Towards Personalizing Shared Autonomy with Human-in-the-Loop Robot Programs

Anonymous Author(s)
Affiliation, Address
anonymous@email.edu

Abstract: As robots develop increasingly advanced capabilities and become more broadly used in the home, a key challenge remains in developing systems that effectively allow people to share control with a robot partner. We introduce the Shared Autonomy Toolkit, which enables users to specify when and where to assume control in a robot program, enabling them to customize task execution to their preferences. In a user study with 8 participants across various skill levels, we investigate how diverse users interact with this system and identify patterns for when humans choose to insert themselves into the loop in a long-horizon manipulation task. Our findings highlight that integrating users into the loop in shared autonomy improves task success rates, perceived sense of control, and user experience, illuminating patterns that can inform how robots learn to adapt across user preferences. Our framework also serves as a scalable platform for collecting high-quality, user-driven intervention data, advancing the development of robots that learn to adapt to humans.

Keywords: Shared Autonomy, Human in the Loop, Systems for Teleoperation

1 Introduction

A key objective in robot learning is to build robots that are capable of operating effectively around diverse users in human-centric environments. Real-world deployment reveals a core tension, where full teleoperation is tedious and repetitive, and autonomous operation is unable to reliably perform complex tasks in diverse and unstructured environments with limited data. This motivates the use of shared autonomy, which integrates human input with robot autonomy to facilitate task completion. While most shared autonomy systems decide when to hand off control based on system-side criteria, it is crucial to equip users to make these decisions. People may prefer to intervene at seemingly suboptimal times to preserve agency, accommodate preferences, or respond to uncertain situations — patterns shaped by their experience, needs, and abilities.

Despite advances in shared autonomy, the ability for end users to flexibly program when and how to assume or relinquish control remains underexplored. End-user programming can enable users, even those without robotics experience, to customize robot behavior for their own needs and contexts [1]. This capability is especially important given current trends in robot learning, which often rely on massive datasets to train generalizable policies. However, recent work shows that a small number of high-quality demonstrations — especially those involving timely human interventions and corrections, lead to more robust and effective policies than large quantities of lower-quality data [2]. Furthermore, prior work has shown that fine-tuning robot foundation models on a small amount of human-provided teleoperation data is essential for producing behaviors that generalize [3]. The key, then, is to facilitate high-quality, human-in-the-loop data collection that allows users to supervise and intervene precisely when robots are most likely to fail.

To address this gap, we introduce a new system for creating modular, reusable robot programs with human-in-the-loop control. Our browser-based interface enables users to record teleoperated

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

demonstrations in-person or remotely, write high level scripts using tools and functions that allows for user input, and monitor program execution with real-time feedback. Users can take control of the robot at preferred points in time, pause the program to verify the state of the robot, and iteratively refine the programs to achieve their intended goals.

We evaluate this system through a user study with 8 participants of diverse skill levels. Our findings uncover when and why users across varied experience levels choose to intervene during shared autonomy, highlighting nuanced preferences for control and autonomy. This work introduces an interactive system for developing robot programs that empowers users to define and flexibly take control during real world tasks, presents insights into user preferences and behaviors in shared autonomy environments, and demonstrates the feasibility and benefits of user-specified control.

Looking ahead, we aim to scale our system, extending the integration of human-in-the-loop interventions and improve the robot's level of autonomy to support complex, personalized, and generalizable robot learning — helping advance the field toward robots that more effectively learn to interact with humans.

2 Related Work

Shared autonomy is a paradigm where a human and a robot blend control of a system to jointly achieve a task. In this framework, the robot receives inputs from a human and combines them with autonomous functions to enable safe interaction [4]. Shared autonomy plays a significant role in enabling assistive robots to be useful to a diverse range of humans. Applications for shared autonomy include remote teleoperation interfaces [5], assistive devices for people with disabilities [6], and developing safe systems for user control [7]. While teleoperation remains important for tasks requiring nuanced or context-specific decisions, it places a high cognitive load on the user and can be repetitive and time-consuming. Prior work has explored blending teleoperation with higher levels of autonomy to reduce user effort while maintaining task performance [8]. However, recent work has shown that while increased robot autonomy can improve task performance, it can also decrease users' sense of agency, underscoring the need to balance assistance with user control [9].

