End-user Programming of Robot Trajectories through Natural Communication

Saman Shahbazi
Computer Science Department
Vrije Universiteit
Amsterdam, the Netherlands

Kim Baraka
Computer Science Department
Vrije Universiteit
Amsterdam, the Netherlands
k.baraka@vu.nl

***Abstract*—Human-robot interaction is evolving toward a more intuitive and natural mode of communication that allows non-expert end-users to program robots. In line with this vision, we introduce PRESS, a system for interactively demonstrating and refining robot trajectories safely in simulation before transferring them to the real robot. PRESS goes beyond existing interactive programming methods by integrating rich sign and spoken language interactions and combining demonstrations and iterative adjustments through different modes of operation. In a pilot user study, we assessed the usability of the sign language system as well as the accuracy of the trajectories with a simulated Pepper robot. Our preliminary empirical results suggest high system usability and the viability of sign language for robot programming. We view this work as a stepping stone towards a more flexible and personalizable use of personal and collaborative robots. This paper includes an open-source implementation of PRESS.**

***Index Terms*—End-user programming; Human-robot interaction; Multi-modal interfaces; Programming by demonstration.**

I. Introduction

Robots require expert knowledge and programming skills to operate and program [1]. To allow non-experts end-users to have more control over robots that collaborate or work for them, there have been recent trends in programming robots through interaction with end-users to allow for more flexible and personalized robot behaviors. By far the most common way for robot programming through interaction is through kinesthetic demonstrations [2]. One limitation of kinesthetic demonstrations is that it may be difficult for untrained users to provide such demonstrations. Another limitation is that these methods do not allow for refining or correcting robot behaviors in the event of an imperfect demonstration, or evolving user preferences. Furthermore, because these demonstrations occur in the real world, the resulting robot trajectories may be unsafe and cause irreversible consequences (e.g., breaking an object). To address these limitations, we introduce a multi-modal social interface that allows non-expert end-users to both *demonstrate and refine robot trajectories in simulation* until a satisfactory trajectory is reached. The interface integrates both *speech and gestures (partially based on American Sign Language (ASL))* for greater flexibility and accessibility.

To have more effective and meaningful relationships with humans, work in the field of human-robot interaction has focused on endowing robots with human-like communicative abilities. These efforts move away from input devices such as mice, keyboards, and joysticks into more intuitive and natural forms of communication [3]. Many researchers note in this context that for humans to have long-term relationships with robots, they must be interested in the interactions that occur between them [4] [5] [6]. We claim that inclusion plays a similar role in the formation of meaningful relationships. There are currently many frameworks that offer speech as a means of interacting with robots, as well as frameworks in which robots are taught to perform sign language to be able to interact with deaf users [7]. However, to the best of our knowledge, no frameworks exist that use both speech and sign language, such as ASL, as inputs to interact with a robot. The World Federation of the Deaf estimates that 72 million people worldwide use sign language but the prevalence of such robot interfaces is inadequately low. Sign language as an interaction modality has other benefits as well. A drawback of speech-based programming is that it is prone to misrecognition in noisy environments [8], such as industrial workplaces. Employing sign-language may enhance distance based communication with robots in such noisy environments. By incorporating sign language into our social interface, we hope to introduce an often overlooked form of communication into robot interaction modalities as well, to increase the likelihood and acceptance of robots in the deaf community.

Concretely, this work contributes *PRESS (Programming Robots Effectively through Speech and Sign language)*, a novel robot programming interface relying exclusively on natural communication. It includes vision-based hand tracking to provide demonstrations, and speech and sign language to refine demonstrated trajectories. The sign-language commands includes six static hand gestures and 12 dynamic gestures classified through two distinct neural network architectures trained on a dataset collected by the researchers consisting of 4,400 labeled static gesture instances and 2,580 video fragments of dynamic gestures instances. The interface uses a robot simulation by design, allowing users to safely refine trajectories before they are sent to the real robot (e.g., using sim2real transfer methods [9], which are out of the scope of this paper). We conducted a preliminary evaluation of PRESS in a pilot study with 10 human participants to assess the accuracy of the programmed trajectories on a simulated Pepper robot, as well as PRESS's usability. An open-source ROS-2

implementation of PRESS[1] is also included in this paper for reproducibility and community building purposes.

