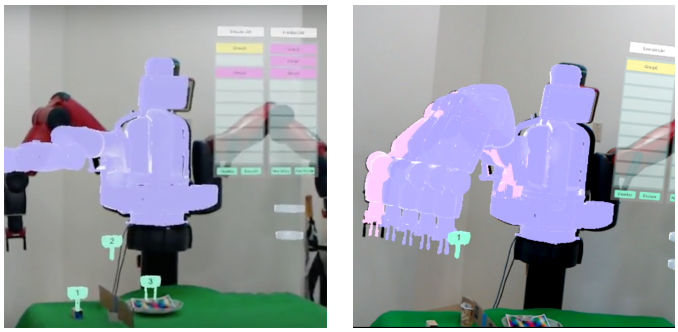
End-User Robot Programming Using Mixed Reality

Samir Yitzhak Gadre[1], Eric Rosen[1], Gary Chien[1], Elizabeth Phillips[1,2], Stefanie Tellex[1], George Konidaris[1]

***Abstract*—Mixed Reality (MR) is a promising interface for robot programming because it can project an immersive 3D visualization of a robot's intended movement onto the real world. MR can also support hand gestures, which provide an intuitive way for users to construct and modify robot motions. We present a Mixed Reality Head-Mounted Display (MR-HMD) interface that enables end-users to easily create and edit robot motions using waypoints. We describe a user study where 20 participants were asked to program a robot arm using 2D and MR interfaces to perform two pick-and-place tasks. In the primitive task, participants created typical pick-and-place programs. In the adapted task, participants adapted their primitive programs to address a more complex pick-and-place scenario, which included obstacles and conditional reasoning. Compared to the 2D interface, a higher number of users were able to complete both tasks in significantly less time, and reported experiencing lower cognitive workload, higher usability, and higher naturalness with the MR-HMD interface.**

***Index Terms*—Robot Programming, Visual Programming, Mixed Reality, Augmented Reality, Human-Robot Interaction**



(a) A screenshot from the MR perspective of a user programming a robot motion. Users are able to specify green waypoints.

(b) After creating the waypoints, users are able to visualize the robot arm motion that is planned through the waypoints.

Fig. 1: Our system's operation, showing the MR interface.

I. Introduction

For robots to become widely used, humans must be able to program their actions. For example, consider the task of binning items. A roboticist might accomplish this task by specifying a series of waypoints in computer code for the robot to visit one by one. If the action needs to be modified, the roboticist would modify the waypoints specified in the code, changing the motion. This method is widely popular, but will not work for end-users. The abstraction of breaking down actions into a series of waypoints could be communicated, but requiring the use of programming languages to specify those waypoints is beyond their scope. Therefore, we will need an alternate method of interfacing with the waypoint action system.

Visual programming is one such methodology. A broad term, visual programming refers to a class of interfaces where objects, variables, classes, etc. are represented as visual shapes, and interacted with accordingly [1]. Visual programming has been a popular tool for programming computers by non-programmers in visual art [2], audio production [3], and education [4] because it allows users to focus on algorithmic thinking, rather than the syntax needed to express their intent.

In the world of robotics, RViz and the Interactive Markers package [5] allow for the creation of visual programming interfaces that control and visualize a robot. Robot researchers have also investigated how effective visual programming is by creating and evaluating their own frameworks [6]. Using our binning task from before, it would be possible to create and modify waypoints visually in a keyboard and mouse interface, but users do not see the waypoints overlayed on the real robotic environment.

In this work, we propose an end-user mixed reality-based visual programming framework for creating and modifying waypoints to create complex, multistep robotic actions (Fig. 1). Users are able to specify and group waypoints together to create primitive motions, as well as adapt these waypoints to perform similar tasks. Our interface allows users to also visualize the entire motion the robot plans to perform through the waypoints, as well as have the robot execute it in real life. We use a commercially available Mixed Reality Head-Mounted Display (MR-HMD), the Microsoft HoloLens [7].

While the MR-HMD at first seems to pose only advantages over other visual programming interfaces by combining the robot workspace and the GUI space for the end-user, there are limitations to the technology that do not make the MR-HMD obviously preferable to a 2D interface. For example, the HoloLens has a limited field of view, so it relies on the user to move around to get a full view of the MR scene. Furthermore, imperfect hand tracking via computer vision makes selection and dragging gestures less reliable than mouse clicks, especially for the novice HoloLens users.