End-user robot programming enables end users to specify the behavior of robots through demonstration and direct program specification [10]. Through end-user robot programming, the user can customize the robot for their context and preferences. In manipulation tasks, many end-user robot programming interfaces allow the user to specify their desired end-effector position as primitives; we follow this paradigm. Although end-user robot programming promises many advantages in enabling general behaviors aligned with the user's context and preferences, non-technical users face many challenges in using these systems [11]. This limits deployment to settings where end users require prior experience, making broader adoption at scale difficult.

Teleoperation frameworks Previous work on systems for user control of a robot have focused on teaching robots with language [12], virtual reality headsets [13], and low-cost DIY controllers [14]. Teleoperation is a critical component of collecting data from human users at scale, and prior work has shown that the manner in which the user provides demonstrations has a significant impact on resulting policy performance [15, 16]. Multiple works study how to design teleoperation systems cheaply [17] or at scale [18, 19, 20], but do not necessarily address how to obtain data for how humans intervene or correct robot behavior. There is a lack of work that approaches this challenge from the lens of understanding user preferences for control from teleoperated demonstrations, highlighting the need for systems that enable this form of data collection.

3 Shared Autonomy Toolkit

We present the Shared Autonomy Toolkit, which enables end users with limited robot programming experience to control robots to perform a range of tasks. Our repository is made open-source so as to facilitate broader usage throughout the community.

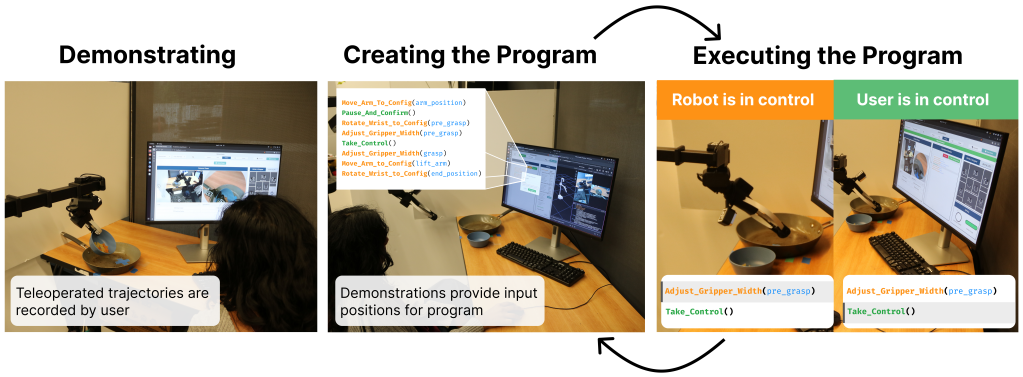


Figure 1: **Example usage of the Shared Autonomy Toolkit.** The workflow begins with the user recording a demonstration by teleoperating the robot. The recording is then used to create a program via a web-based programming interface. Finally, the program is executed on the robot, with the option to iteratively refine the code based on observed performance.

This toolkit comprises four core components, each designed to enable synchronized shared control between human operators and robots. As shown in Figure 1, users first record a demonstration by teleoperating the robot to complete a task. This can be replayed to extract robot configurations and compose a program using a library of predefined functions representing both robot and human control. The toolkit enables iterative refinement, allowing users to revise their program, save task configurations, and store reusable robot programs for future deployment.

3.1 Demonstration Recorder

The toolkit begins with demonstrations recorded via browser-based teleoperation, providing an intuitive way for users to operate a robot without requiring prior experience. This component, displayed in Figure 7, supports this step by allowing users to directly control the robot and save resulting trajectories as structured data (MCAP files). The layout provides the camera streams from the robot, button pads for controlling the robot via teleoperation, and speed controls. This step serves as the basis for extracting specific robot configurations to create a program as described in Section 3.2.

3.2 Trajectory Replayer

The trajectory replayer uses Foxglove to allow users to browse a recorded rosbag of a task demonstration [21]. A screenshot of this component is shown in Figure 8b. The user can drag the slider bar while viewing the camera feed as they would a standard video player, then selecting the point in time within the demonstration from which they want to extract information. A component of the display contains recorded topics such as the joint positions, cameras, and other data that is present at the time of recording. The trajectory replayer is meant to close the loop between demonstration and program, giving users the ability to copy and paste configurations of the robot into the program editor, where they can then save the position for certain critical yet challenging parts of the task, such as an object grasp configuration.

