Efficient Programming of Manipulation Tasks by Demonstration and Adaptation

Sarah Elliott[1], Russell Toris[2], Maya Cakmak[1]

***Abstract*— Programming by Demonstration (PbD) is a promising technique for programming mobile manipulators to perform complex tasks, such as stocking shelves in retail environments. However, programming such tasks purely by demonstration can be cumbersome and time-consuming as they involve many steps and they are different for each item being manipulated. We propose a system that allows programming new tasks with a combination of demonstration and adaptation. This approach eliminates the need to demonstrate repetitions within one task or variations of a task for different items, replacing those demonstrations with a much more time-efficient *adaptation* procedure. We develop a Graphical User Interface (GUI) that enables the adaptation procedure. This GUI allows grouping, duplicating, removing, reordering, and repositioning parts of a demonstration to adapt and extend it. We implement our approach on a single-armed mobile manipulator. We evaluate our system on several test scenarios with one expert user and four novice users. We demonstrate that the combination of demonstration and adaptation requires substantially less time to program than purely by demonstration.**

I. INTRODUCTION

Robot Programming by Demonstration (PbD) is an effective way of programming manipulators and is widely used in industrial settings where robots repeat programmed actions thousands or millions of times. It is also a promising technique for programming mobile manipulators in homes or other task-oriented human environments such as retail stores, warehouses, or service industry domains. The diversity and dynamic nature of these environments will require end-users to do a lot more programming. Hence PbD interfaces must be simple enough for use by novice users and as efficient as possible to minimize the user's programming time.

However, even the simplest PbD systems that are currently available (*e.g.,* [2], [9], [12]) require at least one demonstration of the task in its entirety. If the task is long or complex, providing such a demonstration can be cumbersome and time-consuming. For example, consider a mobile manipulator being programmed to stock shelves in a retail store. Existing systems would require demonstrating how to handle each item separately, as they might each need to be grasped from a different point, placed to display a different side, and positioned differently relative to other items next to or behind it. Furthermore, most systems would require demonstrating the end-to-end task going from an empty shelf to a fully-stocked shelf, so that the robot can observe all variations of the shelf and learn how its actions need to be different in each case. As a result, programming a robot to stock all items in a store could take days or weeks.

Instead, we propose to replace parts of such lengthy demonstrations with a much more time-efficient *adaptation* procedure, building on already demonstrated parts of a repetitive task or transferring a demonstration provided for one task to another similar task. In this paper we present a simple PbD system in which manipulation actions are represented as a sequence of poses that can be relative to objects in the environment or a previously saved pose. We develop a Graphical User Interface that allows users to group, duplicate, remove, reorder, and reposition poses, as well as change their reference frames, so as to *adapt* a demonstration to fit a different part of a task or an entirely different task. We evaluate our approach with an implementation on the single-armed mobile manipulator Fetch using four different tasks. We demonstrate that the proposed duplication+adaptation procedure is much more efficient than demonstrating those sub-tasks or tasks, both for an expert user and five different novice users.

II. RELATED WORK

Programming by Demonstration (PbD), also known as Learning from Demonstration (LfD), is a way of programming robot behaviors [5], [3]. As the name suggests, the user demonstrates the desired behavior to the robot and the robot uses the demonstration to create a program. A majority of the research under the PbD umbrella has focused on learning a model of the robot's behavior from multiple demonstrations using different statistical representations (*e.g.,* [4], [18], [7], [6], [15]), with recent approaches that can capture complex task structures (*e.g.,* [14], [12]) or handle challenging task dynamics (*e.g.,* [10], [13], [16]). Some of the recent work has focused on allowing users to program actions through a single demonstration (*e.g.,* [2], [9], [12]).

Recent work has also focused on issues related to allowing everyday users to program robots [1], [8]. Several works involve GUIs similar to the one developed in this paper. For example, Alexandrova *et al.* present interactive visualizations of actions that allow users to fix errors in an action initialized from a single demonstration. Similarly, Kurenkov *et al.* show that editing can improve upon purely demonstrated programs [11]. In their study, participants preferred demonstrating the program kinesthetically and then editing it rather than programming purely with the GUI or purely by kinesthetic demonstration. These works focus on correcting errors encountered while programming rather than adapting a previously programmed action for a new

[1]Paul G. Allen School of Computer Science and Engineering, University of Washington, Seattle, WA 98195, USA.

[2]Fetch Robotics, Inc., San Jose, CA, 95131, USA.

