

Robot Programming by Demonstration with Interactive Action Visualizations

Sonya Alexandrova, Maya Cakmak
Computer Science & Engineering

University of Washington, Seattle, WA
Email: sonyaa,mcakmak@cs.washington.edu

Kaijen Hsiao
Bosch Research and Technology Center
Palo Alto, CA
Email: kaijen.hsiao@us.bosch.com

Leila Takayama¹
Willow Garage, Inc.
Menlo Park, CA
Email: takayama@google.com

Abstract—Existing approaches to Robot Programming by Demonstration (PbD) require multiple demonstrations to capture task information that lets robots generalize to unseen situations. However, providing these demonstrations is cumbersome for end-users. In addition, users who are not familiar with the system often fail to demonstrate sufficiently varied demonstrations. We propose an alternative PbD framework that involves demonstrating the task once and then providing additional task information explicitly, through interactions with a visualization of the action. We present a simple action representation that supports this framework and describe a system that implements the framework on a two-armed mobile manipulator. We demonstrate the power of this system by evaluating it on a diverse task benchmark that involves manipulation of everyday objects. We then demonstrate that the system is easy to learn and use for novice users through a user study in which participants program a subset of the benchmark. We characterize the limitations of our system in task generalization and end-user interactions and present extensions that could address some of the limitations.

I. INTRODUCTION

General-purpose mobile manipulators have the physical capability to perform a diverse range of useful tasks in human environments. However, pre-programming these robots for all potential uses is impossible—every combination of user, environment and task has different requirements for what the robot needs to do. Instead, *Programming by Demonstration* (PbD) [3] (also known as *Learning from Demonstration* [2]) techniques aim to enable end-users to program a general-purpose robot for their specific purposes, by demonstrating the desired behavior in the context of use.

Existing techniques for PbD require multiple, often many, demonstrations of the same task. Multiple demonstrations provide information that allows the robot to generalize learned actions to unseen situations. Different types of such information include invariance of certain state variables [10, 11], allowed variance of certain state variables [5, 6], characteristics of the task in different parts of the state space [15, 8], relateness to objects in the environment [12], segmentations of the task [12], task constraints [14] or partial-ordering of task steps [13]. To achieve generalizable tasks, these techniques require the user to provide representative demonstrations that cover the state space or alternative settings of the environment. However,

¹This author is currently affiliated with Google[x] but contributed to the work while affiliated with Willow Garage, Inc.

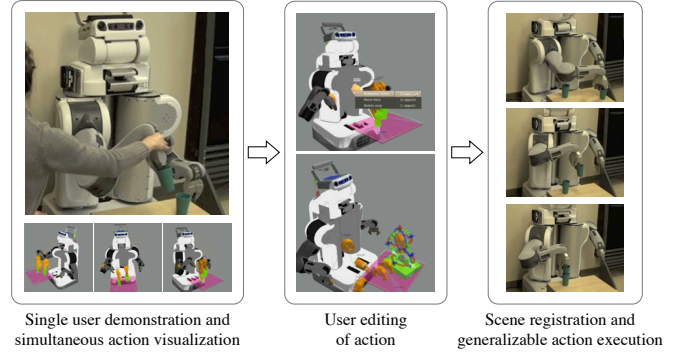


Fig. 1. Framework for PbD with interactive action visualization.

potential users of the system are often unaware of these requirements and are likely to have inaccurate mental models of how the robot learns from the provided demonstrations. Prior work in the area of human-robot interaction has demonstrated that this mismatch of mental models results in datasets that do not satisfy the system’s requirements for learning generalizable tasks [1, 16]. Furthermore, these users state that they dislike having to repeat the same task demonstration [1].

To address these issues, we propose an alternative PbD framework that involves demonstrating the task only once. The user provides additional task information explicitly, through interactions with a visualization of the learned action, rather than providing additional demonstrations. We propose a simple yet powerful action representation that supports the learning of generalizable tasks with this approach by allowing interactive visualizations of the learned action. We present and evaluate a system that implements this approach on a two-armed mobile manipulator. We demonstrate the range of manipulation tasks that can be achieved with our system and validate its usability and intuitiveness through a user-study (N=10).

II. APPROACH

A. Action Representation

We represent an *action* as a sparse sequence of end-effector states relative to discrete landmarks in the environment. We denote the n th action as $\mathcal{A}_n = \{(\theta^f, f, g)_k : k = 1..K\}$, where $\theta^f \in SE(3)$ represents the robot’s 6 Degree-of-Freedom (DoF) end-effector configuration (translation and rotation) in the frame of reference $f \in \{\ell_0, \dots, \ell_{N^d}\}$, which can be any

of the N^d available landmarks (including the robot base ℓ_0). $g \in \{0, 1\}$ represents its binary gripper state (open or closed). We represent landmarks in the environment as a tuple $\ell_i = (\phi, \tau, v^\tau)$, where $\phi \in SE(3)$ is the last recorded configuration of the landmark in the robot's coordinate frame, $\tau \in S^\tau$ is the type of the landmark and $v^\tau \in \mathbb{R}^{d^\tau}$ is a type-dependent descriptor of the landmark consisting of a vector of reals.