3.3 Program Editor

The program editor component, shown in Figure 2a, provides a way to capture generalizable and reusable robot behaviors, through enabling users to transform their demonstrations into flexible programs. They are able to select from a library of functions comprising two categories – the `Robot API` and the `Human API`. The `Robot API` contains functions that control the arm and end effector of the robot to perform user-specified manipulation tasks. The `Human API` consists of two functions that the user can include in their program to assume control over the task by confirming the robot's

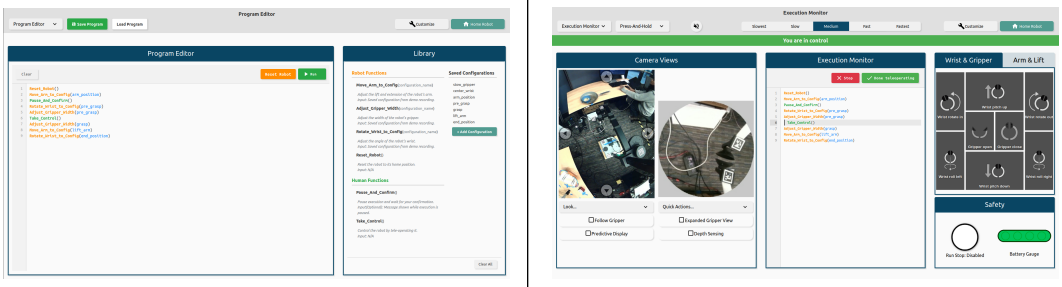
behavior or controlling the robot via teleoperation. Together, these functions allow a user to compose modular and reusable programs using the full range of the robot's degrees of freedom to accomplish a range of manipulation tasks, while satisfying their preferences for control.

- `Robot API`
  - `move_arm_to_config(config)`: Move the arm and wrist of the robot to specified joint state configuration
  - `adjust_gripper_width(config)`: Open or close the gripper to specified width configuration
  - `rotate_wrist_to_config(config)`: Move the wrist of the robot to specified orientation
  - `reset_robot()`: Return the robot to its default configuration
- `Human API`
  - `take_control()`: Stop program execution and switch to teleoperating the robot mid-program to correct its behavior if needed
  - `pause_and_confirm()`: Pause the program to inspect the robot mid-execution before resetting or proceeding with the remaining commands

Users can define and save key points captured from the recorded demonstrations to use as inputs in their programs. These saved configurations ensure task consistency and repeatability. Figure 8a displays a set of example saved configurations from the executed user programs. Programs can be repeatedly saved, re-loaded, modified, and replayed for future execution.

3.4 Execution Monitor

The execution monitor is designed to allow users to pair observations of the robot's behavior at execution time alongside a line-by-line view of what function is being performed by the robot. It enables the user to gain insight into their program behavior in order to understand what they may want to modify, if at all. Additionally, it contains utilities to allow for human control when the user specifies they want to intervene during program execution. For example, when `take_control()` is called, execution pauses and the human user is given control of the robot using the teleoperation interface as shown in 2b. Once finished, the user can return control to the robot to resume executing the remaining program.



(a) Program Editor (b) Execution Monitor

Figure 2: **Program Editor and Execution Monitor interface.** The user composes their program with the functions in the library and by adding configurations from their demonstration to use as inputs. The program can be executed and iteratively modified until the user is satisfied.

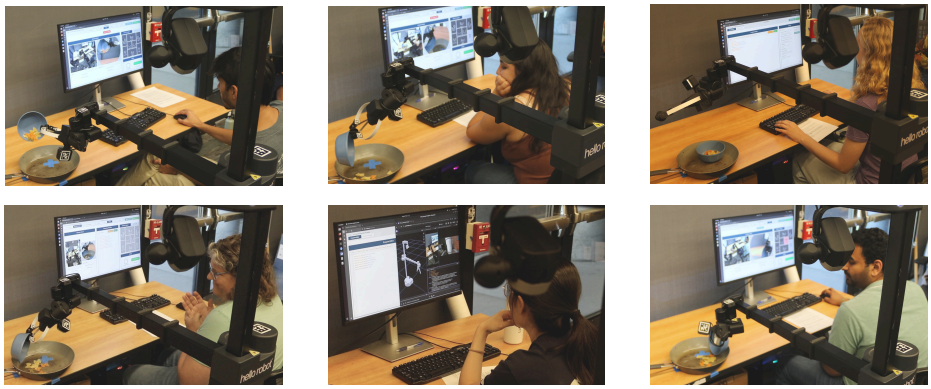


Figure 3: **Real world deployment with humans.** Multiple users of different skill levels operating the robot to perform the task of pouring pasta into a pan using the Shared Autonomy Toolkit.