Hence, we conducted a user study with 20 participants and compared the effectiveness of using a MR interface for programming two similar pick-and-place tasks against a 2D visual programming interface. In the first task, participants programmed primitive robot motions to pick up a block and place it on a platform. In the second task, participants adapted their primitive robot programs to sequentially pick-and-place two cubes on different platforms. Our results show that com-

[1] Computer Science, Brown University
[2] Behavioral Sciences and Leadership, United States Air Force Academy

pared to the 2D interface, a higher number of users were able to complete both tasks in significantly less time. Furthermore, users reported experiencing lower cognitive workload, higher usability and higher naturalness with the MR-HMD interface.

II. Related Work

The traditional way to program a robot is to write code. ROS [8] is an extremely powerful middleware environment for roboticists. ROS includes packages to allow programmers to use languages like C++ and Python to interface with robot hardware. However, leveraging the expertise of end-users that lack software engineering skills would help make robots more widely accessible to everyone.

ROS also includes many graphical user interfaces (GUIs), such as RViz [5], for visualization. Rviz can display robot sensor data on a 2D screen and can be connected to MoveIt! [9] motion planners to enable users to program robot movement via keyboard and mouse. 2D interfaces have been shown to be useful for robot programming, but have their own shortcomings regarding immersiveness and intuitiveness. They force users to interpret 3D information on a 2D platform and use control interfaces that do not match how users interact with the world.

Alexandrova et al. [6] created a 2D visual programming language, RoboFlow, to enable end-users to easily program mobile manipulators to perform general tasks in open-environments. Action editing was important to resolve errors [6].

Alexandrova et al. [10] developed a framework that enables users to provide one demonstration of a task to a robot and then use an intuitive monitor and mouse interface to edit demonstrations for adaption to new tasks. Elliott et al. [11] extended this work to allow for the grouping of several poses relative to landmarks or objects in the scene. Having an intuitive interface for non-experts to edit example motions for robots is especially important in LfD, where collecting many examples needed for learning is not always feasible. In such cases, a more preferable method may be to have the human perform one general demonstration of the task, and then have the human adapt parts of their programmed action for new environments. Both Alexandrova et al. [10] and Elliott et al. [11] used a 2D monitor interface to adapt robot programs.

Language is a well-studied modality for programming robots because it is one of the main ways humans communicate commands to each other. Language has been shown to be an efficient way to parameterize a task through descriptions of key items and locations in the environment [12]. Forbes et al. [13] developed a natural language interface to allow users to program robot motions via Learning from Demonstration (LfD). However, using language to program robots results in many limitations. For example, Forbes et al. [13] studied basic object manipulation tasks that required robot action classes that could easily be identified through natural language. However, tasks that rely on actions that are harder to describe can make language a noisy and difficult interface to robots.

Whitney et al. [14] compared a virtual reality (VR) interface against a 2D monitor interface for teleoperating a robot to perform a cup-stacking task. Whitney et al. [14] found that users had a significantly easier time using the VR headset, because they could navigate and interact with the scene by moving their head and hands naturally. This contrasts with keyboard and mouse actions that a typical 2D interface provides. Like the VR headset, the MR-HMD allows users to navigate and interact with the perceived environment using natural actions. However, an additional benefit of the MR-HMD is that it allows the user to also see the real world.

Rosen et al. [15] created an open-source ROS package, ROS Reality, which enables robots to convey intent. The package allows the robot to display its intended path as a holographic time-lapse trail to a user wearing a MR-HMD. Rosen et al. [15] conducted a user study to compare the speed and task completion rate of novice participants using a HoloLens and a 2D monitor interface to determine if a proposed robot trajectory would collide with the environment. They found that the MR-HMD increased accuracy and lowered task completion time. While Rosen et al. [15] showed the promise of using a MR-HMD for visualizing robot motion to non-experts, it did not address how effective MR-HMD is for programming these motions. The HoloLens's limited hand-gesture tracking capabilities pose the possible issue that MR-HMD may not be an effective interface for creating these robot actions.