II. Related Work

A variety of programming methodologies have been investigated in order to make robot programming more accessible to end users [8]. These methods include natural language based programming (e.g., [10] [11] [12]), visual programming (e.g., [13] [14]), demonstration based programming (e.g., [15] [16]), augmented reality programming [17], and virtual reality programming (e.g., [18] [19]).

Programming by demonstration (PbD) and visual programming have been some of the most common methods for programming robots without explicitly developing a program to do so. Generally, in visual programming, end-users are able to manipulate the robot through interaction with either graphical representations of the robot or by the creation of block based programs. Recent work [20] [21] [22] has combined visual programming with kinesthetic programming, where desired trajectories are first kinesthetically demonstrated by end-users, and subsequently perfected or modified for other tasks through a graphical user interface (GUI). In [13], natural language was also incorporated into the programming of the robot to expand the interaction possibilities. While kinesthetic teaching has many advantages, such as its simplicity and intuitive control of the robot, its strength can also serve as its weakness. The physical manipulation of the robot can be demanding for some users, and it requires the robot to be present and within proximity.

Other studies that are more comparable to ours (e.g., [23] [24] [25] [22]) have employed depth cameras to record user demonstrations, as opposed to recording kinesthetic demonstrations. Landa-Hurtado et al. [24] developed a framework in which a Kinect camera tracks the operator's right hand to program the trajectory. The relative position of the left hand to the right hand was used to control the system's mode. To program trajectories, Moe and Schjlberg [25] used a Kinect camera as well as an accelerometer from a smartphone to extract position and orientation information. Gestures performed by the hand holding the accelerator enabled interaction with the robot through gestures such as "focus" and "click". In this study, the use of an accelerometer enabled rapid interaction with the robot, but it also introduced unnecessary extra accessories. Another study [23] used the Kinect camera and a marker held by the operator to track demonstrations. Similar to the combined visual and kinesthetic programming mentioned before, this study also incorporated a GUI to modify created trajectories.

PRESS primarily falls under the category of programming by demonstration (PbD). However, we also employ natural communication for interacting with the robot, and we visualize trajectories in simulation. In addition to the integration of both speech and sign commands and a richer vocabulary of commands, PRESS's interface distinguishes itself from the discussed PbD studies in the following ways: Firstly, all the (camera based) studies use a depth camera, whereas we use an off-the-shelf webcam, hence more accessible. Secondly, most frameworks use the GUI itself to modify trajectories post programming, whereas we take a completely hands-off approach and use natural communication to program, interact, and modify trajectories with a GUI only used for visual display.

[1]https://github.com/stdcout42/nao-move

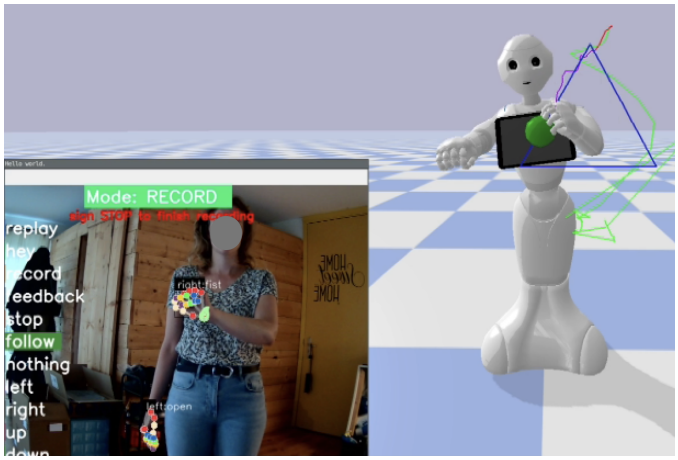

Fig. 1: A user tracing a pre-drawn trajectory using PRESS. Left: webcam feed. Right: simulation environment.