4 Experiments

4.1 User Study Procedure

We ran a within-subjects user study with $N = 8$ total participants comprising 4 novice and 4 experienced users of robotic systems. Participants were recruited by word-of-mouth and advertising through public university channels. Participants were primarily aged 18–24 ($N = 5$, 62.5%), with 2 participants aged 25–34 (25%) and 1 participant aged 45–54 (12.5%). The sample included 4 women, 3 men, and 1 non-binary participant. On a 5-point scale, all participants reported high prior programming experience ($5/5$). Self-reported prior experience with robotic systems averaged 57.5% overall. The study was conducted following IRB protocol in 1-hour long sessions. All participants were compensated for their time.

Each participant was introduced to the system by completing a practice session before being instructed to complete two rounds of a task involving teleoperating, programming, and executing code to control a robot. In one task round, participants were instructed to use a combination of `Robot + Human API` functions, and in another round, were told to complete the task with only the `Robot API` available. The order of each round was randomized to minimize ordering effects. After each task round, participants completed a questionnaire to report their experience interacting with the different system configurations. At the conclusion of the study, participants were asked to evaluate their experience with a final set of questions described in Section A.3.

4.2 Experimental Setup

To evaluate our system, the participants controlled the Stretch robot to perform the task of picking up a bowl filled with pasta and transferring its contents into a saucepan. We chose this task due to its complex manipulation requirements, including multiple failure modes where human correction may be necessary, such as incorrectly specifying the grasp position of the bowl and using the wrong orientation to pour the pasta without missing the target. Additionally, it requires the use of all degrees of freedom of the robot's arm, wrist, and gripper in order to successfully achieve the goals of the task.

Independent Variables We compare the use of a system with human-in-the-loop shared autonomy versus one without by changing the availability of robot control methods in each of the two rounds of the study. In one round, the users are given a greater degree of control over the system by being provided with a library of functions spanning robot control and human intervention. We compare this against a more limited version of the system which only includes the ability to specify robot control, without any human input during execution.

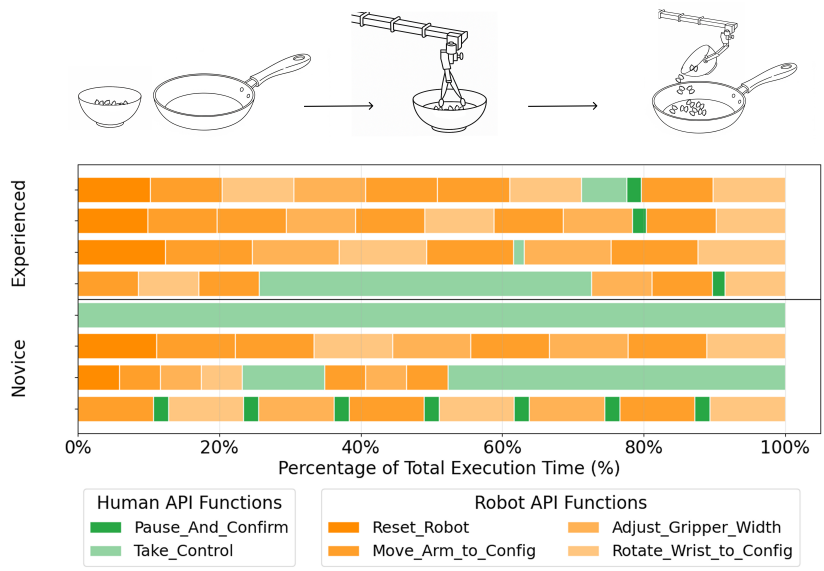


Figure 4: **User programs for pouring pasta task.** Comparison of the programs participants created with the `Robot + Human API` in the Shared Autonomy Toolkit. Novices tended to use the `Human API` functions (green) a proportionally higher percentage of time compared to experienced users.

Dependent Variables We include quantitative metrics of task completion time, total number of program iterations, and the distribution of robot functions versus human control at execution time. In addition, we collected survey-based quantitative measures capturing perceived workload, system evaluations, perceived control, and preference for the Shared Autonomy Toolkit relative to others.

We also gathered qualitative feedback via free-response prompts on when and why participants modified their programs, which functions they preferred, and for what other types of tasks they would use this system.

