Teaching People How to Teach Robots: The Effect of Instructional Materials and Dialog Design

Maya Cakmak
University of Washington
Dept. of Computer Science and Engineering
mcakmak@cs.washington.edu

Leila Takayama*
Willow Garage, Inc.
68 Willow Road, Menlo Park, CA 94025
takayama@google.com

ABSTRACT

Allowing end-users to harness the full capability of general purpose robots, requires giving them powerful tools. As the functionality of these tools increase, learning how to use them becomes more challenging. In this paper we investigate the use of instructional materials to support the learnability of a Programming by Demonstration tool. We develop a system that allows users to program complex manipulation skills on a two-armed robot through a spoken dialog interface and by physically moving the robot's arms. We present a user study (N=30) in which participants are left alone with the robot and a user manual, without any prior instructions on how to program the robot. Instead, they are asked to figure it out on their own. We investigate the effect of providing users with an additional written tutorial or an instructional video. We find that videos are most effective in training the user; however, this effect might be superficial and ultimately trial-and-error plays an important role in learning to program the robot. We also find that tutorials can be problematic when the interaction has uncertainty due to speech recognition errors. Overall, the user study demonstrates the effectiveness and learnability of the our system, while providing useful feedback about the dialog design.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics; H.1.2 [**Models and Principles**]: User/Machine Systems

Keywords

Programming by Demonstration, Spoken dialog systems

1. INTRODUCTION

General-purpose robots, such as mobile manipulators, are becoming increasingly accessible. Unlike single-purpose robots that are designed and pre-programmed to carry out a particular task, general-purpose robots offer the potential for end-users to program the robot for their unique purposes. This presents several interaction design challenges related to building tools that enable end-users to program new capabilities on their robots. It is crucial for such tools to give as much functionality to the user, while being easy to learn and not requiring in-person training of the end-user. A common method to allow end-users to program new capabilities on a robot is Programming by Demonstration (PbD). This involves demonstrating a desired capability to the robot, allowing it to model and reproduce the capability.

Figure 1: Participants in our user study, programming the PR2 to fold a towel by demonstration.

Demonstration is an intuitive way for users to communicate a desired capability. Nonetheless, details of the interaction through which users provide demonstrations might not be directly evident. For instance, even providing a single demonstration requires indicating the start and end of the demonstration, and there can be a number of ways to do that. Existing PbD systems employ vocal commands [2], pedals [18], and buttons on a remote controller [14] or on the robot's arm[1]. All of these methods require instructing users on what to do (*e.g.* which button to press for each functionality). As we start providing more functionalities to end-users (*e.g.* creating multiple actions, browsing actions, executing an action, deleting a previously programmed action, etcetera), the amount of instruction to be given to the users also increase. This makes it difficult for end-users to figure out, on their own, how they can program a robot.

In this work, we set out to design a PbD system for the PR2 robot and instructional materials which would let a naive user to program the robot without any training from an expert. We built a system that allows programming various manipulation skills by physically moving the robot's arms and talking to the robot through a spoken dialog in-

*This author is currently affiliated with Google[x]. The work presented in this paper was conducted while both authors were affiliated with Willow Garage, Inc.

HRI'14, March 3–6, 2014, Bielefeld, Germany.

http://dx.doi.org/10.1145/2559636.2559675 .

[1] `http://www.rethinkrobotics.com/products/baxter/`

terface (Fig. 1). As instructional materials, we explore the use of a step-by-step tutorial and an instructional video, in addition to a user-manual that specifies all the available commands that can be used. We examine the effects of these in a user study (N=30) which demonstrates superior performance when users are trained with a video and highlights the importance of trial-and-error. Based on the data from this user study, we also characterize the impact of certain design choices related to the dialog system.

2. RELATED WORK

Programming by Demonstration (PbD), also known as Learning from Demonstration, has been studied within robotics for the last three decades [4], with recent work focusing more and more on human interaction problems in PbD [5, 2, 19, 12, 21, 15]. The design choices for the dialog interface of our PbD system are influenced by work on spoken interactions with robots within the human-robot interaction (HRI) literature [7, 8, 21]. Work in the field of human-computer interaction which investigate how the system feedback influences human input [9, 16, 6] also have implications for human-robot dialog, and have been considered in the design of our system.

Our work is also influenced by a long line of research on the role of mental models in learning to operate new devices [11] and methods for effective instructional design for learning complex tasks [20, 1]. Particularly relevant work in this area is by Kamm *et al.* who investigate the influence of tutorials on user expertise with a spoken dialogue system for checking e-mail [10]. In their study, the tutorial significantly reduced task completion times and increased user satisfaction ratings. Although the design of instructional materials have not been explicitly studied in the context of HRI, they are largely employed for user studies evaluating functional systems designed for humans. For instance, work by Nguyen *et al.* employed tutorials to teach RCommander [13]—a tool for creating behaviors for a domestic robot.