B. Action Initialization by Demonstration

Actions are initialized with a single demonstration. This involves directly adding *action states* $(\theta^f, f, g)_t$ into the action by demonstrating each state. Before a demonstration, the robot accumulates the list of potential landmarks $L^d = \{\ell_0, \dots, \ell_{N^d}\}$ by searching for landmarks in the environment and recording their initial configuration. Then the user manipulates the robot's arms and starts to add steps into the action at chosen configurations. At any time, the current action state $(\theta^f, f, g)_t$ is determined by the robot's current arm configuration ζ_t (controlled by the human) and the configuration of the landmarks in L^d (fixed at the start of demonstration). The frame of reference f for the state is determined by the proximity of the end-effector to the landmarks. If no landmarks are within a certain threshold d_{max} (empirically determined), then $f = \ell_0 = (\phi_0, robot, null)$ where ϕ_0 is the origin of the robot's coordinate frame, i.e., the pose is considered to be *absolute*. Otherwise, the frame of reference is the nearest landmark, i.e., $f = \ell_i$ for $i = \arg \min_{i \in 1 \dots N^d} d(\theta^{\ell_0}, \phi_i)$ and the pose is considered to be *relative*. The robot's relative end-effector configuration θ^f is computed by transforming the absolute end-effector configuration into the f coordinate frame, i.e., $\mathbf{T}_{\theta^f} = \mathbf{T}_{\phi}^{-1} \mathbf{T}_{\theta^{\ell_0}}$, where \mathbf{T}_{θ} is the transformation matrix corresponding to the configuration θ .

After an action has been initialized, it can be cleared and re-initialized, edited (Sec. II-C), or executed (Sec. II-D).

C. Action Editing

End-user programming (EUP) is an active research area that aims to enable everyday people who are not professional software developers to create custom programs that meet their particular needs [7, 17, 9]. Commonly studied problems in the EUP literature include programming spreadsheets and webpages. One of the core concerns in EUP is to provide the user with an accurate mental model of what the system can represent so as to allow them to directly edit representations. For instance, in spreadsheets the user can program simple functions by providing example input-output pairs; however, more complex functions require the user to edit the formula of a function. Our work aims to apply the same idea to robot PbD. To this end, it is important for the system implementation of our approach to provide visualizations of the action representation described in Sec. II-A. Furthermore, these visualizations should be interactive, to allow the user to directly edit components of this representation.

We allow the user to perform the following edits on the steps $(\theta^f, f, g)_k$ of a learned action \mathcal{A}_n .

- **Reference frame change:** During the initial demonstration each pose is automatically assigned a frame of reference based on its proximity to landmarks. After the demonstration the user can change the reference frame f of any action step k to be another available landmark; $f \leftarrow f_{edit} \in L_d$.
 - **Transformation of configuration:** The user can directly edit the configuration of saved action states; $\theta^f \leftarrow \theta_{edit}$.
 - **Delete pose:** The user can remove action states that were saved during the demonstration; $\mathcal{A}_n \leftarrow \mathcal{A}_n \setminus (\theta^f, f, g)_{edit}$
- In addition to these edits on the programmed action, we allow the user to edit the robot's representation of the world:
- **Delete landmark:** The user can remove landmarks (except robot base ℓ_0) from the set of available landmarks that are considered part of the learned action; $L_d \leftarrow L_d \setminus \ell_{edit}$

D. Action Execution

To perform an action \mathcal{A}_n the robot first accumulates the list of current landmarks $L^e = \{\ell_0, \dots, \ell_{N^e}\}$. The robot then needs to perform a rigid registration between landmarks $L^a \subset L^d$ referenced in \mathcal{A}_n and the set of landmarks L^e available at execution time. This is done through a simple bipartite graph cut algorithm that starts with all possible pairings between landmarks of same type in L^a and L^e . The similarity between two landmarks of the same type is computed with a type-specific, real-valued similarity function $s^\tau(v_1, v_2)$ over the landmark descriptors; $s : \mathbb{R}^{d^\tau} \times \mathbb{R}^{d^\tau} \rightarrow \mathbb{R}$. The algorithm greedily removes the pair of landmarks with highest similarity from the graph until all landmarks L_a of the action are registered. If at least one landmark in L_a remains unregistered, the action is deemed un-executable in the current environment.

Once all landmarks are registered, the absolute end-effector poses required to execute the task are computed based on the current configuration of the landmarks. The desired arm configuration to achieve each end-effector pose is computed using inverse kinematics (IK). If there is at least one pose that has no IK solutions, the action is deemed un-executable in the current environment. Otherwise, the robot reproduces the action by moving through these configurations with a certain velocity profile. After reaching the k th configuration θ_k , the robot changes the gripper state to g_k if different from the current gripper state.

III. SYSTEM

A. Platform

The robot platform used in this work is PR2 (Personal Robot 2), a mobile manipulator with two 7-DoF arms and an omnidirectional base. PR2's arms are passively balanced through a spring counterbalance system. This provides natural gravity-compensation, allowing users to move the robot's arm kinesthetically. When powered, the arms can carry up to 2.2 kg. Each arm has a 1 DoF, under-actuated gripper.

The software written for this work is developed within ROS (Robot Operating System) and was released as an open-source

TABLE I
COMPONENTS OF THE SYSTEM STATE.