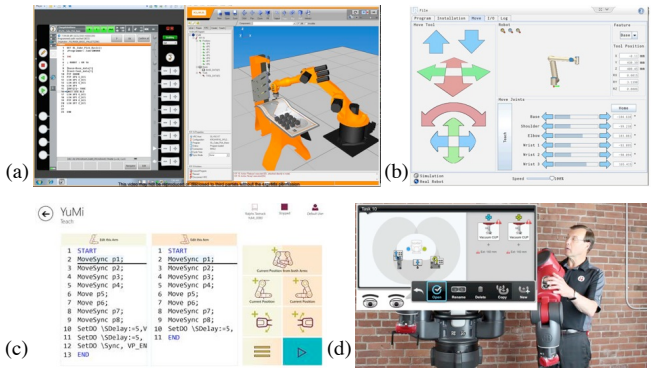

Fig. 1: Example GUIs for industrial robots: (a) Kuka's LWR, (b) Universal Robots's UR5, (c) ABB's YuMi, (d) ReThink Robotics's Baxter. These interfaces have some common elements with the GUI developed in this paper to support adaptation of actions.

purpose. Most relevant to our work, Stenmark *et al.* recently developed an intuitive programming interface for the Yumi robot, providing editing functions for the purpose of re-using previously created skills [17]. Their user study demonstrated that non-expert users obtained better skills when they adapted a skill made by an expert, instead of adapting a skill they made themselves.

Note that many industrial robots are also programmed with a combination of kinesthetic demonstrations and a GUI. Example GUIs are shown for four different industrial robots in Fig. 1. Many of these are highly complex and used by experts. For instance the user manual for the Yumi programming interface is 319 pages; the basic training documents for the UR robot is 216 pages. In addition, most of these systems are designed to do very specific tasks and might assume templates of tasks to be programmed (*e.g.,* pick-and-place, arrange on a grid) rather than allow the user to program arbitrary tasks.

III. APPROACH

A. Action Representation

Our work builds on the PbD system by Alexandrova *et al.* [2]. In our system an *action* is a sequence of 6-dimensional end-effector pose and corresponding binary gripper state (open/closed). Each pose has a reference frame; poses can be relative to (a) the robot's base, (b) a *landmark* in the environments, or (c) the pose preceding it. The key difference in representation from the previous work is the ability to use preceding poses as landmarks, allowing the grouping of poses that are relative to the same environmental landmark.

B. Demonstration

Actions are initialized with a single demonstration. At the beginning of the demonstration, the robot perceives the environment in order to detect task-relevant objects in its workspace. These objects are referred to as *landmarks*. A demonstration is given by physically moving the robot's arm to desired poses, possibly changing the gripper state, and saving the pose through a verbal command. If the saved pose is within a certain distance of one or more of the perceived landmarks, it is automatically considered to be relative to the closest of those landmarks. This means that the reference frame of the pose is set to be that landmark. Otherwise the default reference frame for a saved pose is the robot's base.

C. Action Adaptation

Once an action is initialized, it can be adapted by the user to meet the requirements of the task at hand. Actions can be modified in the following ways.

- Poses in an action can be duplicated. When a pose is duplicated, it is automatically added to the end of the sequence.
- Poses can be deleted.
- The order of poses can be changed.
- The position and orientation of a pose (x, y, z and *roll*, *pitch*, *yaw* values) can be modified.
- The reference frame of a pose can be changed.

In our implementation, these action modifications are done through a Graphical User Interface (GUI). This GUI is expressive enough to allow an action to be programmed entirely through such adaptations of an action initialized with a single arbitrary pose. The opposite is also possible. Users can demonstrate an action and make no modifications at all. Although these types of single-mode programming may be preferable in some cases, our work focuses on the potential benefits afforded by the combination of the two. We refer to this approach as Programming by Demonstration and Adaptation, or PbD+A in short form.

D. Action Execution

Once an action is programmed, it can be executed in a new scene. Before an execution, the robot perceives the environment to detect landmarks referenced in the action, which might have been displaced since the demonstration. Next it computes the absolute end-effector configurations for poses in the action that are *relative to landmarks*, as well as any pose that is *relative to those poses*. Next the robot uses motion planning to determine a path that visits all steps of the action in their new configuration. If a landmark is missing or a pose has moved out of the robot's reach, the robot cannot execute the action and it indicates this to the user through sound and a message on the GUI. Otherwise, it completes the action by executing the motion plan and changing the gripper state along the way as needed.