III. PRESS (Programming Robots Effectively through Speech and Sign Language)

This section presents an overview of the technical approach behind PRESS. To be able to achieve the objective of programming (trajectories) by demonstration and the subsequent refinements to the trajectory, we developed several modes: Imitate, Record, Feedback, Replay, and Move (see Appendix). The end user is able to interact and manipulate the modes through either sign language or speech. The modes and the keywords required to activate them are described in the Appendix Table I. The current mode is always displayed to the user in the webcam feed (Figure 1). A user story for the proposed system may be as follows.
The user:

- uses sign language to initiate record mode.
- programs a trajectory in simulation.
- uses sign language to end the recording.
- uses sign language to activate *replay* mode. The robot (Pepper) replays the programmed trajectory in simulation.
- finds the programmed trajectory unsatisfactory and initiates feedback mode through sign language.
- modifies the trajectory using sign language.

The robot has now saved the programmed trajectory, which can be replayed on demand.

At a high level, the system can be abstracted into four main components: a Visual Input component, a Speech Input component, a Simulation Controller, and a graphical user interface (GUI).

A. Visual Input component

This component is responsible for processing webcam input. It uses body part coordinates extracted from the input to generate messages for the Simulation Controller. These messages are: coordinates for trajectory programming, static hand gestures, dynamic sign language commands, and directional commands to move the robot base with. A fully connected neural network (FCNN) and a long short-term memory (LSTM) network were used to recognize hand gestures and sign language, respectively. In addition to conveying messages to the Simulation Controller, this component also displays the current mode of the robot in the webcam feed window.

When the robot is in its default (imitation) mode, it will attempt to move its end-effector (left-hand) when the visual component detects a closed right-hand (fist) in its feed. The frame rate of the webcam video loop determines the rate at which the right-hand coordinates are sent out. Below, some key parts of this component are briefly described:

- Static hand gestures classification: We collected 4,400 labeled hand gesture instances to classify six static hand gestures. From this data, 21 landmarks were extracted from each hand, preprocessed, and fed into an FCNN (Figure 6a in the Appendix).
- Dynamic sign language classification: To train and classify 12 sign language commands, 2,580 video fragments with 30 frames each were collected. 45 landmarks were extracted from each frame, preprocessed, and fed into an LSTM (Figure 6b in the Appendix) using this data. The 45 landmarks included 21 landmarks per hand (the landmarks of both hands are required for the sign language system) and three additional landmarks relativized to the hip: one from the nose and two from each wrist.
- Move interface: The move interface (Figure 8 in Appendix) consists of a number of interactive direction circles which are projected on the camera feed for the user to interact with. If the robot mode is set to *move* mode, this component outputs a direction whenever a fist is detected on one of the direction circles by the static classifier.

In summary, this component accepts video frames from a camera as input and transmits coordinates, sign language commands, and movement directions as outputs, as well as informational messages to the user via the webcam feed.

B. Speech recognition component

This component is responsible for classifying speech commands. It takes the first two words of a recognized sentence, and if the first word is “hey” (the keyword for the robot to react to), it will communicate the second keyword to the robot. An example would be “hey record”, to change the robot’s mode to record mode.

C. Graphical User Interface component

This component is responsible for controlling the simulation environment and was used primarily in the user study. The GUI does not require any inputs, but it can control the simulation environment (spawning objects, drawing shapes) by sending messages to the Simulation Controller. The GUI was used to conduct the experiments in our study, where it synchronized with the simulator to start, stop, record, time, analyze, and save the data from the experiments. Figure 7 in the Appendix illustrates the GUI.

D. Simulation controller

Th Simulation Controller serves as the link between the end-user and the robot. It receives all of the outputs generated by the preceding modules and uses this information to set the mode of the robot, move the robot’s end-effector or base, and change the simulation environment. The function of this component is determined by the mode the robot is currently in (Figure I in the Appendix lists the modes).

1) Robot modes: All modes (except for *imitation* and *replay* mode) can be initiated through sign language and speech, by respectively signing in American Sign Language (ASL) or speaking the names of the modes. To initiate mode changes, the signed or spoken syntax has to be as follows: ”hey [command]”, where ”hey” (in sign language or spoken) indicates the beginning of a command. ”Hey record,” for example to start recording a trajectory. To exit a mode, the command ”stop”[2] has to be used. When a mode change is initiated, the application will notify the user (through voice) of the change. Below a brief description of each mode:

- Imitation mode: In this default mode, the robot uses inverse kinematics to compute joint angles for the Pepper robot. During movement, a green line is drawn in simulation to indicate the change in position of the end-effector.
- Record mode: This mode is responsible for recording programmed trajectories. Once initiated the user has three seconds before the recording commences. During the duration of the recording, the simulation line indicating the change in position of the end-effector turns red. Each received coordinate will be stored in memory and can be altered in feedback mode.
- Feedback mode: Upon activation of this mode, the user can use a variety of commands to modify the trajectory. These modifications consist of rotations, translations and scaling operations (Table II in the Appendix).
- Replay mode: This mode is activated by using the command ”play” and it will force Pepper to replay the saved trajectory.
- Move mode: This mode will initiate the *move* interface described in the Visual Input Component subsection III-A.

E. Implementation

1) Libraries: The full list of libraries used to make this project possible can be found in the project’s repository[3]. Several key libraries worth mention here are MediaPipe (for skeleton tracking), ROS2-foxy (to help with creating ROS-2

[2]The signs can be viewed at https://github.com/stdcout42/nao-move/tree/main/docs/flipped_demos

[3]https://github.com/stdcout42/nao-move

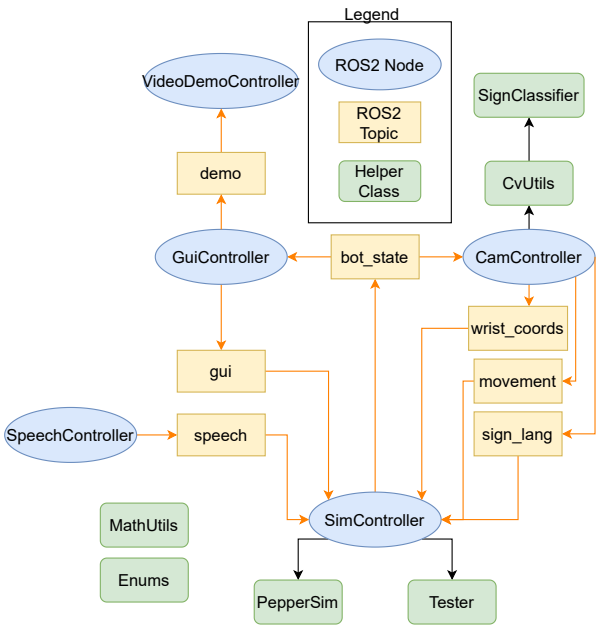

Fig. 2: PRESS architecture: ROS nodes and attached modules.

applications), VOSK (for offline speech recognition), PyBullet (for inverse kinematics calculations), qiBullet (for Pepper simulation), Kivy (for the GUI) and Tensorflow (for the hand gestures and sign language classifiers).

2) MediaPipe: As described in Section III-A, the coordinates (landmarks) of the user's body parts were required to translate the user's demonstrated motion and to classify gestures and sign language. To that end, MediaPipe's Holistic solution[4] was used as it provides landmarks for the face, pose and hands simultaneously. The pose estimation was specifically required because it provided landmarks with depth values relative to the midpoint of the hips.

3) ROS2: The Robot Operating System 2 (ROS2) framework was utilized with Python 3. The most important components of the system can therefore be divided into ROS2 nodes and their helper modules. Figure 2 displays the ROS2 nodes and other classes that make up the entire framework. Topics are an essential component of the ROS network as they provide a means for nodes to exchange messages with one another. Seven ROS2 topics and three ROS messages were developed for this project to act as communication routes between the nodes. The most important ROS node, SimController, is subscribed to all topics except bot state and demo. The actions taken by the SimController upon receiving messages are described in Section III-D1. More details about ROS nodes and topics can be found in the repository.

IV. Evaluation

A user study was conducted to assess the efficacy of the produced system's sign language and the accuracy of the end-effector programming. Ten people aged 23 to 65 were invited to participate in the experiment, two of which had prior programming skills. Informed consent was obtained from each participant for inclusion in the study. The experiment was performed in the PyBullet simulation environment. Appendix VI-D contains a complete system specification and configuration for the experiment.