Walker et al. [16] investigated different MR signalling mechanisms for a drone to communicate with a human. They found quantitative task efficiency was significantly higher when using MR signals than when using physically-embodied signals.

Ni et al. [17] evaluated the effectiveness of augmented reality interface against traditional robot welding programming methodologies. Virtual representations of the robot trajectory were visualized over a 2D video feed of the real robot, enabling novice users to use the augmented reality interface to program new welding paths for the robot to act out. Ni et al. [17] found that the augmented reality interface allowed users to program robot actions more quickly and intuitively, especially for users without prior computer-aided design knowledge. However, augmented reality limits users by forcing them to look away from the robot workspace, which poses safety issues when collaborating with a robot in close-quarters. On the other hand, MR-HMDs allow users to both provide visualizations over the workspace and safely interact with the 3D GUI components using hand gestures.

III. Approach

Our approach is the design of an MR-based interface that allows users to specify waypoints in 3D trajectories. We wanted to test if the notion of adaptation [10, 11] was made more powerful when end-users could edit robot motion in the real world. We therefore created a baseline 2D GUI and a MR-HMD 3D GUI. Our contribution is evaluating the effectiveness of using a MR-HMD interface for programming and editing

robot motion. However, our interface could be integrated with a LfD algorithm to collect example demonstrations from users.

The following subsections address important features of our approach. We discuss waypoints, groups, visualization, and execution. At the lowest level, users program the robot by creating waypoints through which it must move. These waypoints can be collected into groups, which can be visualized in MR and executed in real life.

The waypoint (Fig. 1a) is the fundamental unit of programming in a robot motion. A user adds a waypoint to the tail of a sequence of end-effector poses (3D positions and quaternions) that the robot arm must move through. A waypoint is represented as a hologram of the end-effector, which can be open or closed. For example, if an arbitrary waypoint has state closed, then a command is sent to the gripper to close after it has converged on the waypoint pose. waypoints can be adjusted and deleted by the user.

A group (Fig. 1a) comprises a sequence of waypoints that together form an action. Hence, every waypoint is associated with a group. Groups are split into primitive and adapted groups. Primitive groups are taken to be fundamental building-blocks such as a generic pick-and-place action. After the primitive motions are created, they can be altered—by moving, adding, and deleting individual waypoints—to solve related or more complex tasks. These altered primitives are adapted groups.

Given a group, or a sequence of groups, it is possible to visualize the path that the robot arm will take from waypoint to waypoint (Fig. 1b). Visualization allows the robot to convey how it will move through the specified waypoints in the form of a holographic time-lapse trail. Visualization is key in the editing process as it allows the user to further change an adapted group if they are not satisfied with the trajectory.

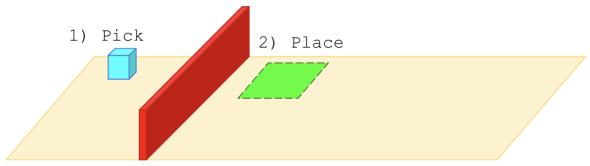
Once a user is satisfied with the adapted groups, they can execute it to make the robot arm move in the real world as per the visualization. We use MoveIt! to make a plan from waypoint to waypoint.

IV. System

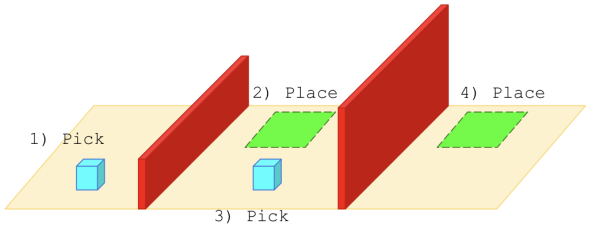
Implementation was split into front-end Unity code for the HoloLens MR-HMD and back-end ROS code for MoveIt! and the Baxter. The 2D monitor baseline used the same back-end as the 3D interface. The front-ends were identical, except that the 2D monitor interface displayed graphics on a screen instead of holograms. Additionally, the 2D monitor interface had a rendering of a point cloud of the workspace (Fig. 3a).

Unity code supported both GUIs. This code made requests to the back-end as waypoints were altered, updated the robot arm movement trails upon receiving motion plans from the back-end, and sent execution requests to the back-end. In order to visualize the robot model and 3D sensor data for the monitor interface, we use ROS Reality [18].

