/p02.html

[43] K. Hsiao and T. Lozano-Perez, "Imitation learning of whole-body grasps," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 5657–5662.

[44] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[45] C. Groth and D. Henrich, "One-shot robot programming by demonstration using an online oriented particles simulation," in *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, Dec 2014, pp. 154–160.

[46] S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto, "Learning and generalization of complex tasks from unstructured demonstrations," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 5239–5246.

[47] K. Ogawara, J. Takamatsu, H. Kimura, and K. Ikeuchi, "Generation of a task model by integrating multiple observations of human demonstrations," *Proceedings 2002 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1545–1550, 2002.

[48] D. H. Grollman and A. Billard, "Donut as i do: Learning from failed demonstrations," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3804–3809.

[49] T. Brys, A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé, "Reinforcement learning from demonstration through shaping," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[50] A. Coates, P. Abbeel, and A. Y. Ng, "Learning for control from multiple demonstrations," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 144–151.

[51] B. Kim, A.-m. Farahmand, J. Pineau, and D. Precup, "Learning from limited demonstrations," in *Advances in Neural Information Processing Systems*, 2013, pp. 2859–2867.

[52] M. Cakmak and A. L. Thomaz, "Designing robot learners that ask good questions," in *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI).* IEEE, 2012, pp. 17–24.

[53] Y. Cui and S. Niekum, "Active reward learning from critiques," in *2018 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2018, pp. 6907–6914.

[54] M. Gervasio, E. Yeh, and K. Myers, "Learning to ask the right questions to help a learner learn," in *Proceedings of the 16th international conference on Intelligent user interfaces*, 2011, pp. 135–144.

[55] M. Racca and V. Kyrki, "Active robot learning for temporal task models," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, 2018, pp. 123–131.

[56] S.-H. Tseng, F.-C. Liu, and L.-C. Fu, "Active learning on service providing model: Adjustment of robot behaviors through human feedback," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 10, no. 3, pp. 701–711, 2017.

[57] C. Mueller, J. Venicx, and B. Hayes, "Robust robot learning from demonstration and skill repair using conceptual constraints," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE, 2018, pp. 6029–6036.

[58] S. Alexandrova, M. Cakmak, K. Hsiao, and L. Takayama, "Robot programming by demonstration with interactive action visualizations," in *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[59] M. Riedl and D. Henrich, "A fast robot playback programming system using video editing concepts," in *Tagungsband des 4. Kongresses Montage Handhabung Industrieroboter.* Springer, 2019, pp. 259–268.

[60] D. Bambušsek, Z. Materna, M. Kapinus, V. Beran, and P. Smrž, "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).* IEEE, 2019, pp. 1–8.

[61] G. Ajaykumar and C.-M. Huang, "User needs and design opportunities in end-user robot programming," in *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, 2020, pp. 93–95.

[62] B. Akgun, M. Cakmak, J. Wook Yoo, and L. Thomaz, "Augmenting kinesthetic teaching with keyframes," in *ICML workshop on new developments in imitation learning*, 2011.

[63] C. P. Quintero, S. Li, M. K. Pan, W. P. Chan, H. M. Van der Loos, and E. Croft, "Robot programming through augmented trajectories in augmented reality," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1838–1844.

[64] D. Franklin, C. Hill, H. A. Dwyer, A. K. Hansen, A. Iveland, and D. B. Harlow, "Initialization in scratch: Seeking knowledge transfer," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 2016, pp. 217–222.

[65] D. Weintrop, D. C. Shepherd, P. Francis, and D. Franklin, "Blockly goes to work: Block-based programming for industrial robots," in *2017 IEEE Blocks and Beyond Workshop (B&B)*. IEEE, 2017, pp. 29–36.

[66] T. Grossman, G. Fitzmaurice, and R. Attar, "A survey of software learnability: metrics, methodologies and guidelines," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2009, pp. 649–658.

[67] A. Bangor, P. Kortum, and J. Miller, "Determining what individual sus scores mean: Adding an adjective rating scale," *Journal of usability studies*, vol. 4, no. 3, pp. 114–123, 2009.

[68] J. Brooke, "Sus-a quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[69] J. Cohen, "Statistical power analysis for the behavioral sciences," *England: Routledge*, 1988.

[70] H. Ravichandar, A. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, 05 2020.

[71] M. A. Rana, M. Mukadam, S. R. Ahmadzadeh, S. Chernova, and B. Boots., "Towards robust skill generalization: Unifying learning from demonstration and motion planning," in *Proceedings of the 2017 Conference on Robot Learning (CoRL)*, 2017.

[72] Y. S. Liang, D. Pellier, H. Fiorino, and S. Pesty, "A framework for robot programming in cobotic environments: first user experiments," in *Proceedings of the 3rd International Conference on Mechatronics and Robotics Engineering*, 2017, pp. 30–35.

[73] Y. S. Liang, D. Pellier, H. Fiorino, S. Pesty, and M. Cakmak, "Simultaneous end-user programming of goals and actions for robotic shelf organization," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6566–6573.

[74] Y. S. Liang, D. Pellier, H. Fiorino, and S. Pesty, "End-user programming of low-and high-level actions for robotic task planning," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN).* IEEE, 2019, pp. 1–8.

[75] M. Burnett, B. Myers, M. B. Rosson, and S. Wiedenbeck, "The next step: from end-user programming to end-user software engineering," in *CHI'06 Extended Abstracts on Human Factors in Computing Systems*, 2006, pp. 1699–1702.

[76] W. Harrison, "From the editor: The dangers of end-user programming," *IEEE software*, vol. 21, no. 4, pp. 5–7, 2004.

[77] T. Robertson, S. Prabhakararao, M. Burnett, C. Cook, J. R. Ruthruff, L. Beckwith, and A. Phalgune, "Impact of interruption style on end-user debugging," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2004, pp. 287–294.

[78] D. Porfirio, A. Sauppé, A. Albarghouthi, and B. Mutlu, "Authoring and verifying human-robot interactions," in *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, 2018, pp. 75–86.

[79] C. Schou, R. S. Andersen, D. Chrysostomou, S. Bøgh, and O. Madsen, "Skill-based instruction of collaborative robots in industrial settings," *Robotics and Computer-Integrated Manufacturing*, vol. 53, pp. 72–80, 2018.