4.3 Hypotheses

- **H1:** The Shared Autonomy Toolkit enables higher task success compared to `Robot API`-only execution.
- **H2:** The inclusion of human-in-the-loop control causes lower user effort for task success.
- **H3:** Users prefer to use a system that enables human control during task execution.

5 Experimental Results

5.1 Quantitative Evaluation

We found that the Shared Autonomy Toolkit enables users to achieve a higher degree of task success than when only using the `Robot API`, shown in Figure 6, supporting **H1**. The data does not support **H2**; participants overall took longer to complete programs and required a higher number of iterations when using the combination of `Robot + Human API` versus robot-only programs. This trend, displayed in Figure 5, is more pronounced for novice users, whereas there is little change in program completion time for experienced participants. We speculate that this result is due to novice users requiring more time to decide which functions to use when presented with more options.

Additionally, novice participants were more likely to use the `Human API` than experienced users. All but one of the participants used at least one `Human API` function in their program. The data in Figure

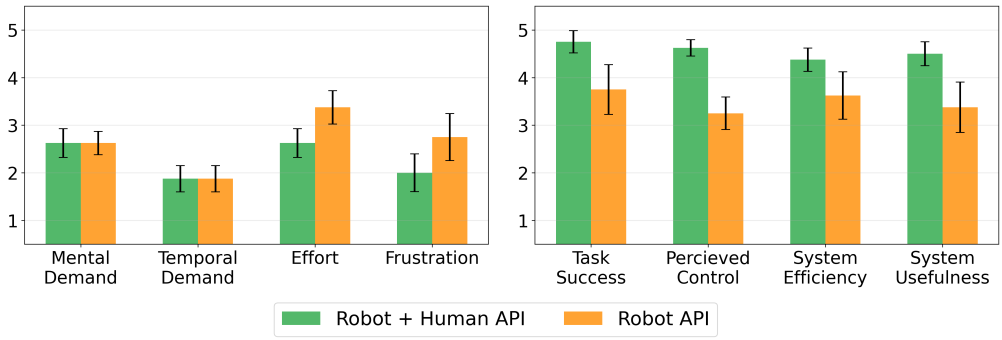


Figure 6: **User study subjective results.** We report responses to subjective questions evaluating their experience interacting with both system configurations of `Robot + Human API` and `Robot API` according to negative sentiment (left) and positive metrics (right).

6 supports **H3**, indicating that across a range of metrics including perceived control (+42.3%) and usefulness (+33.3%), the Shared Autonomy Toolkit scored higher than `Robot API` usage alone.

Experienced users tended to employ the `Human API` function sparingly for correcting and validating behavior, while novices showed more varied use. The most common uses were pre-grasp corrections, teleoperation for grasp, post-grasp confirmation, and end of task corrective actions such as wrist rotations to ensure task success. One participant used `take_control()` for the entire duration of the task and one controlled the robot without any usage of the `Human API`.

5.2 Subjective Metrics

We measured users' subjective responses to each workflow (with human-in-the-loop and without) across metrics described in Section A.3. The results indicate that enabling user control through the `Human API` yields lower perceived user effort and frustration and increased user performance. With the `Human API` available, 7 out of 8 participants rated perceived success at $5/5$; ratings had higher variance when only the `Robot API` was available. The full results for these findings are reported in Figure 6.

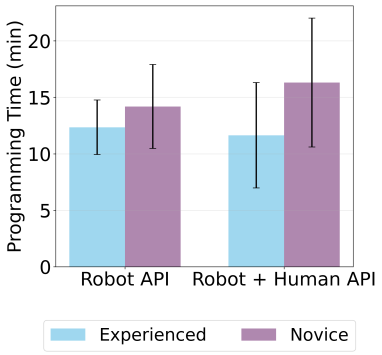


Figure 5: **Quantitative results.** Participants' total time taken to successfully compose a program.

Participants rated the overall system highly on usability: ease of use ($M = 4.31/5$) and intuitiveness ($M = 4.31/5$). Participants reported that they would prefer to use the Shared Autonomy Toolkit over a fully autonomous system ($M = 4.31/5$) and over a fully manual system ($M = 4.00/5$).

Furthermore, we obtained participants' subjective impressions through quotes from their experience during the study. After interacting with the system and completing the task under both conditions, one user noted "I don't think I could have programmed the robot perfectly without using `take_control()` even if I sat here for another hour." Another user shared that "I felt that my task success could go from 70% to 100% through use of the `Human API`".

