ARC-LfD: Using Augmented Reality for Interactive Long-Term Robot Skill Maintenance via Constrained Learning from Demonstration

Matthew B. Luebbers[1], Connor Brooks[1], Carl L. Mueller[1], Daniel Szafir[1,2], Bradley Hayes[1]

***Abstract*— Learning from Demonstration (LfD) enables novice users to teach robots new skills. However, many LfD methods do not facilitate skill maintenance and adaptation. Changes in task requirements or in the environment often reveal the lack of resiliency and adaptability in the skill model. To overcome these limitations, we introduce ARC-LfD: an Augmented Reality (AR) interface for constrained Learning from Demonstration that allows users to maintain, update, and adapt learned skills. This is accomplished through in-situ visualizations of learned skills and constraint-based editing of existing skills without requiring further demonstration. We describe the existing algorithmic basis for this system as well as our Augmented Reality interface and the novel capabilities it provides. Finally, we provide three case studies that demonstrate how ARC-LfD enables users to adapt to changes in the environment or task which require a skill to be altered after initial teaching has taken place.**

I. INTRODUCTION

Robot Learning from Demonstration (LfD) methods enable users to teach desired skills to robots without programming or other forms of robot-specific knowledge [1], [2]. The predominant focus of LfD research to date has been on the initial learning process itself, rather than the maintenance and adaptation of learned models. In a shift of focus to the latter, we introduce *Augmented Reality for Constrained Learning from Demonstration* (ARC-LfD): a system that combines an Augmented Reality (AR) interface and constrained Learning from Demonstration [3] to enable users to teach a robot new skills as well as verify, repair, and edit existing skills. ARC-LfD demonstrates a novel approach to LfD that can mitigate problems arising from poor quality demonstrations, changes in the environment, and adaptations to the task procedure.

When using LfD methods for robot instruction, safe deployment necessitates verification that a skill has been learned properly after the skill has been demonstrated and taught. While verification can be done in simulation, this requires a high-fidelity model of the environment in order for the visualization of the learned skill to be shown in the proper context (and obtaining such a model may be a technical endeavor). After this step is completed, the robot may begin executing the learned skill as long as the environment stays constant, but even small changes in the robot's environment or the desired skill may require an entirely new set of demonstrations to fix it. This requirement for rigidity of environment and task can make long-term deployment and maintenance of the skill difficult in practice.

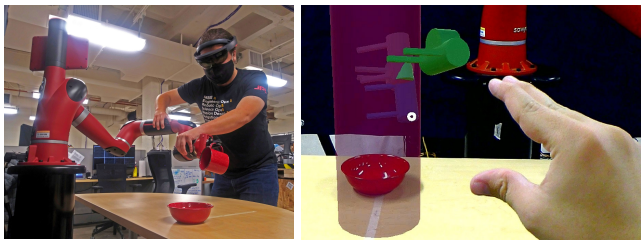

Fig. 1. ARC-LfD combines Augmented Reality with constrained Learning from Demonstration to create a system that enables the teaching, verification, editing, and updating of robot skills using in-situ visualizations.

One approach to handling this rigidity is the creation of end-to-end policy learning systems that aim to model skills more generally. However, such systems may demand a prohibitive number of demonstrations or require unavailable simulation environments to capture user intent, and aren't designed to accommodate user selection of task constraints. Our approach emphasizes transparency and adaptability in a system designed for online skill editing and validation necessary for long-term robot deployment. Through AR visualization, ARC-LfD safely demonstrates to users what skill has been learned and how executing that skill will cause the robot to move through the environment. The AR interface also facilitates the visualization and editing of constraints, enabling users to see how these constraints interact with objects or points of interest in the environment. Furthermore, constraint editing through AR allows the entire training process to take place in-situ without requiring context-switching between the real environment and a 2D display.

The contributions of the ARC-LfD system are as follows:

1) AR visualizations of learned skills, in-situ robot behavior, and constraints without needing a model of the entire environment.
2) An iterative process to verify, repair, and edit existing skills through AR using visualized constraints employed by the underlying LfD algorithm.
3) Three case studies that illustrate how the system enables skill adaptation with no further demonstration.