3. SYSTEM

3.1 Platform

The robot platform used in this work is PR2 (Personal Robot 2) which is a mobile manipulator with two 7 Degree-of-Freedom (DoF) arms and an omnidirectional base. The passive spring counterbalance system in PR2's arms makes them naturally gravity-compensated, giving users the ability to kinesthetically move the arm within its kinematic range. Each arm has a 1 DoF under-actuated gripper and can carry up to 2.2kg. PR2 has a pan-tilt head with a Kinect sensor mounted on top.

The software written for this work is developed within ROS (Robot Operating System) and was released as an open-source package[2]. Speech recognition for the dialog system is done with Pocketsphinx, using a Shure wireless microphone headset. For text-to-speech on the robot we use Cepstral, with the voice *David*.

3.2 One-shot Keyframe-based Programming by Demonstration

The PbD system presented in this work is based on the keyframe-based PbD framework proposed by Akgun *et al.* [2].

[2]`http://ros.org/wiki/pr2_pbd`

Table 1: Components of the system state.

Dialog state	$d \in \{start, programming, execution\}$
Robot state	Joint configurations ζ^R, ζ^L, gripper states $g^R, g^L \in \{0,1\}$ and arm stiffnesses $\alpha^R, \alpha^L \in \{0,1\}$
Experiment state	Number of created skills N, current skill index $n \in \{1..N\}$, skills programmed so far $\{\mathcal{S}_i\}_{i=1}^{N}$ where $\mathcal{S}_i = \{(\zeta^R, g^R, \zeta^L, g^L)_k : k = 1..K_i\}$ and last used command c_{t-1}

We represent a *skill* as a sequence of states that the robot needs to go through: $\mathcal{S} = \{(\zeta^R, g^R, \zeta^L, g^L)_k : k = 1..K\}$, where ζ refers to the robot's 7-DoF arm configurations[3], and g refers to the binary gripper state (open or closed). The superscripts R and L denote right or left end-effectors. Skills are programmed directly with a single demonstration. In other words, the demonstration itself is a sequence of joint states that is used for reproducing the skill. For demonstrations, joint states are manipulated kinesthetically by the user, *i.e.* by physically moving the robot's arms.

To reproduce the skill, the robot moves through recorded states. It moves from one state to the next by first moving both arms with a certain velocity profile and and then changing the gripper states. For instance, if the gripper is closed at step $k-1$ and open at k, the robot will first move to $(\zeta^R, \zeta^L)_k$ with a closed gripper, and then open the gripper. The duration of the movement is determined by the arm that needs to move more. Speeds are adjusted such that both arms reach the next state at the same time.

3.3 Dialog System

The user interaction with the PbD system is done through a simple state-based dialog system with a finite set of input commands. The response to each command differs based on the *system state*. This is the combination of the *dialog*, *robot* and *experiment* states (detailed in Table 1). There are three possible dialog states: *start*, *programming*, and *execution*. The robot state involves the end-effector poses, gripper states (0:closed, 1:open) and arm stiffnesses (0:relaxed, 1:stiff). The experiment state involves the set of skills that have been programmed so far, and the index of the current skill. There are nine different command types (16 unique commands). The commands are listed in Table 2, excluding the two commands TEST MICROPHONE (which has no effect on the system state) and UNDO LAST COMMAND (which reverses the effect of the previous command).

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, when the command OPEN RIGHT HAND is used, the robot says "Opening right hand" while the right gripper opens and the robot glances towards the right gripper.

[3]Our system actually represents arm states with the 6-DoF end-effector states and uses the demonstrated 7-DoF arm state to seed the Inverse Kinematics solver. Within the context of this paper, this is equivalent to directly using joint configurations to represent arm states.

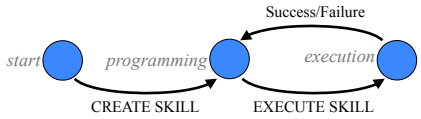

Figure 2: The core finite state-machine for the dialog system.

Table 2: Effect of the commands on the system state. Initialized with $d \leftarrow$ *start*, $N \leftarrow 0$ and $n \leftarrow 0$.