IV. SYSTEM

A. Platform

The robot platform used in this work is the Fetch robot which is a single-arm mobile manipulator [19]. The Fetch has a 7 degree of freedom (DOF) arm that can be manipulated physically by a human while in the gravity compensation mode. The arm has a 1-DOF gripper with parallel fingertips that open to a max distance of 10 cm. For perception of

Fig. 2: Elements of the GUI that support adaption of programmed actions: (a) right-click menu for changing the reference frame of a pose, (b) arrows and rings appear to drag or rotate a pose, (c) the list view in which each pose can be executed, copied, edited, deleted or hidden, using buttons on the list item.

objects the robot uses a Primesense Carmine 1.09 short-range RGBD sensor mounted on the robot's pan-tilt head.

B. Implementation

The backend implementation of our Programming by Demonstration and Adaptation (PbD+A) system uses elements of PR2 PbD[1] from the University of Washington. The Fetch's arm has 7 DOFs, so there are multiple joint configurations the arm could adopt for a given end-effector pose. We use the MoveIt[2] software package for Inverse Kinematics and motion planning. If the robot has detected a surface in the environment then it checks for collisions with the surface in order to avoid colliding with it during execution.

Objects in the world are detected from RGBD sensor data using the Rail Segmentation[3] software package from Georgia Tech. Objects are represented by the dimensions (length, width, height) of the bounding boxes of segmented point clouds on a tabletop. The reference frame is considered to be at the center of the bounding box. The distance threshold for a saved pose to be considered automatically relative to an object is 40cm. Information about all objects which have at least one associated pose is saved as part of the action.

When new information about the robot's environment is acquired, either before an execution or when requested directly by the user, the system attempts to match objects referenced in the demonstration in the newly observed environment. The similarity metric used is the L_1-norm of the dimensions of the objects. To be considered the same object, the L_1-norm of the dimensions must be less than 0.075. This value was determined experimentally.

[1]wiki.ros.org/pr2_pbd
[2]moveit.ros.org
[3]wiki.ros.org/rail_segmentation

C. Graphical User Interface

The GUI is a web browser interface implemented using the Robot Web Tools[4] collection of packages. It allows users to observe and explore the current state of the action through a visualization. The visualization includes the following elements, some of which are illustrated in Fig. 2.

- *Robot:* The current configuration of the robot is shown relative to the saved poses. When the user manipulates the real robot, the visualization updates accordingly. This helps to visualize the action in the robot's 3D workspace.
- *Actions:* The steps of an action are visualized with transparent copies of the robot's gripper, configured according to their pose and the current poses of action landmarks (*i.e.,* detected objects). The grippers appear orange if the poses they represent are reachable by the robot and grey if they are not reachable. Grey arrows connect consecutive poses together to indicate the order of in which the poses will be visited.
- *Objects and surface:* If the robot has detected a surface and objects in the environment, then these will also appear in the visualization. Objects appear as green boxes that reflect the dimensions of the bounding box for the object. The surface is represented by a purple rectangle.

In addition to the visualization, actions are also presented in a *list* view (Fig. 2(c)). A pose can be selected by clicking on either the gripper in the visualization or the list item in the list view. This will highlight both the gripper and the list item for that pose.

Users can perform the following operations using our GUI:

- *Detect objects:* This button causes the robot to point its head forward and down to look for a surface with

[4]robotwebtools.org/

objects and run its perception algorithm. If the robot detects a surface and objects, these will appear in the visualization.

- *Save pose:* This button saves the current end-effector pose and updates the visualization.
- *Delete pose:* This button, available on each item in the list view, removes the pose from the action and updates the visualization. It is also available from the menu that appears when a pose is selected in the visualization.
- *Copy pose:* This button, available on each item in the list view, duplicates a pose. The copy gets added to the end of the list and the visualization is updated.
- *Change pose:* When a pose is selected in the visualization, arrows and rings appear. By clicking and dragging the arrows, the pose can be translated. Similarly, the rings can be used to change its rotation.
- *Change reference frame:* The reference frame of a pose can be changed by right clicking on its visualization. A menu appears with possible frame options and the current reference frame is denoted by an "x".
- *Reorder poses:* The execution order of poses can be changed by dragging and dropping the items in the list view. This will also update the visualization.

The GUI can be accessed on a desktop computer or mobile phone or tablet. On a mobile phone, the visualization of the robot and the demonstrated poses will not appear, but the list of poses and all of the buttons will be present. Using a mobile phone is a convenient way for users to save and delete poses while giving a demonstration.

V. EVALUATION

Our system was evaluated in four experimental settings to illustrate the capabilities of our system compared to a baseline system that only allows basic GUI adaptations. These scenarios were motivated by the problem of picking items from or stocking items on shelves in a warehouse or retail setting.