Experiment state	Actions programmed so far $\mathbb{A} = \{\mathcal{A}_i\}_{i=1}^{N_A}$, current action index $n \in \{1..N_A\}$
Robot state	Joint configurations ζ^R, ζ^L , gripper states $g^R, g^L \in \{0, 1\}$ and arm stiffnesses $\alpha^R, \alpha^L \in \{0, 1\}$
World state	Set of available landmarks $L = \{\ell_1, \dots, \ell_N\}$
Action state	(Fully determined by the <i>robot</i> and <i>world</i> states) States of the robot end-effectors relative to the landmarks in the environment; $(\theta^f, f, g)^R, (\theta^f, f, g)^L$

package². During kinesthetic interactions with the robot, users wear a Shure wireless microphone headset to give commands to the robot. Speech recognition is done with Pocketsphinx, and text-to-speech on the robot is done with a common ROS package³. For perception of landmarks, we use the Kinect sensor mounted on PR2’s pan-tilt head.

B. Implementation Details

Some specifics of our system within the general framework described in Sec. II are as follows. Since the robot has two arms, two separate actions \mathcal{A}_n^R and \mathcal{A}_n^L are maintained for the right and left arms. These actions have equal numbers of steps, as poses are added to both actions simultaneously. During execution, the velocity profiles are adjusted such that both arms reach the next configuration at the same time.

Our implementation has two types of landmarks: (a) tabletop objects with 3D volume and (b) 2D fiducial markers ($S^\tau = \{object, marker\}$). Objects are detected using the ROS tabletop cluster detector⁴, which determines the dominant plane in the Kinect point cloud and then clusters portions of the point cloud above this plane. We represent the landmark corresponding to each object with its bounding box: the frame of reference (ϕ) for the landmark is the box center and the descriptor of the landmark is the dimensions of the box, $v^{object} = (width, length, height)$. Fiducials are detected using the Alvar-based ROS package⁵, which provides its 6-DoF configuration (ϕ) and a unique identifier used as the descriptor of the landmark, $v^{marker} = (id)$. To measure the similarity between landmarks of type *object* we use the L_1 -norm. The similarity between *marker* landmarks is infinite for markers of same id and zero for different ones. In other words, if the action references landmarks with a unique identifier, the execution of that action requires that particular landmark to be present. Similarly, we put a threshold for matching *object* landmarks to avoid matching very different landmarks in the absence of the desired ones.

C. Dialog System

During a demonstration, the user interacts with the PbD system through a simple state-based dialog system with a finite command set. The response to each command differs

TABLE II
LIST OF DIALOG COMMANDS AND THEIR EFFECTS ON THE SYSTEM STATE. FOR COMMANDS THAT DO NOT SPECIFY RIGHT/LEFT, THE OPERATION IS PERFORMED FOR BOTH ARMS, I.E., \mathcal{A}_n^R AND \mathcal{A}_n^L .

Command	Effect of command on system state
RELEASE/HOLD RIGHT/LEFT ARM	Changes the stiffness of the arms; $\alpha^s \leftarrow \neg \alpha^s$ where $s \in \{R, L\}$.
OPEN/CLOSE RIGHT/LEFT HAND	Changes the state of the gripper; $g^s \leftarrow \neg g^s$ where $s \in \{R, L\}$.
CREATE NEW ACTION	Adds an empty action to the action set; $n \leftarrow A +1, \mathcal{A}_n \leftarrow \{\}, A \leftarrow A \cup \mathcal{A}_n$.
RECORD OBJECT POSES	Detects all landmarks in the environment; $L_d \leftarrow L_t$.
SAVE POSE	Saves the current state into the current action; $A_n \leftarrow A_n \cup (\theta^f, f, g)_t$.
EXECUTE ACTION	Starts the execution.
STOP EXECUTION	Stops ongoing execution.
DELETE LAST POSE	Removes the last step from the current action;
DELETE ALL POSES	Clears the current action; $\mathcal{A}_n \leftarrow \{\}$.
NEXT/PREV. ACTION	if $(n > 1 \text{ and } n < A)$: $n \leftarrow n \pm 1$

based on the system state, which is a combination of the *experiment*, *robot* and *world* states (Table I). The experiment state involves the set of skills that have been programmed so far and the index of the current skill. The robot state involves the arm configurations, gripper states (0:closed, 1:open) and arm stiffnesses (0:relaxed, 1:stiff). The world state involves the last perceived landmarks.

The commands are explained in Table II, excluding the command TEST MICROPHONE which has no effect on the system state. The response to a command involves (i) a change in the system state, (ii) a speech response uttered by the robot, and (iii) a gaze action or head gesture. For example, after the user gives the RELAX RIGHT ARM command, the robot lowers the stiffness of the right arm and says “Right arm released” while glancing towards the right arm.

Programming one action typically has the following progression: the user creates a new action with the command CREATE NEW ACTION, and the robot responds “Created action n .” Then the command RECORD OBJECT POSES is used to trigger the robot’s landmark search. The robot responds with “Object poses recorded” if there is at least one object in the environment; otherwise it says “No objects detected.” The user then relaxes the robot’s arms (if needed), moves them to desired target poses, and says SAVE POSE or OPEN/CLOSE RIGHT/LEFT HAND.” Finally, the command EXECUTE ACTION is used for reproducing the action in a new situation. If the action is not executable (Sec. II-D), the robot says “Cannot execute action”. Otherwise, it says “Starting execution” and starts to move its arms. Upon reaching the last pose in the action, the robot says “Execution ended.” During execution, the robot only responds to the command STOP EXECUTION.