6 Conclusion

This work establishes a foundation for better understanding human preferences in shared autonomy to enable more effective robot learning from human data. The results indicate that users exhibit distinct patterns in **when** and **how** they choose to intervene during shared autonomy, with preferences varying based on user experience level and task context. By creating human-in-the-loop systems for users to customize how to execute tasks on a robot, we move a step closer toward a more nuanced

understanding of how to adapt robots to human environments, including where to incorporate humans in the loop for robot learning. By capturing user preferences for control, this work paves the way towards developing robots that can adapt to individual human behaviors and needs.

Our findings also suggest that the use of the Shared Autonomy Toolkit enables people with a range of skill levels to operate robots in multi-step manipulation tasks, a critical component of collecting high-quality human data at scale. Furthermore, every component of our system is open source, providing the potential for more widespread adoption. We hope this system provides a useful set of tools for researchers to gain better insights about enabling robots to adapt to humans while also facilitating the broader usage of robots by everyday people.

7 Future Work

Our aim for the Shared Autonomy Toolkit is to enable everyday users to operate a robot to perform a range of household tasks with human-in-the-loop control. We showed that the system enables even novice users to compose, save, and replay reusable programs from a single demonstration. Towards this goal, we will test the system on a broader range of long-horizon tasks requiring complex manipulation, such as placing laundry in a basket and retrieving items from a drawer. By creating an intuitive, open source system, we can then collect targeted demos for a range of tasks that contain insights on where humans want to be in the loop during shared control; our future work aims to address this by adapting robot policies to learn from in the loop interventions.

Furthermore, our overarching goal is capture and understand nuanced patterns of shared autonomy; one shortcoming of our findings for the task studied in this work is that users primarily tend to use `take_control` when the robot is prone to failure. We hope that evaluating our system on a broader range of tasks and contexts will enable users to focus their interventions on preference-driven corrections rather than compensating for system limitations. Doing so will also pave the way for robots to learn how to incorporate humans into the loop in shared autonomy, such as in collaborative tasks where a robot must learn to adapt from human corrections.

References

[1] S. Alexandrova, M. Cakmak, K. Hsiao, and L. Takayama. Robot programming by demonstration with interactive action visualizations. 07 2014. doi:10.15607/RSS.2014.X.048.

[2] S. Mirchandani, S. Belkhale, J. Hejna, E. Choi, M. S. Islam, and D. Sadigh. So you think you can scale up autonomous robot data collection?, 2024. URL https://arxiv.org/abs/2411.01813.

[3] J. Jang, S. Ye, Z. Lin, J. Xiang, J. Bjorck, Y. Fang, F. Hu, S. Huang, K. Kundalia, Y.-C. Lin, L. Magne, A. Mandlekar, A. Narayan, Y. L. Tan, G. Wang, J. Wang, Q. Wang, Y. Xu, X. Zeng, K. Zheng, R. Zheng, M.-Y. Liu, L. Zettlemoyer, D. Fox, J. Kautz, S. Reed, Y. Zhu, and L. Fan. Dreamgen: Unlocking generalization in robot learning through video world models, 2025. URL https://arxiv.org/abs/2505.12705.

[4] H. J. Jeon, D. P. Losey, and D. Sadigh. Shared autonomy with learned latent actions. *CoRR*, abs/2005.03210, 2020. URL https://arxiv.org/abs/2005.03210.

[5] C. Mower, J. Moura, and S. Vijayakumar. Skill-based shared control. 07 2021. doi:10.15607/RSS.2021.XVII.028.

[6] S. Udupa and C. Menassa. Shared autonomy in assistive mobile robots: a review. *Disability and Rehabilitation: Assistive Technology*, 18:1–22, 06 2021. doi:10.1080/17483107.2021.1928778.

[7] M. Li, X. Song, H. Cao, J. Wang, Y. Huang, C. Hu, and H. Wang. Shared control with a novel dynamic authority allocation strategy based on game theory and driving safety field. *Mechanical Systems and Signal Processing*, 124, 01 2019. doi:10.1016/j.ymssp.2019.01.040.

[8] S. Javdani, H. Admoni, S. Pellegrinelli, S. S. Srinivasa, and J. A. Bagnell. Shared autonomy via hindsight optimization for teleoperation and teaming, 2017. URL https://arxiv.org/abs/1706.00155.

[9] C. Yang, H. Patel, M. Kleiman-Weiner, and M. Cakmak. Preserving sense of agency: User preferences for robot autonomy and user control across household tasks, 2025. URL https://arxiv.org/abs/2506.19202.