Command (c_t)	Effect of command
RELEASE/HOLD RIGHT/LEFT ARM	$\alpha^{R/L} \leftarrow \neg\alpha^{R/L}$
OPEN/CLOSE RIGHT/LEFT HAND	$g^{R/L} \leftarrow \neg g^{R/L}$
CREATE SKILL	$d \leftarrow$ *programming* $N \leftarrow N+1$, $n \leftarrow N$, $S_n \leftarrow \{\}$
SAVE POSE	**if** (d=*programming*): $S_n \leftarrow S_n \cup (\zeta^R, g^R, \zeta^L, g^L)$
EXECUTE SKILL	**if** (d=*programming*) & $(K_n > 1)$: $d \leftarrow$ *execution*,
CLEAR SKILL	**if** (d=*programming*): $S_n \leftarrow \{\}$
NEXT/PREVIOUS SKILL	**if** ($n > 1$ and $n < N$): $n \leftarrow n \pm 1$

The interaction begins in the *start* dialog state, where the user is allowed to test the microphone, change the stiffness of the robot's arms (released or holding a pose) and change the state of its grippers (open or closed). In the *start* state, the robot's response to the rest of the commands is the utterance "No skills created yet". Similarly, the robot has a speech response in every possible error case. Changes in the system state triggered by the commands are summarized in Table 2.

Creating the first skill moves the dialog state to *programming*. In this state the user can save poses into the current skill, delete poses, create more skills, navigate between the created skills, and trigger executions of the current skill. The commands that change the state of the robot have the same effects as in the *start* state. Once in the *programming* state, the dialog never goes back to the *start* state. The EXECUTE SKILL command, triggers a transition to the *execution* state, which involves moving through the poses of the current skill. In this state, the robot does not respond to any commands. The dialog returns to the *programming* state when the execution is over. The transitions between the dialog states is illustrated in Fig. 2.

4. INSTRUCTIONAL MATERIALS

Our goal in this work is to allow naive users to program the PR2 on their own, without any training from an expert. The complexity of the robot platform and the PbD functionality studied in this work, makes it challenging to accomplish this purely through interface design. We cannot expect participants to know the functionality and guess the right commands to use. To address this challenge we turn to *supplementary materials*, which are modalities outside the interaction that communicate information about the interaction. These are supplementary in the sense that they are not needed for the interaction; *i.e.* an expert user does not use them to program the robot. However, they can have an important role in the interactions of novice users.

Table 3: Sample command descriptions as worded in the user manual given to participants.

Command	Description in user manual
RELEASE/HOLD R/L ARM	Use these commands to release the robots arms so you can move them around, or to make them hold a certain pose.
CREATE SKILL	Use this command to create a new skill. PR2 will indicate the name of the skill (for example "skill-1") in its response.
CLEAR SKILL	Use this command to delete all the poses and hand actions that have been saved into the skill so far.

In this paper we explore the use of three types of supplementary materials: user manuals, written tutorials and instructional videos. We refer to the latter two also as *instructional* materials, as these are designed with pedagogical intent. Tutorials aim at allowing users to *learn by doing* [17], whereas videos support them to *learn by observing* [3]. In our user study, described later in Sec. 5, we compare these learning paradigms with a baseline in which only a user manual is provided, forcing them to *learn by exploring*, through trial-and-error. We describe the different supplementary materials designed for our the PbD system in the following.

User manual. A user manual (or user guide), is an extensive technical document that assists the user of a particular system—most commonly consumer electronics or computer software. It outlines the different functionalities of the system and provides instructions on how to use them referring to controls (*e.g.* buttons, menus) and states (*e.g.* lights, status bars) that are available to the user.

In this work the user manual communicates the allowed commands in the spoken dialog and explains their function. The manual includes an introductory paragraph that summarizes what the user can do through the interaction, and a table that contains the list of commands and a description of its purpose and effect. Samples from these descriptions are given in Table 3. The whole user manual is one page.

Written tutorial. A tutorial is a set of step-by-step instructions to complete a task. It is intended to teach the use of a certain system *by example*. In some cases the intended task is completed at the end of the tutorial and could be repeated in the future without the tutorial (*e.g.* how to change the strings of a guitar). In other cases, the tutorial involves a special case of a general task that is not exactly the intended task, and it can be transferred to other instantiations of the task (*e.g.* how to write a ROS service).

The tutorial for our system consists of step-by-step instructions to allow the user to program a set of skills, so as to illustrate the effect of the different commands allowed in the dialog. The steps of the tutorial and the commands practiced in each step are given in Table 4. As an example,

Table 4: Steps of the tutorial for teaching the use of the PbD system.