The hologram of the Baxter was created by parsing and rendering a Unified Robot Description Format (URDF). The rendering of Baxter was overlaid on the real robot in a calibration step.



(a) The primitive task. The user had to program the robot to 1) pick up the cube, then bring it over the wall and 2) place it on the platform.



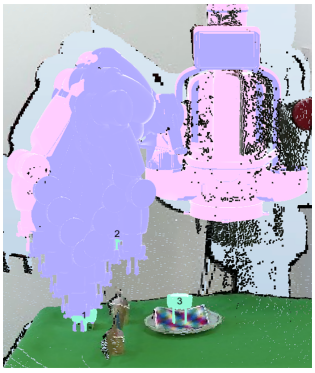
(b) The adapted task. The user had to program the robot to 1) pick up the cube from a new location, 2) place it on the same platform, 3) pick up a second cube, and 4) place on a second platform, all while avoiding walls.

Fig. 2: Our primitive and adapted tasks.

There were two slotted columns attached to the hologram of the robot in the form of a manager menu (Fig. 3b). The right column contained primitive groups and the left column contained adapted groups. A user started by creating a primitive. They could add waypoints or start a new primitive group by clicking the “Add Gripper” and “New Group” buttons, respectively. After creating primitives, the user could drag groups to the left column where they became adapted groups.

The user could add and delete waypoints to alter the primitive so that it was well suited for an adapted task. The left column also acted as a run queue that executed the group actions from top to bottom. By clicking the “Visualize” button, a user was shown a movement trail that the order of the adapted groups implied. By clicking the “Execute” button, the user made the robot execute the desired motion.

The front-end used a websocket client, ROS# [19] to send messages to the back-end. The client connected to ROS bridge [20], which linked the front-end to the Baxter ROS network. The client code marshaled data and sent it to the bridge. The bridge published these messages to the desired ROS topics. If the location, state, or existence of a waypoint was changed, a message was sent to update the back-end’s knowledge of the waypoint. The front-end also sent two other types of messages, one to visualize a motion and one to execute a trajectory. Both of these requests hit MoveIt! code. We used MoveIt! to create a Cartesian plan through the waypoints of the various groups. These plans were sent to the front-end via the Bridge Server. If the end-user wanted to visualize the path, then the motion plan was displayed as a movement trail on the front-end. If the end-user wished to execute the motion plan, MoveIt! communicated with Baxter to actuate its joints.



(a) The robot visualizing its motion through waypoints in our 2D interface. The manager menu is not visible.

(b) A picture of the manager menu included in both the 2D monitor and MR interface. Users can add waypoints, create primitive and adapted groups, and visualize and execute motions.

Fig. 3: (a) Our 2D visual programming interface and (b) manager menu for 2D and MR.

V. Experiment

The aim of our evaluation was to test the hypothesis that the MR interface would be faster and easier to use than the 2D baseline interface. We conducted a user study asking non-experts to program a 7 degree of freedom (7-dof) robot arm to perform a primitive and adapted pick-and-place task using our MR and 2D interfaces. Twenty users—15 male and 5 female—participated in the study. We measured user task completion times, system usability, subjective cognitive workload, and perceived naturalness for both interfaces.

A. Task

Participants completed a primitive and adapted task (Fig. 2). In the primitive task (Fig. 2a), users were instructed to program the robot to pick up a cube, bring it over a wall, and place it on a platform. In the adapted task (Fig. 2b), the participant was asked to program the robot arm to pick up the first block from a new location, move it over the wall, and place it on the same platform. As part of the adapted task, the participant also had to pick up a second block, take it over a wall twice the height of the first, and place it on a second platform.

B. Interfaces

We compared two interfaces:

- **Monitor Interface**: Participants used a 2D monitor, keyboard, and mouse to program and edit robot motions (Fig. 3). Using this interface, a user navigated the virtual scene and oriented the camera view in 3D spacing using a combination of right mouse clicks, drags, and keyboard button presses. To interact with the waypoint programming interface, users were able to use the left-mouse button. This 2D monitor interface framework is similar to that of Alexandrova et al. [10] and Elliott et al. [11].
- **Mixed Reality Interface**: Participants used a HoloLens to program and edit the robot motions. Users were able to translate and rotate their perspective by simply moving their head as they normally would. To interact with the programming interfaces, users perform tapping and dragging hand gestures on the GUI components shown in Fig. 1.