A. Experimental Setup

The scenarios considered in this paper involve manipulating multiple objects on a surface as well as depositing them in a bin. The main objects are boxes that are 7.5 cm in length, 6.5 cm in width and 7 cm in height. We also use a bath toy of similar scale for testing transfer of tasks from one object to another. The surface is 74.5 cm in height and the bin is 47 cm in length, 32 cm in width and 20 cm in height. The bin rests on a surface (of another robot) that is 36.5 cm in height. Fig. 3 shows this configuration.

B. Tasks

We use 4 different scenarios to evaluate the systems. Each scenario has a *source* action and a *target* action. The *source* and *target* actions each have a start state and goal state. For each scenario, the user programs the source action and then that action is available for use in programming the target action.

Fig. 3: Experimental setup with the table and objects in front of the robot and a bin to its side.

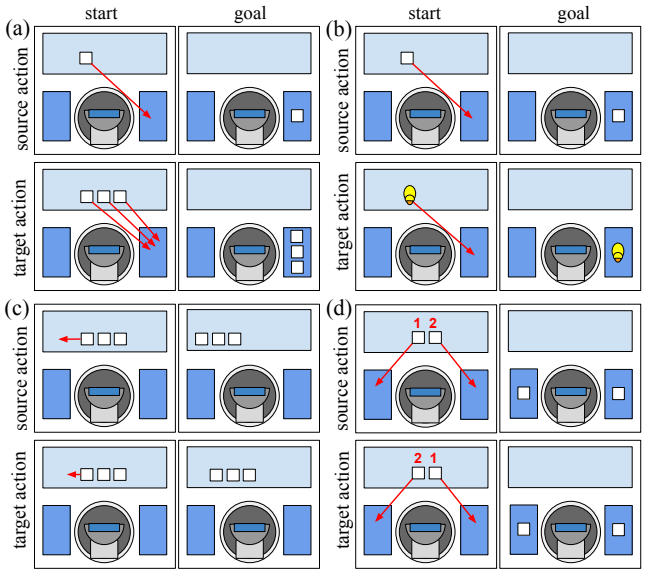

Fig. 4: Source and Target actions for each scenario.

Scenario 1: Fig. 4 (a)
Source Action: One box starts out on the surface in front of the robot. The goal is for the robot to move the box into the bin at its right side. The box can be deposited anywhere inside of the bin. *Target Action:* Three boxes start out on the surface 10 cm apart from each other. The goal is for the robot to move all of the boxes into the bin at its right side. The boxes can be deposited anywhere inside of the bin.

Scenario 2: (Fig. 4 (b))
Source Action: One box is on the surface. The robot needs to move the box into the bin on its right. This is the same as the source action for Scenario 1. *Target Action:* The bath toy is on the surface. The goal is to move the bath toy into the bin to the right of the robot.

Scenario 3: (Fig. 4 (c))
Source Action: Three boxes start out on the surface 10 cm

apart from each other. The robot moves the leftmost box 10 cm to the left. Then it moves the middle box and then the rightmost box. The result is that all of the boxes are 10 cm to the left of their starting positions.

Target Action: For the target action, three boxes start out on the surface 10 cm apart from each other. The goal is for the robot to move each of the boxes 5 cm to the left of their starting positions.

Scenario 4: (Fig. 4 (d))

Source Action: Two boxes start out on the surface 10 cm apart from each other. The robot moves the leftmost box into a bin on its left. Then it moves the rightmost box to a bin on its right.

Target Action: Two boxes start out on the surface 10 cm apart from each other. For the target action, the goal is to first move the rightmost box into the bin on the right and then move the leftmost box to the bin on the left.

C. Experimental Procedure

We compare our PbD+A system to a baseline system that allows only minimal modification of an action after the demonstration is complete. The baseline simply allows users to add and delete poses from the action's sequence. Some of these differences are visible in Fig. 5, which shows the GUIs of our system and the baseline system.

One expert user programmed performed all four programming tasks described in Sec. V-B, over three trials. Five non-expert users programmed the actions in Scenario 1, followed by Scenario 2. All five non-expert users were familiar with Robotics and had varying levels of experience with the Fetch robot. However they were new to our system.

Participants performed Scenario 1 followed by Scenario 2, using both the baseline and our system in counterbalanced order. Prior to programming actions in the two scenarios, participants performed an exercise to familiarize themselves with the robot and the different programming interfaces. First they used the baseline system to program the robot to wave its arm. Then they used the baseline system to program a waving action in the opposite direction, with the arm now positioned on the opposite side of the robot's body. Next they were asked to program the same two actions with the PbD+A system. Users were encouraged to explore the features of both systems during this time. Once participants felt comfortable with the tools, we moved on to the two scenarios.