Tutorial step	Commands practiced
Getting started	TEST MICROPHONE
Moving the arms	RELEASE/HOLD R/L ARM
Using hand actions	OPEN/CLOSE R/L HAND
Programming a skill: Waving	CREATE SKILL, SAVE POSE, EXECUTE SKILL
Adding a hand action into the skill	OPEN/CLOSE R/L HAND, SAVE POSE, EXECUTE SKILL
Deleting a pose and clearing a skill	UNDO LAST COMMAND, CLEAR SKILL
Navigating skills	CREATE SKILL, PREVIOUS SKILL, NEXT SKILL

the wording of the fourth step in the tutorial is as follows:

1. Say CREATE SKILL and listen to PR2's response.
2. Release PR2's right arm and move it to a waving pose. Say SAVE POSE while holding the arm in place.
3. Move the arm to a different pose to the right of the first pose. Say SAVE POSE while holding the arm in place.
4. Save a third pose slightly to the left of the first pose.
5. Let PR2's arm go and say EXECUTE SKILL. Observe the skill playing out.

Instructional video. An instructional video is a video that instructs the user on how to complete a task by demonstrating it. It allows the user to learn a task by observing someone else do the same task. As with tutorials, users may execute the task themselves as they watch the video step-by-step. Note that videos may not have been made with the intent of teaching, but may still serve this purpose (*e.g.* learning how to play a song on the guitar from a video of someone performing it; learning about the basic functionality of Siri from the iPhone commercial). The instructional video for our PbD system involves a person executing the steps of the tutorial given in Table 4[4]. The length of the video is 3 minutes 36 seconds.

5. EVALUATION

We evaluated the effectiveness of our system (Sec. 3) and the impact of the different instructional materials (Sec. 4) through a user study which we describe in this section.

5.1 Study Design

Our study has three conditions in which participants were given a different combination of supplementary materials.

1. *Baseline:* Participant is only provided with the user manual.
2. *Tutorial:* Participant is provided with the written tutorial in addition to the user manual.
3. *Video:* Participant is provided with the instructional video in addition to the user manual.

[4]`http://www.youtube.com/watch?v=Sou_rthgCtE`

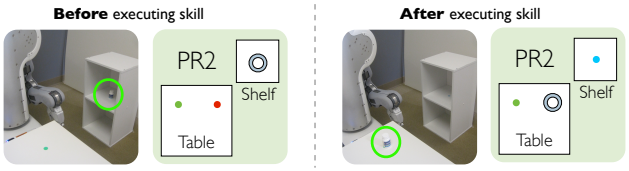

Figure 3: An example skill depiction (*constrained pick-up and place*) from the *skill guide* handed to the participants in the experiment.

We used a between-groups design; *i.e.* each participant was assigned to one of the three conditions. Participants in all conditions were provided with the user manual since a user cannot be expected to know the command set for the dialog. In addition, participants in all conditions were allowed to call the experimenter to ask questions. This is equivalent to calling a *technical support* line to get help on using a product. This was done to measure the occurrence of blockages in the task where the participants felt that the instructional materials were insufficient, while also allowing them to overcome these blockages.

Participants were asked to program four different skills on the PR2. The skills were visually illustrated and explained on the *skill guide* (Fig. 3) which was handed to the participants together with the supplementary materials. The description of the four skills are as follows:

1. Pick-up and place: Pick up the pill bottle from the red dot on the table and place it at the green dot on the table, using the left arm.
2. Constrained pick-up and place: Pick up the pill bottle from the blue dot on the shelf and place it at the red dot on the table, using the left arm (Fig. 3).
3. Pick-up, transfer, and place: Pick up a pill bottle from the green dot on the table with the right hand, transfer it to the left hand, and place it at the red dot on the table.
4. Towel folding: Fold a towel placed on the table (with two corners on the red and green dots) into two.

The second and third skills are progressively more challenging versions of the first one, involving obstacles and coordination of two arms. The last is a transfer task which involves programming a skill with similar constraints as the third skill in a different context. The order of the tasks were maintained across participants.

5.2 Procedure

Participants were scheduled ahead of time for one hour time slots. Upon arrival, they were brought to the experiment area and given an informed consent form to sign. The experimenter briefly stated the long-term goal of the research, and told the participants that their task will be to program new skills on the PR2. The experimenter said that she will not give any instructions on how to program PR2, but that she will provide supplementary materials allowing the participants to figure it out on their own. Next, the experimenter handed the skill guide and went over the four skills by demonstrating them inside the robot's workspace. Depending on the condition, the instructional materials were handed to the participants and explained. In the *video* condition the video was made ready to play on a computer screen inside the experiment area. Participants were told to

imagine that they just bought a PR2 knowing that you can program it, and it came out of the box with the given supplementary material. They were told that if they feel they are stuck, they can request tech support by calling the experimenter. Finally, the participant was equipped with the microphone and the experimenter left the experiment area.