C. Study Design

This study used a within-subjects design where all participants performed both experimental programming tasks using both interfaces. Participants were randomly assigned to the order in which they used each interface to complete the programming tasks. Furthermore, experimenters counterbalanced the order of the two interfaces across study participants. For each interface, participants performed first the primitive task, then the adapted task.

D. Experimental Procedure

Participants began the study by reviewing the informed consent information. After they agreed to participate in the experiment, the experimenters explained the two tasks that the users would be programming the robot to perform. In addition, the experimenters explained the various components of the programming interfaces as described in Section III. After the users understood the task and how the programming interfaces worked, users were randomly selected to first complete tasks on either the 2D monitor or MR interface. For each interface, users were taught the controls before completing the tasks. Then, they moved on to the adapted task. In the adapted task, users were able to access the successful motions they had programmed in the primitive task and edit them to meet their new goal. To record times, experimenters gave a 3-2-1 countdown to begin the task, at which point users were timed for programming the necessary motions up to the point of them having the robot execute the final programmed motion. For both the primitive and adapted task, users were given five attempts, where an attempt would be considered a failure if a) the robot failed to sequentially place the cubes on the correct platforms or b) if the robot arm ran into a wall while executing a motion. If they succeeded, their best time of the five attempts was recorded. After completing both the primitive and adapted tasks, users filled out all usability, workload, and naturalness surveys for that interface, and then moved on to the other interface condition.

E. Dependent Measures

Our objective dependent variables were the task completion rate of completing each of the the programming tasks and the task completion times for both the primitive and adapted tasks.

If users were not able to create a successful robot motion to complete the primitive task in a given interface, then that user did not attempt the adapted task in that interface. Thus, we report task completion rate in the primitive task for each interface as the number of users who were able to have at least one successfully programmed robot motion out of their five tries divided by the total number of participants. Furthermore,

we report task completion rate in the adapted task for each interface as the number of users who were able to successfully program a robot motion out of five tries divided by the number of people who had successfully programmed a robot motion in the primitive task. The subjective dependent measures included the NASA Task Load Index (NASA-TLX) [21], the System Usability Scale (SUS) [22], and a survey intended to measure user perceptions of the naturalness and usefulness of each interface.

NASA-TLX: the NASA-TLX is a subjective measure of perceived cognitive workload. Participants provide ratings of their perceived workload while completing a task on six sub-scales: mental demand, physical demand, temporal demand, effort, frustration, and performance. Five of the sub-scales are rated from 0 (very low demand) to 100 (very high demand) and the performance sub-scale is rated from 0 (perfect performance) to 100 (failure). For this experiment, the weighted measure of paired comparisons among the sub-scales was not included. See Moroney et al. [23] for a discussion. The subjective cognitive workload score was calculated by taking the average of the six sub-scale scores.

SUS: the System Usability Scale assesses overall system usability by asking participants to rate ten statements on a 7-point Likert-type scale from "Strongly Disagree" to "Strongly Agree". The statements cover aspects like system complexity, consistency, and cumbersomeness among others. The SUS scores are converted to a scale of 0 (low usability) to 100 (high usability). See Sauro [24] for more detail on SUS scoring.

Perceived Naturalness Survey: the Perceived Naturalness Survey was derived to measure how natural using the interface felt to each participant. The naturalness of an interface has been discussed as a necessary component of building good user interfaces [25, 26] and can lead to positive outcomes like better learning in computer mediated environments [27], but little work has been done on how to directly measure how "natural" an interface feels to a user. Prior work has suggested that components of interface naturalness include natural mapping—the ability of a system to map its control to changes in the mediated environment in a natural and predictable manner [25], as well as control, maneuverability, direct connections, and salience of input and feedback to name a few. This survey was a first attempt in measuring those concepts of naturalness. The survey included items like, "It was easy to understand how changes in the interface would result in changes in the real-world," "Using the interface, I felt like I had full control over the robot," "The interface felt predictable," and "The interface felt natural," among others. Participants responded to these statements using a 7-point Likert-type scale that ranged from "Strongly Agree" to "Strongly Disagree." Scores on this measure were calculated by taking users average response across the 11 items.