[4]https://google.github.io/mediapipe/solutions/holistic.html

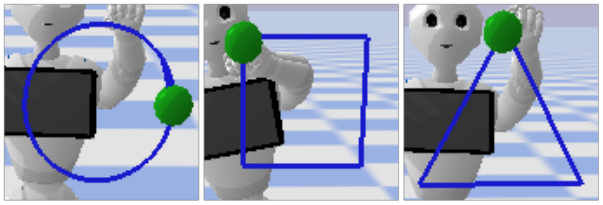

Fig. 3: The three pre-drawn trajectories the participants had to program by following along as closely as possible with the green marker.

A. Study design

In part 1, the accuracy of the trajectory programming was measured by calculating the Root Mean Squared Error (RMSE) relative to pre-drawn trajectories. Second, the time was recorded for the subject to initiate record mode as well as end record mode after programming the trajectory. In part 2, the efficacy of the move module was measured by timing the time it took the subjects to initiate the mode and move the robot's base to a specified marker. Finally, in part 3, the ease of moving the end-effector was measured by loading a table, a ball, and a tray into the environment and recording the number of attempts it took for the subjects to hit the ball into the tray.

Following the conclusion of the experiment, we used the System Usability Scale (SUS) [26] to measure general system usability as well as individual questionnaire items to learn about participants' experiences with the system.

The three parts of the experiment are described in more detail below.

1) Part 1: trajectory programming: The accuracy of the end-effector movement was first measured in part one by having participants trace along three different pre-drawn trajectories (see Figure 3) and then computing the RMSE of the trajectories. Second, the time it took participants to initiate and complete the activity using sign language was noted.

The explicit steps were as follows: the tester would start the experiment via the GUI, at which point the program would wait for the participant to sign "hey." When a "hey" was detected by the framework, the program started the timer. The participant's next task was to sign "record." Upon successfully doing so, they had to move the robot's end-effector to a green marker located on the pre-drawn trajectory. When the end-effector came within 10 cm of the marker, the marker would start to trace the pre-drawn trajectory. The participants were instructed to stay as close as possible to this marker and to sign "stop" whenever they were done tracing the trajectory (when the green ball stopped moving). Upon receiving the "stop"

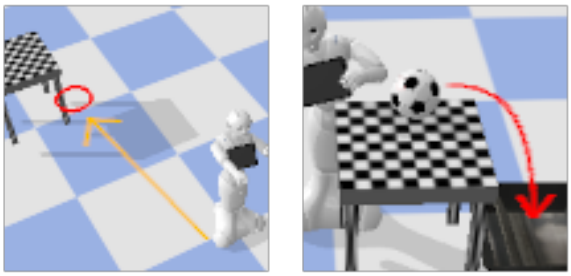

Fig. 4: (Left) Participants had to initiate move mode and move the robot to the red circle. (Right) Participants had to hit the ball with the robot's end-effector (left hand) into the tray on the ground.

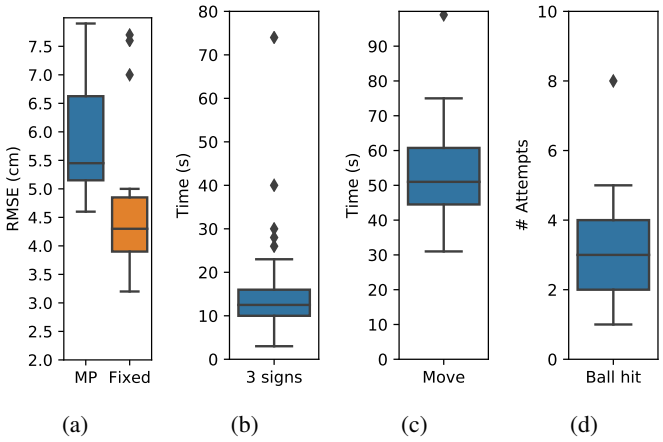

Fig. 5: (a) RMSE for trajectories drawn by participants using extracted depth coordinates, and using fixed coordinates, respectively. (b) Time it took the participants to perform three sign commands. (c) Time it took the participants to sign three commands + move Pepper's base by 2.1 m. (d) Number of participant attempts until ball hit tray.

command, the RMSE and the time it took the participants to complete the task were recorded.