For each scenario, we briefly describe how the *target* action can be programmed using the baseline and the PbD+A system using demonstrations and adaptations. After completing the two scenarios, the non-expert users completed a survey about their experience.

D. Experimental Results

We compare the two conditions (baseline vs. PbD+A) quantitatively in terms of the time it took to program each action, and qualitatively, in terms of the interaction steps used to program the actions. Table I summarized the experimental

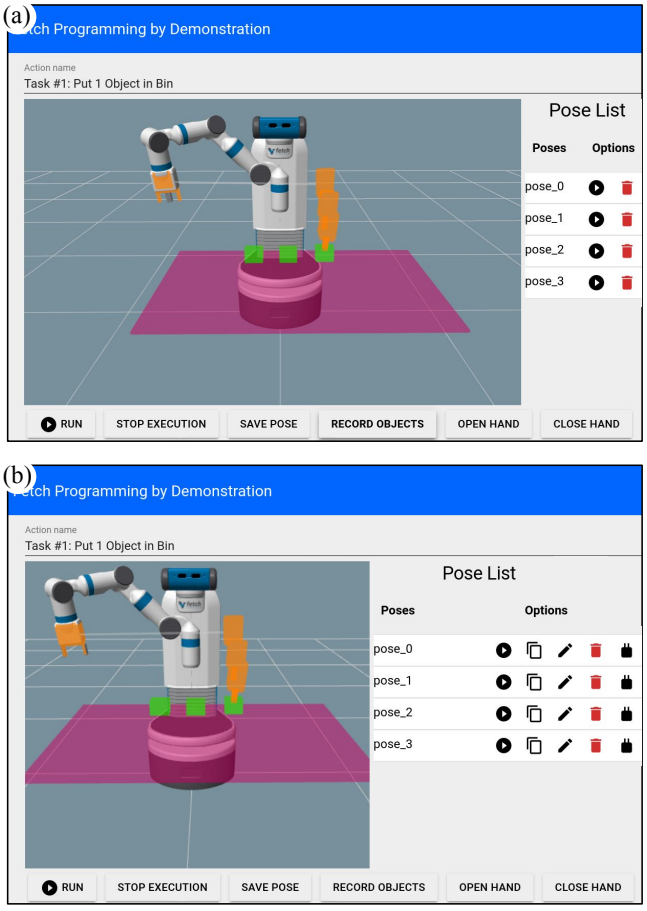

Fig. 5: (a) Baseline interface allowing minimal editing (b) Our PbD+A system allowing extensive modifications.

results for the expert programmer on four scenarios and novice programmers on the first two scenarios. Overall, we observe that programming the target task with PbD+A is much faster than programing it by demonstration, both for novice and expert users, across different tasks. In the following we go through the different tasks one by one and characterize how adaptation takes advantage of the similarities in the source and target actions to achieve these efficiency gains.

1) **Scenario 1:**

a) Baseline: Using the baseline system, programming the source action (placing one object in the bin) for this scenario involves the following steps:

1) **DEMO:** Move arm to center gripper above 1st object
2) **DEMO:** Open gripper
3) **DEMO:** Move arm so that object is inside gripper
4) **DEMO:** Close gripper
5) **DEMO:** Move gripper above bin
6) **DEMO:** Open gripper

On average, the non-expert users took 124 seconds to successfully program the source action. The expert user took an average of 42 seconds over 3 trials.

To program the target action (placing three objects in the bin), it is possible to build on the provided source action, even in the baseline system. The source action will deposit the first object in the bin. Additional demonstrations must be provided in order to deposit the other two objects into the bin. The additional demonstrations add new poses to the action. Hence, the process of programming the target action involves the following steps:

1) **DEMO:** Move arm to center gripper above 2nd object
2) **DEMO:** Open gripper
3) **DEMO:** Move arm so that object is inside gripper
4) **DEMO:** Close gripper
5) **DEMO:** Move gripper above bin
6) **DEMO:** Open gripper
7) **DEMO:** Move arm to center gripper above 3rd object
8) **DEMO:** Open gripper
9) **DEMO:** Move arm so that object is inside gripper
10) **DEMO:** Close gripper
11) **DEMO:** Move gripper above bin
12) **DEMO:** Open gripper

Although it is possible to build on the source action, it is still necessary to make nearly the same physical demonstration two more times. Non-expert users on average spent 245.8 seconds programming the target action. When performed the an expert user, this process took 88.3 seconds on average over 3 trials.

b) PbD+A: Using our system, the programming of the source action is the same as for the baseline system in this case. It took non-expert users 131.0 seconds on average to program the source action and the expert user 46.3 seconds on average over 3 trials.