F. Hypotheses

Overall, we expected users to perform better (lower task completion times, higher task completion rate, lower workload, higher usability and preference) on the MR interface than the 2D monitor interface in both the primitive and adapted tasks.

H1: Users will quantitatively perform better on the MR interface than the 2D monitor interface for completing the primitive task. Better performance is categorized as a) lower task completion time and b) higher task completion rate.

H2: Users will quantitatively perform better on the MR interface than the 2D monitor interface for completing the adapted task. Better performance is categorized as a) lower task completion time and b) higher task completion rate.

H3: Users will report higher subjective impressions of the MR interface than the 2D monitor interface for completing the primitive task. Better subjective performance is categorized as a) lower reported workload, b)higher usability, and c) higher perceived naturalness of the interface.

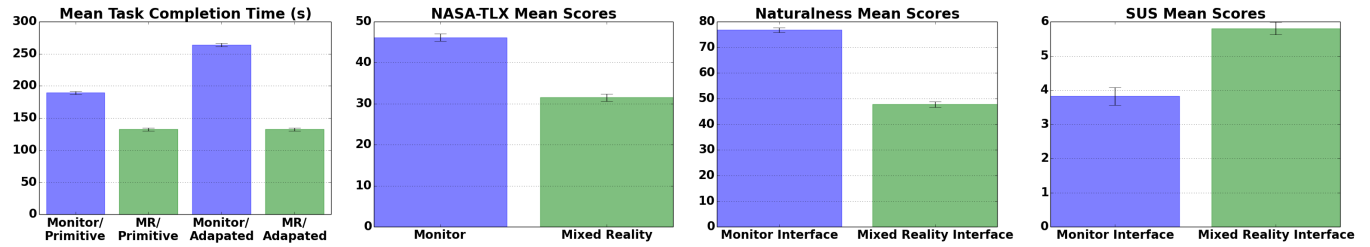
G. Results

1) Primitive and Adapted Task Completion: Two paired samples t-tests were conducted to evaluate differences in task completion times between the 2D monitor and MR interfaces for both the primitive and adapted tasks (Fig. 4a). There was a statistically significant difference in mean task completion times (in seconds) between the 2D interface (M=189.44, SD=105.33) and the MR interface (M=132.50, SD=92.40) on the primitive task, $t(17)$=2.77, p=0.013, Cohen's d=0.57. There was also a significant difference in mean task completion times between the 2D (M=263.98, SD =123.88) and MR (M=145.28, SD=108.43) interface for the adapted task, $t(17)$=5.656, p <0.001, Cohen's d=1.02. Additionally, all participants (100%) were able to complete the primitive and adapted tasks using the MR interface. On the other hand, fewer (N=18) participants were able to complete the primitive task using the 2D interface. Since these participants were not able to complete the primitive task, they were unable to adapt that plan in the adapted task. Thus, they did not complete either. Using the MR interface, participants completed both the primitive and adapted tasks faster than in the 2D interface and with higher task completion rate. Thus, Hypotheses 1 and 2 were supported.

2) Workload: Statistically significant differences were also found between subjective ratings of mental workload between the 2D and MR interfaces (Fig. 4b), $t(19)$=4.07, p =0.001, Cohen's d=0.93. Participants reported significantly lower cognitive workload using the MR interface (M=31.54, SD=15.69) than the 2D interface (M=46.13, SD=15.59).

3) Usability: Participants also reported significantly higher usability scores for the MR interface (M=5.8, SD=0.67) as compared to the 2D interface (M=3.82, SD=1.37), $t(19)$=7.73, p <0.001, Cohen's d=1.83 (Fig. 4d).

4) Naturalness of the interface: Finally, participants also reported that interacting with the MR interface (M=47.75, SD=21.73) felt significantly more natural than using the 2D interface (M=76.75, SD=15.88), $t(19)$=8.72, p <0.001, Cohen's d=1.52 (Fig. 4c). Taken together, the results of the subjective measures of workload, usability, and perceived naturalness support Hypothesis 3.



(a) Task completion times for each interface/task.