II. RELATED WORKS

A. Learning from Demonstration

Robot Learning from Demonstration (LfD) encompasses a set of methods that strive to learn successful robot behavior

Corresponding author: matthew.luebbers@colorado.edu

[1] Computer Science Department, College of Engineering and Applied Science, University of Colorado Boulder, Boulder, Colorado, USA

[2] ATLAS Institute, University of Colorado Boulder, Boulder, Colorado, USA

This work was funded in part by NSF Award #1830686 and the U.S. Army Research Lab STRONG program.

models from human input [2]. A human operator interacts with a robotic system through some mode of demonstration, usually through kinesthetic demonstration (e.g., physical interaction), teleoperation (e.g., remote control), or passive observation (e.g., motion tracking observation). While the mode may vary, demonstrations ideally communicate the nature of the skill to the robot such that the learned model effectively resembles some latent ground truth model held by the demonstrator [1]. The methods by which robotic systems learn such models span a broad spectrum, but are generally categorized into three classes: 1) plan learning, 2) functional optimization, and 3) policy learning [4]. Most importantly, LfD methods enable non-roboticists to quickly teach robots useful skills and forgo the need for expert robotics programming knowledge.

ARC-LfD uses an LfD method that falls under the policy learning categorization, where the goal is to learn models that output either robot trajectories or low-level actions directly. Work by Akgun et al. [5] introduces Keyframe-based LfD, a method that learns a sequential waypoint (i.e. keyframe) model of a skill through the clustering of demonstrated trajectories. Keyframe models essentially produce coarse trajectories for the robot to execute by employing motion planning algorithms to traverse from waypoint to waypoint.

ARC-LfD utilizes an enhanced variant of this technique called *Concept Constrained Learning from Demonstration* (CC-LfD) [3]. During demonstration, users annotate behavioral constraints (through real-time dictation) to be applied to the learned model. Akin to prior work in learning from human teachers [6], [7], this algorithm is motivated by the insight that although traditional state data captured by the robotic learner does encode certain aspects of the task, the users' internal model might have latent information not communicated through traditional kinesthetic demonstration. Thus, by enabling the user to also communicate behavioral constraints (e.g., "a cup must remain upright until over the bowl"), the robotic learning system is given additional information that helps produce a more robust and successful model. To this end, CC-LfD requires far fewer demonstrations to teach a successful skill model than robot state demonstration trajectories alone produce, and enables post-hoc skill repair and adaptation through constraint application.

Keyframe-based LfD methods are agnostic to the mode of demonstration as they operate on the resulting trajectories. However, ARC-LfD utilizes kinesthetic demonstration, where users physically manipulate the robotic system to produce demonstration trajectories. Akgun et al. [8] showed that kinesthetic demonstration generally produces more successful skill models and is the preferred mode of demonstration by end-users when compared with teleoperation. However, Wrede et al. [9] described how kinesthetic demonstration can be limited by non-experts users' lack of robotics knowledge. For example, they showed that resultant models learned through kinesthetic demonstration perform poorly when users guide robots close to configuration space Jacobian singularities. Furthermore, Villani et al. [10] surveyed a multitude of industrial environments in which robots are deployed, describing highly variable and potentially dangerous collaborative environments and tasks. Such environments challenge kinesthetic demonstration as complex structures and dangerous conditions make kinesthetic demonstration infeasible to model or unsafe for humans.

Given these concerns, safety and adaptability become paramount, both for the design of safe human-robot collaborative environments [11] and for the mechanisms by which robots build skill models [12], [3]. The ARC-LfD system utilizes an AR interface that enables users to both visualize learned skills and to define a strict set of behavioral restrictions via the application and editing of constraints. The benefit is twofold: 1) constraint application helps facilitate encoding additional information, shifting the burden of end-user expertise away from robotics and towards the task consideration, and 2) AR enables a user to operate in an environment where certain features (dangerous objects, difficult arrangement, etc.) that make kinesthetic demonstration either infeasible or dangerous can be virtualized, communicating skill-essential behavioral restrictions as encoded constraints.

B. Augmented Reality Interfaces for Robotics

In order to facilitate an additional visual interface for an LfD system without requiring user context-switching [13], we use AR. AR interfaces for robotics have a proven track record [14], [15], enabling new methods of enhancing robotic control [16], [17], [18], [19], collaboration in human-robot teaming [20], [21], safe movement in shared spaces [22], [23], and communication of robot knowledge [24], [25], [26]. Motivated by this existing body of work, we use AR to create an interface for LfD that previews learned skills and allows editing of constraints directly in the robot's environment.

Through ARC-LfD, users are able to examine a sample trajectory from a learned skill visualized in AR through an overlay in the workspace environment. Such skill visualization is intended to improve safety as the operator can "preview" robot behavior without the need for actual skill execution [27]. Prior work has established this potential through user studies: Walker et al. [22] conducted a user study which found that showing flying robot paths in AR made users more efficient and comfortable when sharing an environment with these robots. Similarly, Rosen et al. [23] found that AR visualization of possible robotic arm trajectories improved participants' accuracy and quickness in identifying collisions with objects in the environment. These studies substantiate the notion that AR visualizations of robot trajectories may improve user understanding with respect to the path a robot will take and how that trajectory will interact with the environment.