Our system also allows the user to build on the source action, however no additional demonstrations are necessary. Instead, modifications are made to the source action using the GUI. Relevant poses are copied and then moved to locations corresponding to the other two objects. This process involves the following steps:

1) **MOD:** Copy each of the poses from the source action
2) **MOD:** Change the reference frames of the 2nd, 3rd and 4th copies to be relative to the poses that precede them. This has the effect of grouping the first four poses together. These are the poses for picking up the object. The poses for dropping the object in the bin do not need to move.
3) **MOD:** Move the first copied pose 10 cm to the right (*i.e.,* above the 2nd object). The other poses that are relative to this one will also move.
4) **MOD:** Copy each of the poses from the source action
5) **MOD:** Change the reference frames of the 2nd, 3rd and 4th copies to be relative to the poses that precede them
6) **MOD:** Move the first copied pose 20 cm to the right (*i.e.,* above the 3rd object).

In this case, the poses from the source action can be copied and then relocated as a group in order to avoid making multiple similar demonstrations. Our system replaces multiple instances of physically moving the robot's arm with mouse clicks. When performed by non-expert users, this process took on average 190.6 seconds, which was a substantial improvement over the 245.8 second taken in the baseline. For the expert user this process took 29 seconds on average over 3 trials, which was also significantly lower than the 88.3 seconds taken in the baseline.

				Non-Expert Users		Expert User	
Scenario & Action	System	# DEMO	# MOD	Avg. Time (s)	Std. Dev	Avg. Time (s)	Std. Dev
1, source	Baseline	6	0	124.0	30.3	42.3	7.5
1, target	Baseline	12	0	**245.8**	43.0	**88.3**	9.9
1, source	PbD+A	6	0	131.0	42.8	46.3	5.8
1, target	PbD+A	0	6	**190.6**	37.0	**29.0**	4.6
2, source	Baseline	6	0	124.0	30.3	42.3	7.5
2, target	Baseline	6	0	**118.6**	23.2	**52.3**	4.0
2, source	PbD+A	6	0	131.0	42.8	46.3	5.8
2, source	PbD+A	0	2	**56.6**	11.5	**14.0**	3.0
3, source	Baseline	21	0			103.7	9.1
3, target	Baseline	21	0			**90.0**	9.5
3, source	PbD+A	7	6			68.7	4.2
3, target	PbD+A	0	2			**32.0**	3.0
4, source	Baseline	12	0			89.0	6.6
4, target	Baseline	12	0			**89.3**	7.6
4, source	PbD+A	12	0			87.3	6.1
4, target	PbD+A	0	1			**18.7**	2.1

TABLE I: Time spent programming with baseline versus the PbD+A system. Types of interactions used for programming are characterized by the number of demonstration steps (DEMO) or modification steps (MOD).

2) **Scenario 2:**

a) Baseline: The source action here is the same as the one for the first scenario, which non-expert users were able to perform in an average of 124 seconds and the expert user performed in an average of 42 seconds.

In this case, the baseline approach cannot easily make use of the source action when programming the target action. The bath toy manipulated in the target action requires a slightly lower grasp than the box manipulated in the source action. It is necessary to start from scratch and demonstrate moving the bath toy into the bin. Using the baseline approach, non-expert users were able to program this target action in 118.6 seconds on average. When performed by the expert user, this process takes 90 seconds on average over 3 trials.

b) PbD+A: Once again, the source action is the same as in the first scenario, which non-expert users programmed in 131.0 seconds on average. The expert was able to program the source action in 42.3 seconds on average.

With our system, users were able to build on the source action to program the target action, and once again no additional demonstrations are necessary. In this case, the poses can be easily repositioned to grasp the bath toy, which requires a sightly lower grasp. Non-expert users were able

to program the target action in an average of 56.6 seconds and the expert user took an average of 14 seconds.

3) **Scenario 3:**

a) Baseline: The expert user was able to program the source action in an average of 103.7 seconds. In this case, the baseline approach cannot build on the source action to program the target action. Instead it is necessary to start from scratch and demonstrate moving each of the three objects individually. The result is giving 3 very similar demonstrations. When performed by an expert user, this process takes 90 seconds on average over 3 trials.

b) PbD+A: With our system, programming the source action took the expert user 68.7 seconds on average. This was much faster than programming the source action with the baseline system because our system allowed the user to only demonstrate picking up one object and then copy the resulting pose to make the whole action.