In addition, since the participants were required to program trajectories that were projected on a plane, it was anticipated that the MediaPipe pose estimator, which can generate 3D coordinates relative to the user's hip, would provide consistent estimated depth values. To test this, six participants were randomly assigned to program trajectories using 2D coordinates obtained from MediaPipe and a fixed depth value, while the remaining four participants were assigned to program trajectories using MediaPipe's 3D coordinates.

2) Part 2: move interface: The move feature was tested by timing how long it took the subjects to initiate move mode (via sign language), move the robot to a specified marker 2.1 m away (in simulation space), and finally exit move mode (by signing "stop").

3) Part 3: end-effector efficacy: Part three assessed the efficacy of end-effector control by having subjects attempt to hit a ball placed on the table in front of the pepper robot into a tray. The number of attempts it took for the subjects to be able to hit the ball into the tray was recorded.

V. Results and Discussion

A. Summary of results

1) Trajectory programming: The participants traced a total of thirty pre-drawn trajectories (three tracings per person). The mean RMSE of the thirty trajectories was 5.2 cm ($SD = 1.35$). The mean RMSE for the four participants that completed their tracings with MediaPipe's depth coordinates was 5.9 cm ($SD = 1.1$), compared to 4.7 cm ($SD = 1.3$) for the six participants who completed their tracings with a fixed depth. Figure 5a shows the the box-plot for the two different settings.

Additionally, the mean time it took the participants to sign the three required commands to program a trajectory (as explained in Section IV-A1), was 16.5 s ($SD = 13.2$) (Figure 5b).

2) Move interface: The mean time it took the subjects to activate *move* mode, move the robot's base to a marker 2.1 m in the distance, and to deactivate *move* mode was 56.1 s ($SD = 18.0$) (Figure 5c).

3) End-effector efficacy: It took the participants four attempts on average ($SD = 4$) to be able to hit the ball into its designated tray (Figure 5d).

4) System usability: The average SUS score for this study was 83, which indicates acceptable, good usability [26].

B. Quantitative observations

Part one of the experiment revealed that the mean error for all programmed trajectories was 5.2 cm. This is nearly an order of magnitude greater than other PbD frameworks with 3D-capable cameras [23] [27], but it is important to note that the experiments conducted to measure this error varied greatly between studies. We suspect that the scaling of coordinates when they were mapped to the simulation environment was an additional cause of the large disparity.

Additionally, the average time for the participants to sign the three commands, "hey," "record," and "stop," was 16.2 s. First, it has to be taken into account that the subjects were not familiar with the signs and sometimes had to look up the sign or ask for instructions. This also resulted in the participants occasionally making errors in the sign language demonstrations, necessitating repeated attempts. However, this may also be telling of the initial learning curve that a sign language framework requires for users who do not familiar with the sign language. Lastly, some participants forgot to signal "stop" at the conclusion of the task, resulting in a longer recorded time. Despite the likelihood that these times are longer than those of other modalities, the results indicate that sign language is a viable mode of communication when interacting with robot frameworks.

Second, a two-sample t-test showed a statistically significantly lower mean error ($p = 0.014$) for the group in which

the depth value of the coordinates was fixed (RMSE = 4.7 cm ($SD = 1.3$)) as compared to the one with depth estimation (RMSE = 5.9 cm ($SD = 1.1$)). In conclusion, even though the participants programmed trajectories that were on a plane to mitigate the 3D shortcomings of a single-lens camera, there is evidence that using MediaPipe's (pose) depth estimation can still introduce noise into the trajectories. It is important to note that the sample size for this experiment is small, and as shown in 5a, there were outliers that necessitate the conduct of a larger study in order to draw appropriate conclusions.

In the second part of the experiment, participants took an average of 56 s to move the robot base 2.1 m away. This included signing the commands "hey," "move," and "stop." The main barrier here appeared to be that users had difficulty reading the directions displayed in the interface. In addition, the interface directions were relative to the robot, which may have added to the confusion.

In the final phase of the experiment, it took the participants an average of approximately four attempts to hit the ball into the designated tray. In this scenario, it appeared that participants were once again confused as to which direction to swing the end-effector. In addition, it took the participants multiple attempts to apply the correct amount of speed in order to strike the ball with sufficient force.