²http://wiki.ros.org/pr2_pbd

³http://wiki.ros.org/sound_play

⁴http://wiki.ros.org/tabletop_object_detector

⁵http://wiki.ros.org/ar_track_alvar

D. Graphical User Interface

The graphical user interface is a critical component of our system for the visualization and editing of programmed actions. Fig. 2(a) shows the visualization of a sample action and illustrates the different visualization elements that show the current action representation. These elements include:

- *Landmarks:* Object landmarks are shown with green transparent boxes that correspond to the bounding box. Similarly marker landmarks are shown with a flat green box aligned with the detected fiducial.
- *Poses:* End-effector poses corresponding to the steps of the programmed action are visualized with orange 3D models of the robot’s gripper. The gripper is open or closed according to the demonstration.
- *Relativeness:* Yellow arrows from end-effector pose visualizations to landmark visualizations indicate that the landmark is the frame of reference for the pose.
- *Order/progression:* Thin gray lines between consecutive poses, together with numbers on top of each keyframe, indicate sequential ordering relationships of poses.

The end-user editing functionality described in Sec. II-C is implemented by making these visualizations interactive. The user can navigate in a third person view in 3D using the mouse. When the mouse scrolls over anything that is interactive, the visualization of that element gets highlighted.

- *Reference frame change:* Users can right click on saved poses to reveal a linear popup menu (Fig. 2(b)). The first entry in this menu is “Reference frame” which points to a second layer menu with the list of possible landmarks, with the current (automatically assigned) landmark selected. By selecting a different landmark in this list, the user can change the frame of reference for this pose.
- *Transformation of configuration:* If the user clicks once on a visualized end-effector pose, this reveals a 6-DoF control around the pose (Fig. 2(c)), allowing the user to pull the pose in any of three directions or rotate it around these directions. This allows the user to change the poses of the action steps saved during the initial demonstration. The control disappears if the user clicks on the end-effector again or if a different end-effector is selected. Another way to edit the configuration of saved end-effector poses is to reassign the pose to the current arm configuration. This functionality is provided through the third entry in the pop-up menu (Fig. 2(b)).
- *Delete pose:* Users can delete a particular action step (Fig. 2(b)) by selecting Delete from the pop-up menu. This removes both right and left arm poses, even though the user interacts with only one of the two.
- *Delete landmark:* Users can also remove landmarks from the robot’s world representation by right clicking on the visualization of the landmark. This is the only action allowed on landmark visualizations (Fig. 2(d)).

Our interface also visualizes the detected table plane (pink colored plane in Fig. 2); however, it is not interactive because we did not consider the detected table plane as a landmark.

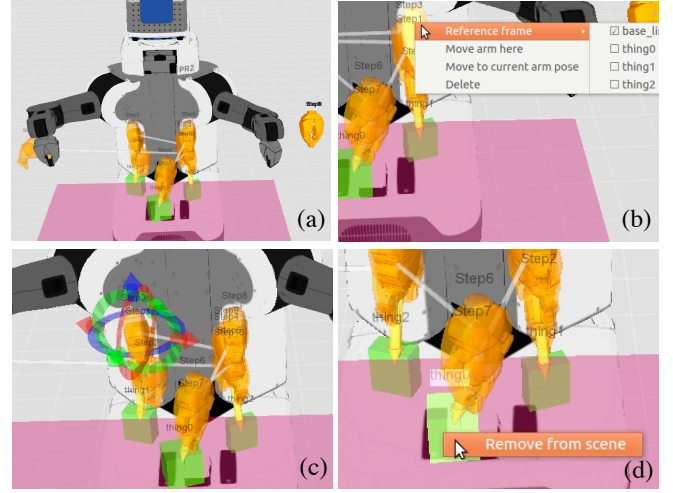


Fig. 2. Visualization and editing elements in the Graphical User Interface.

This was due to large variances in the plane detection method used. Instead we attached a fiducial to the table to allow relativeness to the table. In addition to the 3D visualization, our GUI has a side panel that has buttons corresponding to each speech command. The panel has an icon for each programmed action allowing direct navigation to previously programmed actions.

IV. SYSTEMATIC EVALUATION

We first demonstrate our system’s ability to learn a diverse set of manipulation tasks. For this we created a benchmark task list that consists of 12 representative manipulation tasks (Table III). The objects used in these tasks are shown on Fig. 5(a).

A. Protocol

All tasks are separately programmed and tested by one of the authors. A subset of the tasks were also programmed by another author to ensure consistency. For each task, the experimenter first spends time programming an action through a combination of demonstrations and editing. The experimenter can execute the action as part of this process and start from scratch if needed. Once the experimenter is pleased with the learned action, it is tested in five different initial conditions that differ in terms of the positions and orientations of the objects involved in the task. For each test, the success of the task is noted. Success is task specific and it is judged by the experimenter. For instance, for the task of picking up and placing an object (Task 1), the success criterion is the object being located at a fixed position and orientation at the end of the execution. For the task of stacking cups (Task 6), it is required that all cups are stacked together, whereas the position of the stack of cups does not matter.