[10] G. Ajaykumar, M. Steele, and C.-M. Huang. A survey on end-user robot programming. *ACM Comput. Surv.*, 54(8), Oct. 2021. ISSN 0360-0300. doi:10.1145/3466819. URL https://doi.org/10.1145/3466819.

[11] M. Swaminathan, L.-J. Hsu, M. M. Thant, K. J. Amon, A. S. Kim, K. M. Tsui, S. Sabanović, D. J. Crandall, and W. Khoo. If [yourname] can code, so can you! end-user robot programming for non-experts. In *Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '24, page 1033–1037, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703232. doi:10.1145/3610978.3640644. URL https://doi.org/10.1145/3610978.3640644.

[12] J. Grannen, S. Karamcheti, S. Mirchandani, P. Liang, and D. Sadigh. Vocal sandbox: Continual learning and adaptation for situated human-robot collaboration, 2024. URL https://arxiv.org/abs/2411.02599.

[13] E. Rosen and D. K. Jha. A virtual reality teleoperation interface for industrial robot manipulators, 2023. URL https://arxiv.org/abs/2305.10960.

[14] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators, 2024. URL https://arxiv.org/abs/2309.13037.

[15] H. Li, Y. Cui, and D. Sadigh. How to train your robots? the impact of demonstration modality on imitation learning, 2025. URL https://arxiv.org/abs/2503.07017.

[16] M. A. Rana, D. Chen, S. R. Ahmadzadeh, J. Williams, V. Chu, and S. Chernova. Benchmark for skill learning from demonstration: Impact of user experience, task complexity, and start configuration on performance, 2019. URL https://arxiv.org/abs/1911.02725.

[17] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets, 2021. URL https://arxiv.org/abs/2109.13396.

[18] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[19] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, V. Guizilini, D. A. Herrera, M. Heo, K. Hsu, J. Hu, M. Z. Irshad, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset, 2025. URL https://arxiv.org/abs/2403.12945.

[20] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, S. Savarese, and L. Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation, 2018. URL https://arxiv.org/abs/1811.02790.

[21] Foxglove: Visualization and management for temporal and multimodal robotics data. https://foxglove.dev/product. Accessed: August 15, 2025.

[22] V. Ranganeni, V. Dhat, N. Ponto, and M. Cakmak. Accessteleopkit: A toolkit for creating accessible web-based interfaces for tele-operating an assistive robot. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, UIST '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706288. doi: 10.1145/3654777.3676355. URL https://doi.org/10.1145/3654777.3676355.

[23] S. G. Hart and L. E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Human mental workload*, 1(3):139–183, 1988.

[24] R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 140:1–55, 1932.