(b) NASA-TLX scores for monitor and MR

(c) A graph of the results from our Naturalness Survey.

(d) A graph of the results from the SUS Survey.

Fig. 4: A figure of selected quantitative and qualitative results. Standard error is shown on the bars.

5) Order effects: Additionally analyses were conducted to test whether the order in which participants interacted with each interface biased the results. There were a few results affected by order. First, if participants interacted with the 2D interface after interacting with the MR interface, they rated the 2D interface significantly lower on usability than participants who interacted with 2D interface first. This result may suggest that participants were disappointed or frustrated by the 2D interface after having first interacted with the MR system, mean difference=23.31. $t(18)$=3.17, p=0.008, Cohen's d=1.46. A look at the frustration subscale of the NASA-TLX revealed that participants did indicate significantly higher frustration scores for the 2D interface as compared to the MR interface, $t(19)$=4.13, p=0.001, Cohen's d=1.27, mean difference=31.5. Similar results were found for the perceived naturalness measure. If participants interacted with the 2D interface after interacting with the MR interface they rated the 2D interface significantly lower on how natural the interface felt than participants who interacted with the interfaces in the reverse order, $t(18)$=4.29, p=0.002, Cohen's d=2.00, mean difference=1.98. The inverse was also true for perceive naturalness of the MR interface. If participants interacted with the MR interface second, then they rated the MR interface significantly higher on the measure of naturalness of the interface than participants who interacted with the MR interface first, $t(18)$=2.45, p=0.033, Cohen's d=1.14, mean difference=0.69.

VI. Discussion

Overall our results strongly support the potential to use MR interfaces for novice users to program and adapt robot motions. In our user study, participants were significantly faster and better able to complete the programming tasks using the MR interface than the 2D keyboard and mouse interface. In addition, participants reported lower levels of cognitive workload (e.g., frustration, effort, mental, temporal, and physical demand) in the MR interface. Users reported that the MR interface felt more usable and natural when completing the programming tasks. Our findings on the order of completion may also reveal the subjective appeal of using the MR interface as well. If participants used the 2D interface after interacting with the MR interface they rated the 2D interface significantly lower in usability and naturalness than if they had not been exposed to the MR interface first. The opposite was also true, if participants used the MR interface after having been exposed to the 2D, they rated the MR interface significantly higher than if they had not previously worked with the 2D interface. These findings may suggest that when participants had an opportunity to compare the 2D interface to the MR interface, they greatly favored the MR interface or became frustrated with the 2D interface after having worked with the interface that was easier to use. These findings are promising for implementing programming interfaces that are quick and easy for non-roboticists to use, may represent a large improvement over the more cumbersome keyboard and monitor interfaces, and represent a clear path forward in the development of next generation robot programming interfaces.

VII. Conclusion

This paper contributes to the study of using MR-HMDs for end-user robot programming. We describe how our system works, including how different parts of our programming interfaces are visualized and interacted with by users working on a 7-dof robot arm. We evaluated the effectiveness of using our MR system for programming and editing robot motions in a primitive and adapted pick-and-place task against a 2D monitor and keyboard equivalent. Our study showed that users were able to more quickly and accurately program and edit robot motions using MR than the 2D baseline. In addition, users found that our MR interface required less workload, was easier to use, and was more natural to use in comparison to the baseline. This study shows the large promise MR has for making robot programming more accessible to all end-users.

In the future we plan to investigate extensions to our interface for LfD, to allow users to program robot demonstrations with MR interfaces. We also plan to explore semantic mapping MR interfaces, which allow users to use MR to annotate the environment for a robot.

References

[1] B. Myers, "Visual programming, programming by example, and program visualization: a taxonomy," in *ACM sigchi bulletin*, vol. 17, no. 4, 1986, pp. 59–66.