B. Results

Figure 3 shows snapshots from successful executions of each of the twelve tasks⁶. Figure 4 shows starting conditions for several of the tasks, along with whether that particular

⁶Sample executions can be viewed at <http://youtu.be/ZHkOeAmWsvk>

TABLE III
BENCHMARK TASK LIST.

#	Type	Description	Success
1	Single-armed pick up and place (1)	Pick up a rigid cleaning sponge from the top and place it in a fixed spot.	4/5
2	Single-armed pick up and place (2)	Pick up a bottle from the side and place it in a fixed spot.	4/5
3	Single-armed non-prehensile manipulation	Push a cleaning sponge without grasping it.	5/5
4	Multi-armed pick up and place (1)	Pick up an object (tray) with both arms and place it in a fixed spot.	3/5
5	Multi-armed pick up and place (2)	Constrain a plate with one hand and pick it up with the other.	5/5
6	Multi object organization (1)	Stack cups.	5/5
7	Multi object organization (2)	Drop smaller objects (3 building kit blocks) into a box.	4/5
8	Assembly (1)	Screw the lid onto a bottle.	3/5
9	Assembly (2)	Put the lid on a pan.	4/5
10	Disassembly (1)	Unscrew the lid on a jar.	5/5
11	Disassembly (2)	Remove the lid from the pan.	4/5
12	Object state change	Close a lunchbox.	4/5

condition resulted in a successful execution. Table III shows task success statistics. We make the following observations.

Representational power. The diversity of the tasks that can be programmed with our system (Fig. 3) demonstrates its representational power. A large number of everyday tasks can be represented as end-effector movements and gripper actions relative to objects (landmarks) in the environment. By allowing multiple landmarks, our representation captures tasks that involve relations of multiple objects. For instance, in the cup stacking task (Task 6) the gripper closing poses are relative to the cups that go on top of the stack and the gripper opening poses are relative to the cup on which other cups get stacked (visualized in Fig. 2). This capability is also exploited in assembly tasks (Task 8 and 9).

The perception bottleneck. While the success rate is overall high (83% on average) we observed common failures in a subset of the tasks. Only four out of the 12 tasks were successful in all five tests. All failures in task executions were due to landmark localization errors: in certain situations (e.g., changed perspective on objects as in Fig. 4 (top row)) the robot would localize the object with a slight offset, resulting in a different landmark origin relative to the true geometry of the object. Consequently the grasp on the object was more likely to fail or be different from the demonstration. This was especially problematic in tasks that require a high level of precision in positioning the grippers relative to the objects, such as picking up a tray by its handles, screwing the lid on the bottle or putting the lid on the pan (Tasks 4, 8 and 9).

V. VALIDATION WITH END-USERS

We evaluated the usability of our system as an end-user tool through a user study in which participants programmed a subset of the tasks considered in Sec. IV.



Fig. 3. Frames from sample executions of the tasks during testing.

A. Setup

For the user study, the robot was positioned near a table with a computer. A large table was placed in front of the robot for manipulation tasks and the objects involved in the tasks to be programmed were made available (Fig. 5(b)). Participants had access to the robot, as well as a screen and a mouse to interact with the GUI. The screen was pointed towards the robot such that the participant could view the visualization of the task as they were interacting with the robot. Participants were equipped with a wireless microphone headset to free their hands for kinesthetic interactions with the robot.

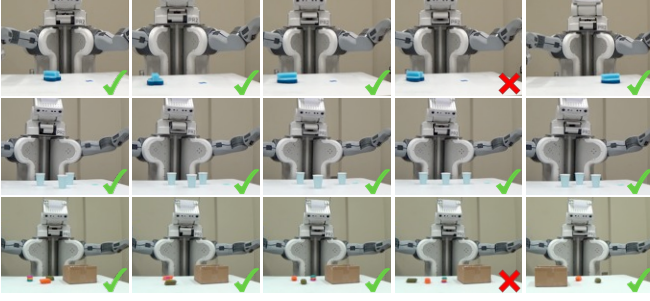


Fig. 4. Examples of test conditions during evaluation.

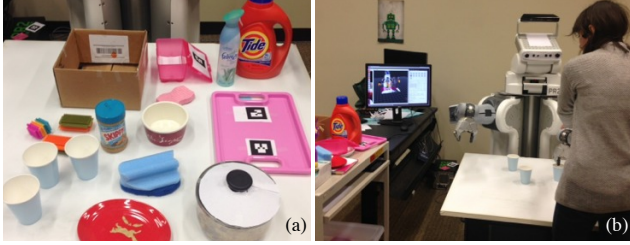


Fig. 5. (a) Objects used in benchmark tasks. (b) Experimental setup for the user study.

B. Protocol

Each participant was scheduled ahead of time for a one-hour time slot. Upon arrival, each participant was asked to sign an informed consent form. The goal of the research was described, along with the general properties of the system. The participant was given a full list of available commands with their descriptions (Table II). To demonstrate the functionality of the system to the participant, the experimenter programmed the task of picking up a sponge with one hand and passing it to the other hand. As part of this demonstration, all commands were exemplified and the functionality of the GUI and the visualization were illustrated.