A Appendix

A.1 Additional System Component Details

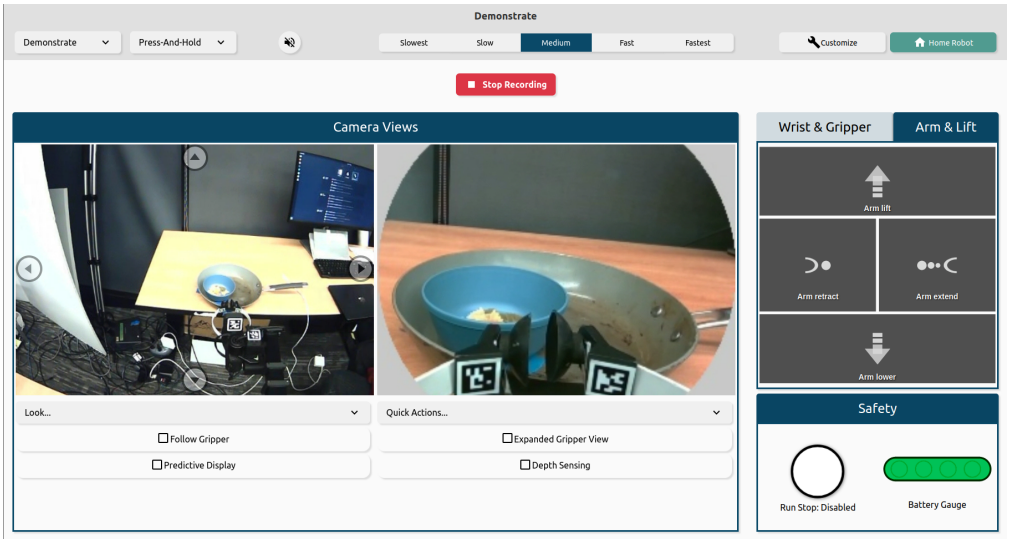


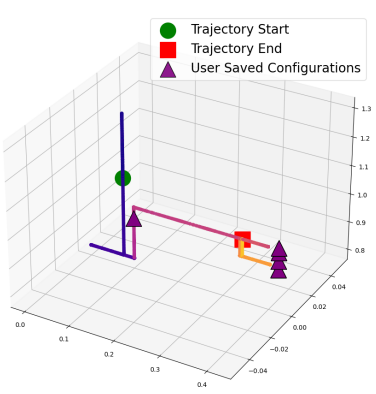
Figure 7: **Demonstration recorder interface.** The user teleoperates the robot through the controls on the right panel. The demonstration can be recorded by pressing the center button (in red).

Demonstration recorder The demonstration recorder extends the teleoperation library from Hello Robot, a customizable web-based interface designed to accommodate novice end users [22]. The demonstration recorder interface is shown in Figure 7. The system utilizes ROS2, WebRTC, NodeJS, and TypeScript. The system runs a headless browser onboard the robot, which communicates with ROS2 via rosbridge—a protocol converting JSON messages to ROS2 commands and vice versa. When the system interface launches, the robot browser opens a WebSocket room, awaiting a connection from an operator. The user can join via IP address on local networks or a URL to join remotely. Only one peer connection is supported at a time to prevent command conflicts. Once a connection is established, the system interface loads, providing direct but secure control. Commands, such as moving the robot forward, are sent from the browser, translated through rosbridge, and executed by the robot. The robot browser can also relay information back to the operator. When the session ends, the peer connection closes, and another operator may access the toolkit.

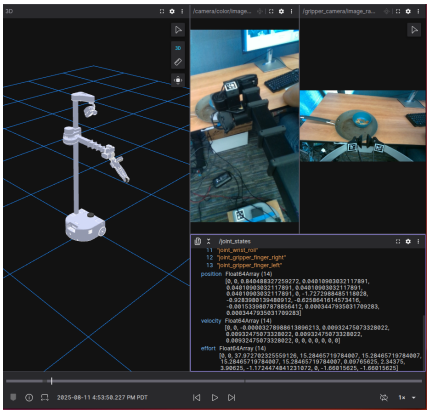
Figure 8a illustrates example saved configurations from a user program with respect to a robot trajectory. The saved configurations are extracted from the Foxglove tool in 8b.

A.2 Robot Platform

The Hello Robot Stretch 3 is a low-cost mobile manipulator. Stretch has a telescoping arm that extends 50cm horizontally and is attached to a prismatic lift that reaches 110cm vertically. The arm has a 3 degree of freedom dexterous wrist and 1 degree of freedom gripper. The movement of the arm is orthogonal to the movement of the diferential drive base. Stretch has three cameras: a realsense D435if camera and wide-angle camera attached to a pan-tilt head, and a realsense D405 camera mounted on the gripper. Stretch's lightweight (24.5kg) and safe design, and physical capabilities make it feasible for long term deployment in end-users' homes. Stretch has open source software in Python and ROS2.



(a) User saved configurations



(b) Foxglove interface

Figure 8: Visualization of user saved configurations overlaid on top of robot trajectory. The user extracts saved configurations in Foxglove from the bottom right panel, using the slider at the bottom to select where in the recorded demonstration they want to pause.

A.3 Additional User Study Details

Our participant questionnaire for each task round was adapted from NASA-TLX [23] (excluding Physical Demand), rating Mental Demand, Temporal Demand, Effort, Frustration, and Performance (operationalized as perceived task success). Additional per-condition items (on a 5-point Likert scale) included: "I felt I had control over the robot in accomplishing this task," "The system helped me complete the task efficiently," and "The system was useful in helping me accomplish the task." We also collected two free-response prompts per condition: "When and why did you modify your program?" and "What parts of this task do you prefer to do yourself instead of having the robot execute it autonomously?"

At the conclusion of the session, participants were asked to evaluate their overall experience by rating their agreement with the following statements on a 5-point Likert scale [24]:

- This system was easy to use.
- This system was intuitive to use.
- I prefer using this shared autonomy system compared to one that is fully autonomous.
- I prefer using this shared autonomy system compared to one that is fully manual control.
- I felt confident that my program would produce the behavior that I intended.