In addition to visualizing the robot's possible future movement, ARC-LfD supplies visual cues that describe the robot's ability to adhere to user supplied behavioral constraints on a learned skill. This is akin to helping users understand the internal state of the robot, another functionality that has been explored within the space of AR for human-robot interaction. Through AR, information such as the robot's battery life [25] or sensor readings [24] can be communicated to users

through a heads-up display. This is particularly useful when performing complex tasks such as controlling a robot as it prevents disruptive context-switching when averting attention away from the environment towards a 2D display [13]. Using AR to visualize a robot's knowledge in the form of a learned skill or action can also provide a realistic demonstration of this knowledge without requiring extensive modeling of the environment to use in simulation [26].

The final type of interaction supported by AR in ARC-LfD is the ability to create and manipulate constraints on a learned skill. Visualizing constraints in the physical environment allows users to see the exact effect of applying these constraints [28]. Yamamoto et al. [16] illustrated that applying virtual constraints was an effective tool for robot-assisted surgery, allowing surgeons to specify thresholds that the robot should not cross. In our case, the constraints are both shown and edited in the environment in which the skill will be executed, allowing users to move constraints around physical objects to ensure the skill can be performed safely.

Generally, we are motivated in designing ARC-LfD by a rich history of research into LfD as well as strong results from prior work at the intersection of AR and robotics that demonstrate AR interfaces outperform 2D and tablet-based interfaces for visualizing information critical to human-robot interactions. In the next two sections, we describe the algorithmic basis for ARC-LfD followed by the design and capability features of the AR interface.

III. CONCEPT CONSTRAINED LEARNING FROM DEMONSTRATION

As shown in Figure 2, ARC-LfD consists of two components communicating via the Robot Operating System (ROS): a Concept Constrained Learning from Demonstration subsystem (CC-LfD), which serves as a backend for skill learning, and an AR subsystem for visualization and user interactions with a learned skill. In this section, we present an overview of CC-LfD; however, we point the reader to the original paper for a more thorough review: [3].

CC-LfD is an augmentation of Keyframe-based Learning from Demonstration [5] that incorporates the ability to utilize constraints, consisting of concepts (e.g., "X is above Y", "Z is powered on", etc.) encoded as Boolean planning predicate classifiers, to produce a more representative learned model of the demonstrated skill. The motivation behind incorporating predicate-based constraints is to overcome the limited capacity of demonstrated robot state (e.g., end-effector state) trajectories alone to encode all critical aspects of a skill that a human operator intends the robot to learn. For example, when teaching a robot a cup carrying task, robot state data alone will not adequately capture the concept of "keeping a cup upright." By leveraging logical combinations of predicate-based constraints, CC-LfD biases waypoint sampling from learned keyframes, resulting in a dramatic reduction in the required number of demonstrations to both train a successful model and repair a poorly performing skill as compared to introducing additional high-quality demonstrations.

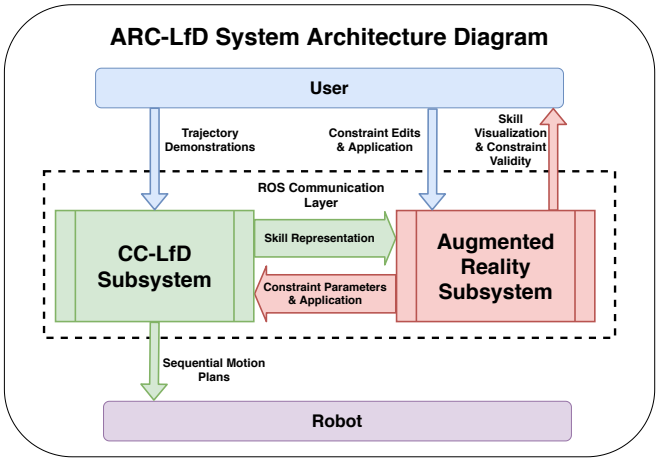

Fig. 2. A diagram of the ARC-LfD system architecture. The user (blue) supplies the initial demonstrations to the CC-LfD subsystem (green). During the editing phase, the user also supplies constraint edits and their keyframe application to the AR subsystem (red). In return, the AR subsystem supplies skill and keyframe constraint validity visualizations to the user. Through a Robot Operating System (ROS) communication layer, the CC-LfD and AR subsystems exchange skill representation, constraint parameterization, and constraint application information. Finally, the CC-LfD subsystem provides sequential motion plans for the robot (purple) to execute.