To program the target action the user was able to make use of the source action, and once again no additional demonstrations are necessary. In this case, the poses can be modified to move the objects over only 5 cm instead of 10 cm. One way to do this is to group the poses for moving a object together and then shift them all over by 5 cm. When performed by an expert user, this process takes 32 seconds on average over 3 trials.

4) **Scenario 4:**

a) Baseline: The expert user spent 89 seconds on average programming the source action. In this case, the baseline approach does not build on the source action to create the target action. When performed by the expert user, the target action took 89.3 seconds to program on average over 3 trials.

There is also a more complicated way to take advantage of the source action by deleting the poses associated with the leftmost object all together. This leaves an action that simply moves the rightmost object into the bin on the right. Then it is necessary to add to the action by demonstrating moving the leftmost object into the bin on the left. When performed by an expert user, this process takes 41.3 seconds on average over 3 trials.

b) PbD+A: With our system, users can make use of the source action to create the target action, and once again no additional demonstrations are necessary. In this case, the poses can be easily reordered to move the rightmost object before the leftmost object. When performed by an expert user, this process takes 18.7 seconds on average over 3 trials.

Overall, we found that our system reduced the time it took to program the robot in the the 4 scenarios described above. Programming the target actions took novice users 69% more time (58.2 seconds) on average when using the baseline system than when using our PbD+A system. The expert user was able to realize even more time savings, taking on average 260% more time (56.6 seconds) to program target actions using the baseline, compared to the PbD+A system. While the expert user was consistently faster than any of the novice users, these numbers show that users are expected to multiply their efficiency gains when they become experienced using the PbD+A system. This also suggests that the learning curve for the PbD+A system might be steeper than that of the baseline system.

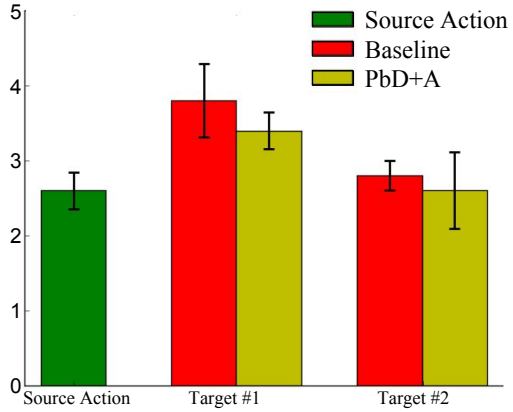

Fig. 6: The perceived difficulty of programming the source and target actions on a 5-point Likert scale (5:very difficult, 1:very easy) in Scenarios #1 and #2 with different systems.

E. Survey Results

All five of the participants agreed that the baseline system was *easier to learn* and *more intuitive*. However, 4 out of 5 participants also said that PbD+A was *easier to use once learned*. In addition, all of the participants stated that PbD+A was the most expressive and most efficient out of the two systems. This indicates that PbD+A has a non-zero barrier to entry, but that users found it to be valuable after becoming familiar with it.

As shown in Fig. 6, participants found adapting the previously programmed action using PbD+A slightly easier than starting from scratch using the baseline.

Open-ended user feedback indicated that people preferred PbD+A for programming “complicated” tasks or tasks with “multiple objects”. On the other hand, they preferred to program “easier”, “shorter” or “smaller” tasks with the baseline system. This indicates that there is some overhead associated with using the PbD+A system that users wish to avoid when programming less complicated tasks. Users also commented that the usability of the interface could be improved. One user said that “It was hard to remember all of the things I could do with the interface”, indicating that the design of the interface could be improved to be more discoverable and potentially reduce mental load on the user.

VI. DISCUSSION

This work demonstrates that adapting an existing action can be much faster than demonstrating a new action in some cases. The main contribution of our work is an action representation that supports intuitive adaptations as well as a GUI to enable novice users to perform such adaptations.

Although our adaptation GUI was successful in enabling large efficiency gains, it could be improved in many ways.

Some users commented that features of the GUI were difficult to remember and this difficulty may have contributed to the difficulty of programming using our system. This might not be an issue for repeat users, unlike the first time users in our study. In general, we believe that reducing the amount of repetitive physical demonstration necessary to program a robot will improve user experiences.

Our system could also be extended in several ways. The current PbD+A system supports programming only arm movements. However, the underlying software implementation is extensible so that an action could include head and base movements as steps in the action. In addition, the perception capabilities of the current system are limited to tabletop objects, which are represented by simple bounding boxes. Better object matching or recognition using color and more detailed shape information would make our system more robust and generalizable.