[2] Blender Online Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Blender

Institute, Amsterdam, 2018. [Online]. Available: http://www.blender.org
[3] C. '74, "Max software tools for media," https://cycling74.com/products/max/, 1985–2018.
[4] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, "Scratch: Programming for all," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, 2009.
[5] D. Gossow, A. Leeper, D. Hershberger, and M. Ciocarlie, "Interactive markers: 3-d user interfaces for ros applications [ros topics]," *IEEE Robotics & Automation Magazine*, vol. 18, no. 4, pp. 14–15, 2011.
[6] S. Alexandrova, Z. Tatlock, and M. Cakmak, "Roboflow: A flow-based visual programming language for mobile manipulation tasks," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5537–5544.
[7] Microsoft, "Hololens," 2016, https://www.microsoft.com/en-us/hololens [Accessed: 2018].
[8] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system." in *Open-source software workshop of the Int. Conf. on Robotics and Automation*, 2009.
[9] W. Garage, "*MoveIt!*" 2007, https://moveit.ros.org/ [Accessed: 2018].
[10] S. Alexandrova, M. Cakmak, K. Hsiao, and L. Takayama, "Robot programming by demonstration with interactive action visualizations." in *Robotics: science and systems*, 2014.
[11] S. Elliott, R. Toris, and M. Cakmak, "Efficient programming of manipulation tasks by demonstration and adaptation," in *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2017.
[12] T. Kollar, S. Tellex, M. Walter, A. Huang, A. Bachrach, S. Hemachandra, E. Brunskill, A. Banerjee, D. Roy, S. Teller, and N. Roy, "Generalized grounding graphs : A probabilistic framework for understanding grounded language," in *Journal of Artificial Intelligence Research*, 2013.
[13] M. Forbes, R. Rao, L. Zettlemoyer, and M. Cakmak, "Robot programming by demonstration with situated spatial language understanding," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2014–2020.
[14] D. Whitney, E. Rosen, E. Phillips, G. Konidaris, and S. Tellex, "Comparing robot grasping teleoperation across desktop and virtual reality with ros reality," in *International Symposium on Robotics Research*, 2017.
[15] E. Rosen, D. Whitney, E. Phillips, G. Chien, J. Tompkin, G. Konidaris, and S. Tellex, "Communicating robot arm motion intent through mixed reality head-mounted displays," in *International Symposium On Robotics Research*, 2017.
[16] M. Walker, H. Hedayati, J. Lee, and D. Szafir, "Communicating robot motion intent with augmented reality," in *ACM/IEEE International Conference on Human-Robot Interaction*, 2018, pp. 316–324.
[17] D. Ni, A. Yew, S. Ong, and A. Nee, "Haptic and visual augmented reality interface for programming welding robots," *Advances in Manufacturing*, vol. 5, no. 3, pp. 191–198, 2017.
[18] D. Whitney, E. Rosen, D. Ullman, E. Phillips, and S. Tellex, "ROS Reality: A Virtual Reality Framework Using Consumer-Grade Hardware for ROS-Enabled Robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.
[19] Siemens, "*ROS#*," 2017, https://github.com/siemens/ros-sharp, [Accessed: 2018].
[20] C. Crick, G. Jay, S. Osentoski, and O. Jenkins, "Ros and rosbridge: Roboticists out of the loop," in *International conference on Human-Robot Interaction*, 2012, pp. 493–494.
[21] N. H. P. R. Group, "Task load index (nasa-tlx) v1. 0 computerised version," *NASA Ames Research Centre*, 1987.
[22] J. Brooke, "Sus-a quick and dirty usability scale," *Usability evaluation in industry*, pp. 189–194, 1996.
[23] W. Moroney, D. Biers, F. Eggemeier, and J. Mitchell, "A comparison of two scoring procedures with the nasa task load index in a simulated flight task," in *Aerospace and electronics conference*, 1992, pp. 734–740.
[24] J. Sauro, "Measuring usability with the system usability scale (sus)," 2011, http://www. measuringusability. com/sus.php [Accessed: 2018].
[25] P. Skalski, R. Lange, and R. Tamborini, "Mapping the way to fun: The effect of video game interfaces on presence and enjoyment," in *Proceedings of the Ninth Annual International Workshop on Presence*. Cleveland State University Cleveland, OH, 2006, pp. 63–64.
[26] S. Ramm, J. Giacomin, D. Robertson, and A. Malizia, "A first approach to understanding and measuring naturalness in driver-car interaction," in *International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 2014, pp. 1–10.
[27] S. Bailey, "Getting the upper hand: Natural gesture interfaces improve instructional efficiency on a conceptual computer lesson," Ph.D. dissertation, University of Central Florida, 2017.