C. Survey results

From the survey, it was revealed that the *move* interface was deemed the least intuitive by the participants. In addition to the previously mentioned difficulties when using the move module, namely that it was difficult for some users to read the interface and that the movement was relative to the robot itself, we suspect that there was simply too much information on the screen for the users to simultaneously comprehend (simulation window, webcam feed with the move interface).

Furthermore, the survey revealed that the participants were most impressed by the system's speed in recognizing a closed hand and the subsequent imitation. When asked about the sign language system, 22% of the users believed that the signs could be more intuitive and 67% of the users believed it was easy to use the sign language commands to interact with the robot.

D. Limitations and improvements

1) Depth: Naturally, due to the single-lens, single-image-sensor camera used in this framework, it is nearly impossible to create (initial) trajectories that require depth-dependent motion. However, with the use of the feedback system, these initial trajectories can be modified (for instance, rotated) to meet the criteria for a path containing depth. While the feedback system permits users to create virtually any path imaginable, it adds unnecessary complexity to the framework. Therefore, one of the most significant enhancements is a system that can map locations in the depth dimension more accurately.

2) Reference studies: Using a standard webcam, our (limited) study has measured the usability of sign language and the accuracy of programmed trajectories. In order for the project's conclusions to be more meaningful, comparisons must be made with other results, particularly in terms of interfacing. Experiments in the future can evaluate the usability of various modalities, such as the use of a joystick. Moreover, despite the fact that experiment participants reported the sign-language to be easy to use, it remains unclear whether the use of natural communication enhances and sustains human-robot relationships over the long term.

3) Implementational improvements: The interface that performed the worst during the experiment was the move interface, for reasons that have already been mentioned. An alternative to the approach that was used for this project is to implement sign language to also incrementally control the base of the robot.

Furthermore, the implementation can be improved in numerous ways, which are addressed in the project's repositories. These include, but are not limited to: combining classifiers to improve efficiency; setting a fixed rate for the input motion capture; adding timestamps to the recorded trajectory; and developing a system to save and load multiple trajectories.

VI. Conclusion

We introduced PRESS, a novel system for programming robot trajectories by demonstration and refinement using natural communication. Using a regular RGB camera, a microphone, and a capable computer, the system enables users with no programming experience and those who cannot necessarily benefit from speech recognition to effectively program trajectories in a simulated environment before transferring the resulting trajectories to the real robot.

Our preliminary evaluation results suggest that sign language is a viable communication modality for robot interfaces and that the system was deemed usable by users (SUS score of 83). Future research could concentrate on testing sign language interfaces with users who truly benefit from them, such as people from the deaf community, through a user-centered design approach to inform command and gesture choices, training methods, and other interaction considerations. The sign language system may also benefit users who must interact with a robot in noisy environments, such as collaborative robots in industrial settings, or in environments were speech is not an option, e.g., underwater robots.

Acknowledgments

K. B. would like to acknowledge the contributions of Ojas Patel, Rakesh Johny, Rohan Rao, and Roshan Rajan to preliminary implementations of the ideas in this paper, as well as Prof. Andrea L. Thomaz (Socially Intelligent Machines Lab at the University of Texas at Austin) for her mentorship.

References

[1] G. Biggs and B. Macdonald, "A survey of robot programming systems," 01 2004.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[3] S. Iba, C. Paredis, and P. Khosla, "Interactive multi-modal robot programming," vol. 1, pp. 161 – 168 vol.1, 02 2002.

[4] M. Ligthart, M. Neerincx, and K. Hindriks, "Core elements of social interaction for constructive human-robot interaction," *CoRR*, vol. abs/2110.04054, 2021.

[5] I. Alcubilla Troughton, K. Baraka, K. Hindriks, and M. Bleeker, "Robotic improvisers: Rule-based improvisation and emergent behaviour in hri," in *Proceedings of the 2022 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '22, p. 561–569, IEEE Press, 2022.

[6] M. E. U. Ligthart, M. A. Neerincx, and K. V. Hindriks, "Memory-based personalization for fostering a long-term child-robot relationship," in *HRI 2022: ACM/IEEE International Conference on Human-Robot Interaction, Sapporo, Hokkaido, Japan, March 7 - 10, 2022* (D. Sakamoto, A. Weiss, L. M. Hiatt, and M. Shiomi, eds.), pp. 80–89, IEEE / ACM, 2022.