We observed that one result of adapting demonstrated actions is that the resulting action can be more consistent. When programming an action with repeated sub-tasks entirely by demonstration, there can be unwanted variation between the user's demonstration of those sub-tasks. Our system can prevent this because sub-tasks are essentially copies of each other.

When programming long, complex tasks with PbD+A, a significant amount of time is spent on considering how to structure the task. We hypothesize that many everyday tasks share a common structure or have similar sub-tasks. In the future, we would like to discover common types of robot tasks and create *templates* that have an appropriate, functional outline for the task. This could allow users to program actions purely by adaptation, using a template as the starting point.

VII. CONCLUSION

We propose a system that combines Programming by Demonstration with the ability to adapt the demonstrated action using a Graphical User Interface. We refer to this hybrid approach as PbD+A. In this system, a demonstration is a sequence of 6D *poses* that can be relative to objects in the environment or other poses. Once an action is initialized through a single demonstration, it can be modified using the GUI designed for action adaptations. The GUI allows for editing the position and orientation of individual poses in the action as well groups of poses. A user can also copy, remove and reorder poses within in action.

The advantages of this approach over existing approaches is that it does not require the user to make multiple demonstrations of similar parts of a task. We describe an implementation of the system using a Fetch robot and evaluate the system by having 1 expert and 5 novice users program the robot to manipulate a set of objects on a surface. We demonstrate that PbD+A avoids repeated demonstrations of similar sub-tasks and therefore reduces the time needed to program complex tasks. PbD+A has the potential to allow robot programmers to more efficiently program a wide range of tasks, especially those with repetitive sub-tasks.

ACKNOWLEDGMENTS

This work was supported by Fetch Robotics and the National Science Foundation Award IIS-1552427 "CAREER: End-User Programming of General-Purpose Robots."

REFERENCES

[1] Baris Akgun, Maya Cakmak, Jae Wook Yoo, and Andrea Lockerd Thomaz. Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In *ACM/IEEE international conference on Human-Robot Interaction*, pages 391–398, 2012.

[2] S. Alexamdrova, M. Cakmak, K. Hsaio, and L. Takayama. Robot programming by demonstration with interactive action visualizations. In *Robotics: Science and Systems (RSS)*, 2014.

[3] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[4] Christopher G Atkeson and Stefan Schaal. Learning tasks from a single demonstration. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 2, pages 1706–1712, 1997.

[5] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Robot programming by demonstration. In *Springer Handbook of Robotics*, pages 1371–1394. Springer, 2008.

[6] Sylvain Calinon and Aude Billard. Statistical learning by imitation of competing constraints in joint space and task space. *Advanced Robotics*, 23(15):2059–2076, 2009.

[7] S. Chernova and M. Veloso. Confidence-based policy learning from demonstration using gaussian mixture models. In *Intl. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, 2007.

[8] Sonia Chernova and Andrea L Thomaz. Robot learning from human teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(3):1–121, 2014.

[9] Christian Groth and Dominik Henrich. One-shot robot programming by demonstration using an online oriented particles simulation. In *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on*, pages 154–160, 2014.

[10] Seungsu Kim, Elena Gribovskaya, and Aude Billard. Learning motion dynamics to catch a moving object. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 106–111, 2010.

[11] A. Kurenkov, B. Akgun, and A. L. Thomaz. An evaluation of gui and kinesthetic teaching methods for constrained-keyframe skills. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[12] A. Mohseni-Kabir, C. Rich, S. Chernova, C. L Sidner, and D. Miller. Interactive hierarchical task learning from a single demonstration. In *ACM/IEEE Intl. Conf. on Human-Robot Interaction*, 2015.

[13] K. Mülling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, 2013.

[14] S. Niekum, S. Chitta, A.G. Barto, B. Marthi, and S. Osentoski. Incremental semantically grounded learning from demonstration. In *Robotics: Science and Systems*, volume 9, 2013.

[15] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2009.

[16] J. Schulman, J. Ho, C. Lee, and P. Abbeel. Learning from demonstrations through the use of non-rigid registration. *International Journal of Robotics Research*, 2013.

[17] M. Stenmark, M. Haage, and E.A. Topp. Simplified programming of re-usable skills on a safe industrial robot: Prototype and evaluation. In *ACM/IEEE Intl. Conf. on Human-Robot Interaction*, 2017.

[18] A. Ude, C. G. Atkeson, and M. Riley. Programming full-body movements for humanoid robots by observation. *Robotics and Autonomous Systems*, 47:93–108, 2004.

[19] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich. Fetch & freight: Standard platforms for service robot applications. In *IJCAI Workshop on Autonomous Mobile Service Robots*, 2016.