Next, the participant was then given a list of three tasks. The tasks were tasks 1, 5 and 7 from Table III, in that respective order. The participants were instructed to program the tasks in order. After the tasks were explained by the experimenter, the microphone sensitivity was adjusted for each participant, the camera was turned on and the experimenter moved away to a different part of the room. The participants were instructed to work on a task for as long as they liked, until they were satisfied with the robot’s performance, and then demonstrate the task execution to the experimenter.

C. Metrics

The experiments were recorded from a video camera overseeing the experiment area. The recordings were used to measure the success of all programmed actions when executed by the participant and the time spent on programming each action. The generalizability of programmed actions was measured through tests performed later by one of the authors, on five different scenarios similar to those used in the systematic testing (Sec. IV). As part of the exit survey, the participants were asked to rate the robot performance on a scale from

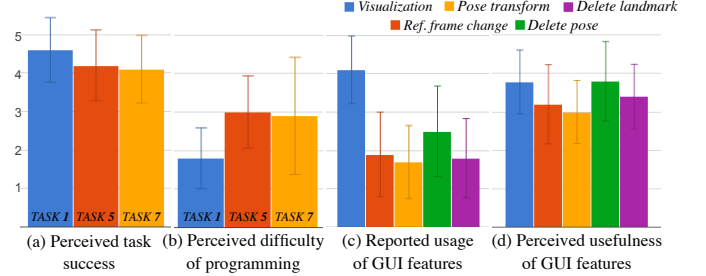


Fig. 6. Summary of questionnaire responses.

1 (complete failure) to 5 (perfect success) as well as their perceived difficulty of programming each task from 1 (very easy) to 5 (very hard). The survey also measured the cognitive load index using the NASA-TLX questionnaire. Finally, participants were asked questions about their usage of and their perceived usefulness of different components of the system: they were asked how often they used each feature, with the options being 1 (never), 2 (once), 3 (2-5 times), 4 (5-10 times) and 5 (more than 10 times), and how useful they thought each feature was, with the options being 1 (extremely useful), 2 (useful), 3 (somewhat useful), 4 (almost not useful) and 5 (not at all useful).

D. Results

Our user study was completed by 10 participants (6 males, 4 females, age range 19–26) recruited from a university campus community. Descriptive statistics from this study together with observations and usage characteristics are given in the following.

100% success: All participants in our study were able to program and successfully execute all three tasks. Most of the participants required more than one attempt to program all tasks, starting over once or several times by deleting all poses or creating a new action. However, three participants successfully programmed the first task (Task 1) on their first attempt, and two participants programmed the second task (Task 5) on their first attempt.

Generalization: Table IV shows the success rate of actions programmed by participants in five novel scenarios, together with the generalization performance obtained in the systematic tests (Sec. IV). We observe that the average generalization performance obtained by the participants is on par with that of an expert user of the system for Tasks 1 and Task 7. Some participants were able to program actions that succeeded in all five test conditions for these two tasks, exceeding the expert users’ performance.

The most common reason for poor generalization was observed to be false relativeness to objects. For instance, in Task 5, the actions programmed by participants 4 and 5 included placement poses that were relative to the initial pose of the plate. As a result, when the initial plate configuration was changed, the plate would not be placed at the desired absolute pose, even if it was successfully grasped. This error is fixable in the GUI, but the participants were likely not aware of it as they only tested the action in one scenario. This was partly

because participants were asked to program actions to their satisfaction and were not *explicitly* told to program actions that would generalize to novel scenarios.

Other problems that occurred in the generalization test were similar to those encountered in the systematic evaluation, i.e., predominantly due to limitations in perception. The overall generalization performance by participants was worst in Task 5. Most likely reasons for this are (i) the difficulty of coordinating two arms, and (ii) the error sensitivity of the initial contact point with the plate (when pushing to constrain it).

Learnability: Table V shows the time taken to program each task. While we do not observe a consistent pattern across all participants, the data suggests a learning effect: some people spent more on the first task than on the second or third one, even though the first task was the easiest amongst all three. It is also worth noting that while some people spent much more than the average amount of time on the second task (Task 5), and others on the third task (Task 7), that was mutually exclusive. There were no participants that spent a lot of time on both of them. This suggests that most of the time is spent learning to interact with the robot and with the system. In addition we observe that the time spent by novices is significantly larger than the time spent by one of the authors on teaching the three tasks (Table V). This demonstrates that our system allows for experienced users to further increase their efficiency in programming new tasks.

Six participants finished programming all three tasks in less than 45 minutes from their arrival. Four of these participants chose to program an additional task. Two chose to program unscrewing the lid off of a jar (Task 10) from the benchmark, one chose to program cup stacking (Task 6), and one chose to program an assembly task of connecting two building blocks together. All four participants (participants 1, 8, 5 and 6) were successful in programming their task of choice, spending 9:00, 11:50, 5:30 and 4:30 minutes respectively. This provides additional evidence for the learning effect, since the tasks chosen by the participants are arguably more difficult than the tasks offered in the user study, yet the participants spent about the same amount of time or less on these tasks. Moreover, the fact that even novice users are able to program tasks from the benchmark list strengthens the systematic evaluation result.