The CC-LfD algorithm requires a set of demonstrated robot trajectories annotated with constraints. These trajectories are aligned via Dynamic Time Warping [29] to preserve point-to-point spatio-temporal similarity across trajectories. Once the trajectories are aligned, annotated constraints are combined via a Boolean logical *AND* across all demonstrations. Sequential clusters of aligned trajectory points provide the basis for the nodes of a directed acyclic graph representative of a learned skill.

Individual keyframe models are created by fitting distributions on the data within each cluster. Keyframes inherit the set of constraint annotations preserved during the alignment step. Importantly, constraint set change-point regions demarcate special keyframes of data known as *boundary keyframes*. Consecutive keyframes whose variational distance is below a threshold parameter are culled from the keyframe graph. This produces a more sparse keyframe representation and eliminates backtracking behavior during skill execution. Boundary keyframes are never deleted as they represent pertinent structural change-points for the learned skill. To better ensure each keyframe is representative of a constraint-compliant distribution, a rejection sampling step produces a constraint-compliant set of points that is used to rebuild the keyframe distributions. Finally, skill execution is accomplished by sequentially sampling constraint-compliant waypoints from a directed path through the keyframe graph, subsequently constructing motion plans between waypoints.

ARC-LfD introduces an advancement over CC-LfD by enabling post-hoc application of constraints as opposed to requiring constraint application during demonstration. This new approach facilitates an iterative update process that alters keyframe constraints and the corresponding distributions, providing the basis for ARC-LfD to achieve skill adaptation.

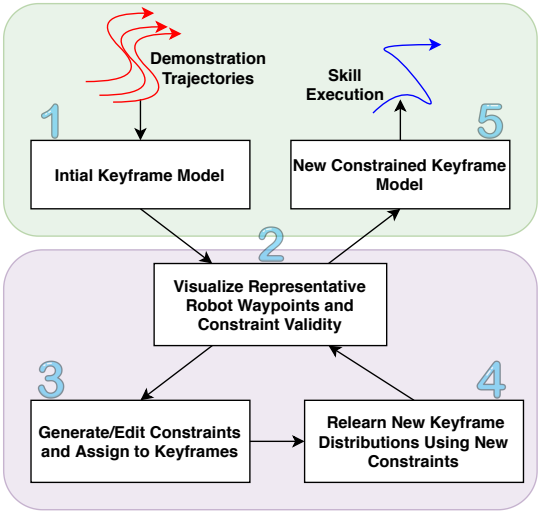

Fig. 3. Flowchart indicating how ARC-LfD integrates into CC-LfD. Steps 2, 3, and 4 repeat until the user is satisfied. The pink region (bottom) indicates AR-based steps whereas the green region (top) indicates that AR is not strictly required.

ARC-LfD first generates an initial keyframe model of the skill (Fig. 3, Step 1), which is visualized as an instantiation of the keyframe waypoints that the robot will execute (Fig. 4). This visualization includes the validity of each waypoint relative to the keyframe's applied constraints (Fig. 3, Step 2). Using the AR interface, the user generates new constraints, or edits existing constraints (Fig. 3, Step 3), and assigns them to a chosen keyframe. This initiates a model rebuilding phase where keyframe distributions are relearned using the same rejection sampling and distribution fitting steps as CC-LfD (Fig. 3, Step 4). If the user is satisfied with the visualized robot behavior, skill execution can proceed as carried out by the CC-LfD algorithm (Fig. 3, Step 5).

IV. AUGMENTED REALITY SYSTEM DESIGN

The second subsystem of ARC-LfD (see Fig. 2) is an AR interface deployed on a HoloLens, a mixed reality headset developed by Microsoft. A headset was chosen over alternative tablet-based passthrough AR solutions due to its hands-free nature, freeing users' hands for interaction with the robot, and its ability to show different imagery to different eyes, enabling superior depth perception [30]. Users wearing the HoloLens are able to see holographic visualizations of relevant keyframes and constraints projected onto the robot's workspace. User interaction is achieved through performing pinching gestures known as *air taps* on these visualizations and on menu buttons pinned above the robot (see Fig. 1).

A. Skill & Constraint Representation

For a given skill, each keyframe generated by CC-LfD is sent to the AR interface and visualized as a hologram of the robot's end-effector, whose position and rotation are representative of a randomly sampled valid waypoint within that keyframe. The combination of these keyframe visualizations traces out a trajectory that the robot would follow to execute the skill. To aid the user in evaluating a candidate trajectory at a glance, the end-effector holograms are colored in a gradient from green to gray to indicate the ordering of the keyframes, and any waypoints in violation of an applied constraint are colored bright red (see Fig. 4).