[7] S.-Y. Lo and h. Huang, "Realization of sign language motion using a dual-arm/hand humanoid robot," *Intelligent Service Robotics*, vol. 9, 10 2016.

[8] G. Ajaykumar, M. Steele, and C.-M. Huang, "A survey on end-user robot programming," *ACM Comput. Surv.*, vol. 54, oct 2021.

[9] K. Dimitropoulos, I. Hatzilygeroudis, and K. Chatzilygeroudis, "A brief survey of sim2real methods for robot learning," in *International Conference on Robotics in Alpe-Adria Danube Region*, pp. 133–140, Springer, 2022.

[10] S. Lauria, G. Bugmann, T. Kyriacou, and E. Klein, "Mobile robot programming using natural language," *Robotics and Autonomous Systems*, vol. 38, pp. 171–181, 03 2002.

[11] M. Stenmark and P. Nugues, "Natural language programming of industrial robots," pp. 1–5, 10 2013.

[12] N. Buchina, P. Sterkenburg, T. Lourens, and E. Barakova, "Natural language interface for programming sensory-enabled scenarios for human-robot interaction," 10 2019.

[13] S. Alexandrova, M. Cakmak, K. Hsiao, and L. Takayama, "Robot programming by demonstration with interactive action visualizations," 07 2014.

[14] C. Datta, P. Tiwari, I.-H. Kuo, and B. Macdonald, "End user programming to enable closed-loop medication management using a healthcare robot," 12 2011.

[15] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz, "Keyframe-based learning from demonstration," *International Journal of Social Robotics*, vol. 4, pp. 343–355, Nov 2012.

[16] D. Müller, C. Veil, M. Seidel, and O. Sawodny, "One-shot kinesthetic programming by demonstration for soft collaborative robots," *Mechatronics*, vol. 70, p. 102418, 10 2020.

[17] J. Lambrecht, M. Kleinsorge, M. Rosenstrauch, and J. Kruger, "Spatial programming for industrial robots through task demonstration," *International Journal of Advanced Robotic Systems*, vol. 10, p. 1, 05 2013.

[18] L. Hamon, P. Lucidarme, E. Richard, and P. Richard, "Virtual reality and programming by demonstration: teaching a robot to grasp a dynamic object by the generalization of human demonstrations," 01 2011.

[19] C. Maragkos, G.-C. Vosniakos, and E. Matsas, "Virtual reality assisted robot programming for human collaboration," *Procedia Manufacturing*, vol. 38, pp. 1697–1704, 01 2019.

[20] G. Ajaykumar, M. Stiber, and C.-M. Huang, "Designing user-centric programming aids for kinesthetic teaching of collaborative robots," *Robot. Auton. Syst.*, vol. 145, nov 2021.

[21] Y. S. Liang, D. Pellier, H. Fiorino, and S. Pesty, "iropro: An interactive robot programming framework," *CoRR*, vol. abs/2112.04289, 2021.

[22] J. Huang and M. Cakmak, "Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts," in *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '17, (New York, NY, USA), p. 453–462, Association for Computing Machinery, 2017.

[23] H.-D. Zhang, S.-B. Liu, Q. Lei, Y. He, Y. Yang, and Y. Bai, "Robot programming by demonstration: a novel system for robot trajectory programming based on robot operating system," *Advances in Manufacturing*, vol. 8, 05 2020.

[24] L.-R. Landa-Hurtado, F.-A. Mamani-Macaya, H.-R. Valenzuela-Coloma, J. Fuentes Maya, and R.-F. Mendoza-Garcia, "Kinect-based trajectory teaching for industrial robots," 03 2014.

[25] S. Moe and I. Schjølberg, "Real-time hand guiding of industrial manipulator in 5 dof using microsoft kinect and accelerometer," pp. 644–649, 08 2013.

[26] A. Bangor, P. Kortum, and J. Miller, "Determining what individual sus scores mean: Adding an adjective rating scale," *J. Usability Studies*, vol. 4, p. 114–123, may 2009.

[27] A. Vakanski, F. Janabi-Sharifi, and I. Mantegh, "An image-based trajectory planning approach for robust robot programming by demonstration," *Robotics and Autonomous Systems*, vol. 98, 10 2017.