Varied user experiences: Figure 6(a-b) shows the survey responses for perceived success and difficulty of the tasks that participants programmed. The first task was perceived as the easiest one and was thought to be most successful. The other tasks were perceived as harder, but the perceived success was still very high. This suggests that the participants' perceived task difficulty were consistent with our expectation and that the perceived difficulty did not prevent users from programming the tasks to their own satisfaction.

Use of system components: We assessed the frequency of use and the usefulness of visualization and editing functionality described in Sec. II-C and III-D. The results are presented in Figure 6 (c-d). The visualization was used most by all participants and was deemed very useful. The second most used feature was selectively deleting poses in the GUI. The

TABLE IV
GENERALIZABILITY OF ACTIONS PROGRAMMED BY PARTICIPANTS
(SUCCESSFUL EXECUTIONS OUT OF 5 TESTS).

	Task 1	Task 5	Task 7
Participant 1	5	5	4
Participant 2	5	2	5
Participant 3	1	3	5
Participant 4	3	1	5
Participant 5	1	1	3
Participant 6	5	4	5
Participant 7	5	5	3
Participant 8	5	4	4
Participant 9	5	3	1
Participant 10	5	0	4
<i>Average</i>	4	2.8	3.9
<i>St.dev.</i>	1.70	1.75	1.29
Expert	4	5	4

TABLE V
TASK COMPLETION TIMES (MM:SS).

	Task 1	Task 5	Task 7	Total
Participant 1	03:10	05:35	06:10	14:55
Participant 2	03:25	18:50	12:05	34:20
Participant 3	05:00	07:20	18:15	30:35
Participant 4	10:00	35:30	09:50	55:20
Participant 5	09:45	04:40	17:30	31:55
Participant 6	12:30	10:20	10:20	33:10
Participant 7	04:40	33:50	04:50	43:20
Participant 8	02:30	03:50	12:40	19:00
Participant 9	07:50	09:50	07:05	24:45
Participant 10	05:20	08:10	07:00	20:30
<i>Average</i>	06:25	13:47	10:34	30:47
<i>St.dev.</i>	03:24	11:46	04:36	12:04
Expert	1:20	3:05	4:50	9:15

other features were used less, but were still considered useful. This suggests that the contribution of the action visualization was larger than that of the editing functionality in enabling the programming of useful actions. In other words, the visualization helped users get a clear mental model of how the actions were represented (e.g., what the robot could see on the table, when poses ended up being relative to these objects, etc.) and they provided demonstrations that created correct actions rather than requiring post-editing (e.g., if they intended a pose to be absolute, they made sure to move the end-effector sufficiently far from all objects on the table).

VI. DISCUSSION

A. Relation to Previous Work

Our approach is inspired by the framework proposed by Niekum *et al.* [12]; however, it differs in two fundamental ways: (i) rather than automatically segmenting a continuous action demonstration, we directly solicit segmented demonstrations in the form of sparse targets (e.g., as in [1]), and (ii) rather than discovering the frame of reference for segments from invariances in multiple demonstrations, we use a simple heuristic (proximity) to assign frames of reference to segments.

Our action representation has two key expressivity limitations compared to existing approaches. Representing action segments with a single keyframe, rather than a continuous

trajectory as in [12], limits the types of movements that can be expressed. For instance, smooth curved movements, such as scrubbing a surface with a brush, are difficult to capture with a series of sparse poses. Second, representing each segment of the action as a single target pose, rather than learning a distribution over possible target poses from multiple demonstrations, limits the generalizability of programmed actions. Probabilistic targets, as in [1], would give the robot alternative ways of executing an action in a given scenario. Nonetheless, our evaluation (Sec. IV) demonstrates that our simple action representation has sufficient expressivity to capture a diverse set of object manipulation actions and generalizes reasonably well to different initial configurations of objects.

The interaction paradigm proposed in this paper is comparable to Cakmak & Thomaz’s *active PbD* approach of collecting few demonstrations from the end-user, and then having the robot generate questions to gather more information [4]. While answering questions might be more intuitive for end-users initially, the information bandwidth of verbal questions are limited. Our approach provides a more efficient way of getting information from the user by exploiting rich interactions with a graphical interface. Additionally, in our approach, users decide what information to provide, as opposed to letting the robot decide what information they should provide. Note that our approach requires actions to be visualizable, while active PbD requires actions to be amenable to different question types. Hence, the choice of interaction paradigm in PbD strongly depends on the choice of action representation.

Our approach also has similarities with typical lead-by-the-nose interfaces provided with many industrial robots, which allow users to kinesthetically demonstrate desired robot arm poses or paths; however, these demonstrations result in open-loop, joint-angle trajectories that do not allow for generalization. More similar is the teaching interface for Rethink Robotics’ Baxter Robot, which allows people to kinesthetically demonstrate overhead pick-and-place trajectories relative to objects perceived by the robot.

B. Limitations and Possible Extensions

Next, we highlight some of the limitations of our representation, system, and interface, and we propose extensions that will provide greater generalizability and usability.

1) *Relaxed relativity*: In developing the benchmark task list we noticed that certain manipulation tasks that naturally occur in human environments were not captured by our representation. An example is *pushing a book to the edge of the table*. Our framework requires pushing to a fully specified location whereas this task only specifies one dimension of the target location. Our framework could be extended to have such relaxed specifications of action steps.