TABLE I
EDITABLE CONSTRAINTS AND ADJUSTABLE PARAMETERS IN ARC-LFD

Editable Constraints			
Constraint Type	**AR Visualization**	**Parameters**	**Example**
Height Above/Below	Plane w/ Arrows	Reference Height, Direction	Fig. 4, top-right
Orientation	Orientation Validity Cone and Fan	Orientation, Affordance Angle	Fig. 4, bottom-left
Over-Under	Cylinder	Position, Validity Radius	Fig. 4, bottom-right

Our test implementation incorporates three constraint types, representing a subset of possible parametric, predicate-based constraint templates for ARC-LfD, selected to provide coverage over a number of common robotic manipulation task setups. These are height constraints (the robot's end-effector must stay above or below a given height), orientation constraints (the robot's end-effector must maintain a given rotation, within a given affordance), and over-under constraints (the robot's end-effector must stay above a given location, within a given radius). Each constraint type has its own associated visualization: a plane with arrows indicating the valid direction for height constraints, a cone and fan overlaid onto an end-effector showing the affordance for each axis for orientation constraints, and a cylinder representing the radius around a target for over-under constraints. When the user selects a keyframe with a constraint applied, that

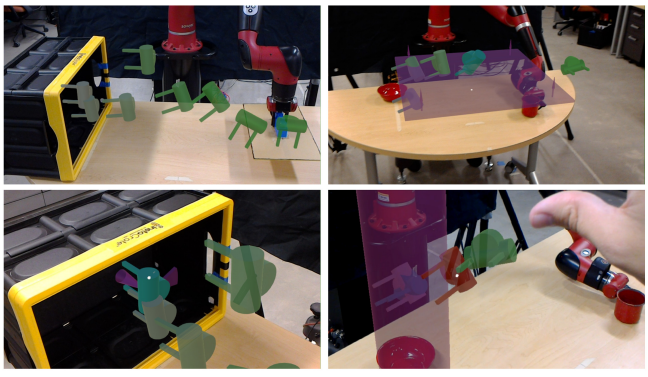

Fig. 4. ARC-LfD allows the user to visualize trajectories as a series of keyframes (top left). Selecting a keyframe will show holograms representing any constraints active at that keyframe, such as the height constraint (top right) indicating the end-effector must stay above the plane, the orientation constraint (bottom left) overlaid on the selected end-effector to show its proper rotation, and the over-under constraint (bottom right) indicating the end-effector must stay within the cylinder. Note that in the bottom right image, one keyframe has the over-under constraint applied, but is not located inside the cylinder, placing it in violation of the constraint, and coloring its hologram red to alert the user.

constraint hologram appears, positioned, rotated, and scaled according to its parameters, and colored a translucent purple to maximize visibility of the trajectory and environment. For a summary of these constraints, their AR visualizations, their editable parameters, and references to examples, see Table I.

B. Constraint Editing & Application

ARC-LfD lets users edit existing constraints and create new ones from a template via the AR interface (see Fig. 5). The user accesses the constraint editing interface by selecting a constraint type and slot with the menu buttons above the robot. The user will then have the trajectory visualization cleared from their view and a lone constraint visualization will be rendered. The user can edit the parameters of their chosen constraint type (see Fig. 5), seeing the visualization update in real-time, which allows them to match constraints to environmental features (e.g., placing an over-under constraint on top of a target object for a pick-and-place task).

Once a user is satisfied with their new constraint, they press a confirmation button, which synchronizes the representation across the AR and CC-LfD subsystems of ARC-LfD. They are then able to apply that constraint to a keyframe or range of keyframes through the constraint application menu until they have added the constraint to the desired areas of the skill trajectory. Once this process is complete, and the trajectory has been satisfactorily inspected, the user selects the "Send to Robot" button to send the new constraint application to the CC-LfD subsystem, which initiates a rebuilding and resampling of the skill. After the CC-LfD subsystem has relearned a set of new keyframe distributions, it sends them back to the AR subsystem and updates the trajectory visualization to inform the user if the system adequately captured their intent, and whether the skill is likely to be executed successfully. This process of trajectory evaluation, constraint editing, and constraint application can be repeated until the user is satisfied.

V. SYSTEM VALIDATION

In order to validate the ARC-LfD system, we examine its operation within three test cases representative of potential task scenarios asked of robot manipulators. These case studies exemplify how ARC-LfD allows a user to demonstrate a skill, visualize the learned skill, then adapt the learned skill to two different environment setups (an "initial setup" and "secondary setup") using edited constraints. One of our research team members acted as a user to demonstrate the system's functionality. Eight kinesthetic demonstrations were provided as the basis for each skill using the Rethink Robotics Sawyer platform. Once the ARC-LfD system had generated a skill model learned from these demonstrations, the user was shown a sample trajectory of this skill. The user then edited and applied constraints with consideration given to the specific environment setup. ARC-LfD used the applied constraint to adapt the initial learned skill and sent a representation of the updated skill back to the user for visual inspection. Finally, the skill was executed on the robot.

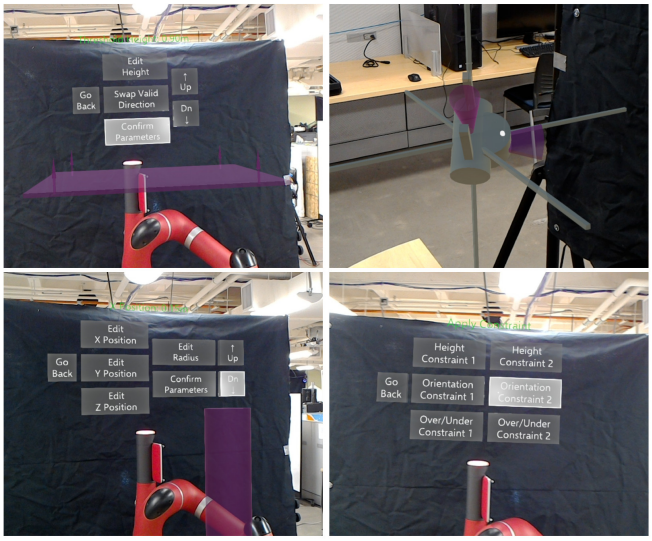

Fig. 5. Users can customize constraints from templates via the AR interface. After selecting a height (top left), orientation (top right), or over-under (bottom left) constraint, they edit its parameters and see the corresponding visualization update in real-time. Once satisfied, they can apply the newly edited constraint to the model by selecting it from the application menu (bottom right), and by selecting which keyframes the constraint should apply to. After this process, they will send a request to the robot to rebuild and revisualize the model using any new constraints, and evaluate whether the robot has correctly learned the skill.

These case studies demonstrate situations in which ARC-LfD allows a user to assess and edit a skill in response to changes in the environment or task setup. This illustrates a novel capability over CC-LfD as a user can craft and visualize constraint annotations to ensure successful model adaptation to differing task setups sans additional demonstrations. In these example applications, the entire process (skill visualization, creation and application of a constraint, skill updating within the CC-LfD subsystem, visualization of the updated skill, and approval of execution) took an average of 120 seconds per skill. Videos of the execution from each case study can be found at: https://youtu.be/G9TJIKVod4A.

A. Case Study I (Precise Placement): A Placement Task with Orientation Change at Goal Pose

The first case study emulates situations in which the goals of the task are modified after initial demonstrations are given.

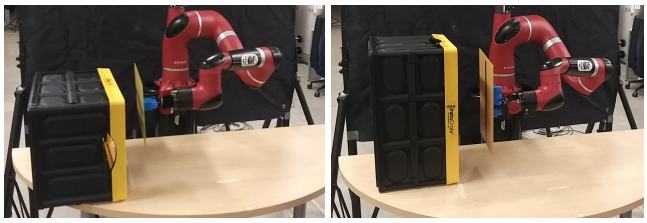

Fig. 6. For Case Study I, the robot inserts a rectangular object into a similarly-sized rectangular crate. In this case study, the user applies orientation constraints to the final keyframes in the trajectory in order to match the initial setup (left) with a horizontal crate or the secondary setup (right) with a vertical crate.

In this task, the robot's objective was to place a rectangular object into an upright crate, with minimal clearance. If the object was placed using the wrong orientation, a collision with the crate would occur. The user first provided demonstrations with varied orientations of the object. We evaluate the task for two different orientations of the crate, horizontal and vertical, with no additional demonstrations provided between conditions. In both cases, the user applied an orientation constraint to the last few keyframes of the task specifying the respective desired orientation. With the added constraints, the ARC-LfD system enabled the robot to successfully place the object without collision. The setup of this case study is shown in Figure 6.