2) *Combining actions in simple programs*: Our system required explicitly repeating similar tasks; for instance, for stacking three cups, the user had to demonstrate the stacking of two cups one by one. Also the action is only applicable when there are three cups. Instead our framework could be extended to allow simple high-level programs that sequence

demonstrated actions or nest them in *while* or *do-until* loops. This way, the user could demonstrate stacking of one cup and then have it executed with an arbitrary number of cups until the action is no longer applicable (all cups are stacked).

3) *Richer landmark descriptors*: The most significant limitation of our system was the landmark perception. More sophisticated perception algorithms could allow greater generalization. For instance, a handle detector would allow programming manipulation actions that can be used for all objects that have similar handles. A corner detector would allow programming of folding non-rigid clothes. A symmetry detector (together with *relaxed relativity* (Sec. VI-B1)) would allow actions on symmetric objects to be performed in more situations.

4) *Automatic arm switching*: Our implementation did not exploit the fact that the robot’s two arms and grippers are the same. A simple extension will allow executing actions programmed with one arm using the other arm or switching the roles of the two arms in bi-manual actions, allowing greater generalization.

5) *User interface*: The most common feedback provided by participants regarding problems with the user interface was about the difficulty of interacting with co-located poses in the 3D environment. This could be addressed by providing an alternative way of interacting with action steps, such as a list or filmstrip view of the action steps. This would allow easy selection of the step to be edited. Second, participants thought that the progression of the task (i.e. the ordering of steps) was not elicited well in the 3D visualization. This could be improved by varying certain features of the visualization (e.g. color) over the progression of the action.

VII. CONCLUSION

We propose a novel Programming by Demonstration framework that involves initializing actions with a single demonstration and then providing additional task information explicitly, through interactions with a visualization of the learned action. This is in contrast with most existing techniques that require multiple, often many, demonstrations to capture the additional task information. We propose a simple yet expressive action representation that allows learning generalizable actions with this approach. The simplicity of the representation supports its visualization and interactive editing by end-users. We described an implementation of our framework on a PR2 robot and demonstrated its expressive power by evaluating it on a diverse task benchmark that involves 12 manipulation tasks with everyday objects. We further presented a user study (N=10) that demonstrates the ease of learning and intuitiveness of our system for novice users. Although our approach has limitations, we have shown that a large number of everyday tasks can be programmed using our framework. Therefore, we believe that the PbD framework presented has great potential for allowing ordinary users to teach future household, service, and industrial robots to perform useful tasks.

REFERENCES

- [1] B. Akgun, M. Cakmak, J. Yoo, and A.L. Thomaz. Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2012.
- [2] Brenna Argall, Sonia Chernova, Manuela M. Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.
- [3] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot Programming by Demonstration. In *Handbook of Robotics*, chapter 59. 2008.
- [4] M. Cakmak and A. L. Thomaz. Designing robot learners that ask good questions. In *Proc. of the Intl. Conf. on Human-Robot Interaction (HRI)*, 2012.
- [5] S. Calinon and A. Billard. What is the teacher’s role in robot programming by demonstration? Toward benchmarks for improved learning. *Interaction Studies.*, 8(3), 2007.
- [6] S. Calinon and A. Billard. Statistical learning by imitation of competing constraints in joint and task space. *Advanced Robotics*, 23(15):2059–2076, 2009.
- [7] A. Cypher. Eager: Programming repetitive tasks by example. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 33–39. ACM, 1991.
- [8] E. Gribovskaya and A. Billard. Learning nonlinear multi-variate motion dynamics for real- time position and orientation control of robotic manipulators. In *IEEE-RAS Intl. Conf. on Humanoid Robots*, 2009.
- [9] A. J. Ko, B.A. Myers, and H. Aung. Six learning barriers in end-user programming systems. In *Visual Languages and Human Centric Computing, 2004 IEEE Symposium on*, pages 199–206. IEEE, 2004.
- [10] M. Muhlig, M. Gienger, S. Hellbach, J.J. Steil, and C. Goerick. Task-level imitation learning using variance-based movement optimization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1177–1184, 2009.
- [11] M. Muhlig, M. Gienger, S. Hellbach, J.J. Steil, and C. Goerick. Automatic selection of task spaces for imitation learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009.
- [12] S. Niekum, S. Osentoski, G. Konidaris, and A.G. Barto. Learning and generalization of complex tasks from unstructured demonstrations. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2012.
- [13] S. Niekum, S. Osentoski, S. Chitta, B. Marthi, and A.G. Barto. Incremental semantically grounded learning from demonstration. In *Robotics: Science and Systems*, 2013.
- [14] M. Phillips, Vi. Hwang, S. Chitta, and M. Likhachev. Learning to plan for constrained manipulation from demonstrations. In *Robotics: Science and Systems*, 2013.
- [15] 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.
- [16] H.B. Suay, R. Toris, and S. Chernova. A practical comparison of three robot learning from demonstration algorithms. *Intl. Journal of Social Robotics, special issue on LfD*, 4(4), 2012.
- [17] J. Wong and J.I. Hong. Making mashups with marmite: towards end-user programming for the web. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1435–1444. ACM, 2007.