B. Case Study II (Changing Environment): Introducing New Obstacles in a Pick-and-Place Task

In the second case study, the robot's task involved moving an object from one side of a table to another. This task is representative of pick-and-place kitting tasks with known initial/goal locations but configurations of obstacles that may change over time. For this case study, the user provided 8 demonstrations of moving the robot's arm across the table from right to left. The initial environment setup had no obstacles in the way. In the test condition, we placed stacked foam obstacles halfway across the table. By applying a height constraint, the user is able to edit the skill so that the robot could still complete the task without colliding with the new obstacles, without requiring additional demonstrations. This case study exemplifies how a generic constraint can be used in lieu of a simulated collision obstacle required by motion planning. Images from this case study are given in Figure 7.

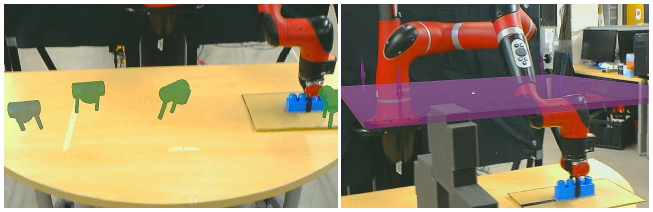

Fig. 7. In Case Study II, the robot completes a pick-and-place task either with or without an obstacle present. The initial environment (left) has no obstacles on the table, allowing the robot to freely move the object from right to left across the table. The test condition setup (right) introduces an obstacle halfway across the table, requiring the user to apply a height constraint that ensures the robot lifts its payload over the obstacle to complete the task.

C. Case Study III (Changing Goal): Moving the Receptacle for a Pouring Task

The third and final case study we conducted involved a task in which the robot poured a cup of material into a receptacle. The modification for this case study consisted of moving the receptacle to a different position. Using ARC-LfD's over-under constraint, the user was able to specify where on the table the pouring part of the task should begin. This allowed the robot to execute the cup pouring task successfully with two different end goal positions without any new demonstrations. Figure 8 illustrates the environment setup and constraint applications for this case study.

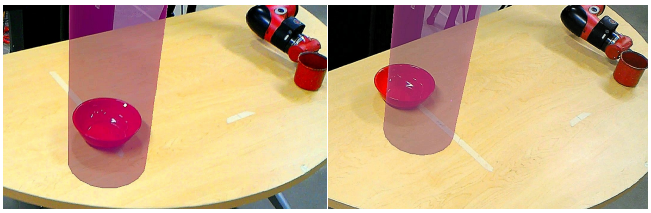

Fig. 8. Case Study III involves the robot pouring a cup into a bowl positioned at different points on the table. In the initial setup (left), the bowl is placed toward the front of the table, while in the test condition (right), the bowl is placed further back. In both cases, the user applies an over-under constraint to the trajectory representation in order to ensure the pouring motion takes place at the correct position.

D. Discussion

These three case studies exhibit the functionality of ARC-LfD and its ability to make LfD systems more robust. Case Studies I and III illustrate that ARC-LfD can make a set of demonstrations robust to changes in the task, provided sufficient variance of demonstrations in the set: through application of constraints to an existing skill, the robot can execute an altered version of a task. Case Study II shows how ARC-LfD can make learned skills robust to changes in the environment through using constraints that alter the skill trajectory to fit a new execution context. Furthermore, the interface of ARC-LfD enables users to conduct these alterations after demonstrations have been given, allowing for any-time editing of a skill. In addition to its functionality for verifying and previewing skills directly in the environment, ARC-LfD introduces a method for maintaining robotic skills even if the particulars of task and environment shift over time. We posit that ARC-LfD presents a safer-by-construction alternative to general end-to-end policy learning systems, trading generally unneeded levels of model expressivity for system transparency, enabling successful safer skill execution across a broad range of robotics tasks.

VI. CONCLUSION

ARC-LfD is proposed as a step toward producing practical, real-world-ready LfD systems that allow non-roboticists to conduct training and evaluation of robotic systems. The use of AR for in-situ visualizations relaxes the requirement of a model of the environment to use in simulation for verification of learned skills. Through visualizing a sample trajectory directly in the environment, users can preview the robot's skill execution contextualized by the actual environment itself. The control flow of ARC-LfD provides an improvement over CC-LfD, allowing users to separate demonstration from constraint application.

Finally, the proposed constraint editing interface relaxes the static environment assumption often levied for successful LfD skill deployment. ARC-LfD enables direct skill repair and editing, creating constraints contextualized in the environment and applying them to keyframes of an existing skill. Thus, ARC-LfD fills a critical technical gap in LfD systems, enabling long-term skill assessment and validation as the environment or task requirements change over time.

REFERENCES

[1] 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.

[2] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *ICML*, vol. 97. Citeseer, 1997, pp. 12–20.

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

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

[5] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz, "Keyframe-based learning from demonstration," *International Journal of Social Robotics*, vol. 4, no. 4, pp. 343–355, 2012.

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

[7] A. Jain, B. Wojcik, T. Joachims, and A. Saxena, "Learning trajectory preferences for manipulators via iterative improvement," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2013, pp. 575–583.

[8] B. Akgun, K. Subramanian, and A. L. Thomaz, "Novel interaction strategies for learning from teleoperation." *2012 AAAI Fall Symposium: Robots Learning Interactively from Human Teachers*, vol. 12, 2012.

[9] S. Wrede, C. Emmerich, R. Grünberg, A. Nordmann, A. Swadzba, and J. Steil, "A user study on kinesthetic teaching of redundant robots in task and configuration space," *Journal of Human-Robot Interaction*, vol. 2, no. 1, pp. 56–81, 2013.

[10] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, 2018.

[11] Y. Shen, G. Reinhart, and M. M. Tseng, "A design approach for incorporating task coordination for human-robot-coexistence within assembly systems," in *2015 Annual IEEE Systems Conference (SysCon)*. IEEE, 2015, pp. 426–431.

[12] F. Berkenkamp, A. Krause, and A. P. Schoellig, "Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics," *arXiv preprint arXiv:1602.04450*, 2016.

[13] H. Hedayati, M. Walker, and D. Szafir, "Improving collocated robot teleoperation with augmented reality," in *2018 ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2018, pp. 78–86.

[14] S. A. Green, M. Billinghurst, X. Chen, and J. G. Chase, "Human-robot collaboration: A literature review and augmented reality approach in design," *International Journal of Advanced Robotic Systems*, vol. 5, no. 1, p. 1, 2008.

[15] D. Szafir, "Mediating human-robot interactions with virtual, augmented, and mixed reality," in *2019 International Conference on Human-Computer Interaction (HCI)*. Springer, 2019, pp. 124–149.

[16] T. Yamamoto, N. Abolhassani, S. Jung, A. M. Okamura, and T. N. Judkins, "Augmented reality and haptic interfaces for robot-assisted surgery," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 8, no. 1, pp. 45–56, 2012.

[17] J. Weisz, P. K. Allen, A. G. Barszap, and S. S. Joshi, "Assistive grasping with an augmented reality user interface," *The International Journal of Robotics Research*, vol. 36, no. 5-7, pp. 543–562, 2017.

[18] M. E. Walker, H. Hedayati, and D. Szafir, "Robot teleoperation with augmented reality virtual surrogates," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2019, pp. 202–210.

[19] C. Brooks and D. Szafir, "Visualization of intended assistance for acceptance of shared control," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[20] K. Chandan, V. Kudalkar, X. Li, and S. Zhang, "Negotiation-based human-robot collaboration via augmented reality," *2019 AAAI Fall Symposium: AI for HRI*, 2019.

[21] E. Rosen, D. Whitney, M. Fishman, D. Ullman, and S. Tellex, "Mixed reality as a bidirectional communication interface for human-robot interaction," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.

[22] M. Walker, H. Hedayati, J. Lee, and D. Szafir, "Communicating robot motion intent with augmented reality," in *2018 ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2018, pp. 316–324.

[23] 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 *Robotics Research*. Springer, 2020, pp. 301–316.

[24] K. Kobayashi, K. Nishiwaki, S. Uchiyama, H. Yamamoto, S. Kagami, and T. Kanade, "Overlay what humanoid robot perceives and thinks to the real-world by mixed reality system," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE, 2007, pp. 275–276.

[25] T. Kot and P. Novák, "Utilization of the oculus rift hmd in mobile robot teleoperation," in *Applied Mechanics and Materials*, vol. 555. Trans Tech Publ, 2014, pp. 199–208.

[26] M. Diehl, A. Plopski, H. Kato, and K. Ramirez-Amaro, "Augmented reality interface to verify robot learning," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, pp. 378–383.

[27] D. Krupke, F. Steinicke, P. Lubos, Y. Jonetzko, M. Görner, and J. Zhang, "Comparison of multimodal heading and pointing gestures for co-located mixed reality human-robot interaction," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[28] D. Sprute, K. Tönnies, and M. König, "A study on different user interfaces for teaching virtual borders to mobile robots," *International Journal of Social Robotics*, vol. 11, no. 3, pp. 373–388, 2019.

[29] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[30] J. P. McIntire, P. R. Havig, and E. E. Geiselman, "Stereoscopic 3d displays and human performance: A comprehensive review," *Displays*, vol. 35, no. 1, pp. 18–26, 2014.