The experimenter waited outside the experiment area and responded to tech support requests. If the participant completed all four task within 40 minutes past their arrival, they were told that they have time to program one additional skill of their choice. When done with programming participants were administered a browser-based survey.

5.3 Evaluation Metrics

The experiments were recorded from a video camera overseeing the experiment area. In these recordings, all utterances by the participant directed to the robot were transcribed by two independent coders and categorized as one of the commands or as a wrong command. In addition the recordings were used for measuring (i) success of four skills, (ii) time spent on programming each skill, and (iii) the number of tech support requests.

The exit survey involved five parts. The first part measured the cognitive load index using the NASA-TLX questionnaire, and asked two additional questions to assess the perceived success of each skill they programed, and their difficulty. The second part asked the participant to specify how much the supplementary materials and trial-and-error contributed to their understanding of the different commands. The third part involved questions about the commands and asked the participants to rate their agreement with the following statements about the user manual:

Overall usage	I used the user manual extensively.
Introduction	I carefully read the introductory paragraph of the user manual.
Commands	I carefully read the descriptions of the speech commands.

The fourth part (automatically skipped in the *baseline* condition) asked participants to rate their agreement with statements related to their usage of the tutorial or the video.

Completion	I completed the whole video/tutorial.
Usefulness	The video/tutorial was useful in giving me an understanding of the speech commands.
Redundancy	Parts of the video/tutorial were redundant.
Completeness	The video/tutorial made the user manual unnecessary.

The last part of the survey collected demographic information and information on habits related to technology usage and instructional materials.

6. RESULTS

Our study was completed by 30 participants (15 female, 15 male, gender-balanced across three conditions) in the age range of 19 to 70 (M=39.57, SD=15.53). In the following we present the main findings of the user study grouped in terms of observations relating to the impact of the instructional materials and the design choices of the PbD system used in the study.

6.1 Impact of instructional materials

We first analyze the impact of the instructional materials on the interaction with the robot. We make the following observations.

Video is most effective. Fig. 4(b) shows the number of participants who successfully programmed the four different skills in each condition. For each skill, more participants were successful in the *video* condition than in the other conditions. Participants in this condition are more successful particularly in programming the first two skills. The time spent programming the first skill was significantly reduced by the instructional video as compared to the baseline ($t(18)$=3.42, p<.05)(Fig. 4(a)). In addition, the overall time for programming all skills was smallest in the *video* condition (M=25.03 minutes, SD=5.04), although not significantly less than in the *baseline* (M=30.95, SD=7.65) or *tutorial* (M=29.63, SD=12.26) conditions. Thus, the video allowed people to be both more successful and efficient in programming the skills on the robot.

The time that participants spent on the tutorial versus the video prior to programming were about the same (Fig. 4(a)). This was on average longer than the duration of the video, because people either paused the video at times or they watched parts of or the whole video twice. On the contrary, participants tended not to complete the tutorial. Only two out of the 10 participants in this condition executed the tutorial until the last step, whereas all 10 participants in the *video* condition watched the whole video at least once. This was reflected in the survey question that asked participants to rate their agreement with *"I completed the whole tutorial/video."* The rating was significantly higher in the *video* condition (χ^2=-3.06, p<.005 in a Kruskal-Wallis test). This means that people got more information out of the video, even though they spent equal time on the tutorial.

The effectiveness of the video was also reflected on the number of tech support requests, which were significantly fewer than in the baseline ($t(18)$=8.49, p<.01). Only one participant in the *video* condition made a tech support request, as compared to eight in the *baseline* and four in the *tutorial* condition, where the average number of requests were 2.38 (SD=1.85) and 1.50 (SD=0.58) respectively. We note that the questions asked by participants during tech support did not provide them any information that was not in the user manual. By watching the whole video, participants were exposed to the available functionality even if they might not remember the exact commands or understand exactly how they work.

The usage of UNDO LAST COMMAND is a clear example of the increased awareness about available functionality due to the instructional video. This command allows the user to easily recover from speech recognition errors which were frequent in our experiment. We observed that participants in the *video* condition used this command more frequently (M=4.20, SD=3.79) than those in the *baseline* (M=1.80, SD=1.40) or *tutorial* (M=1.80, SD=2.10) conditions. Part of the reason for the ignorance about the undo command was that it was presented last in the user manual and it was practiced towards the end of the tutorial, which most participants did not get to. Another reason, we believe, was that there were other ways of recovering from errors, al-

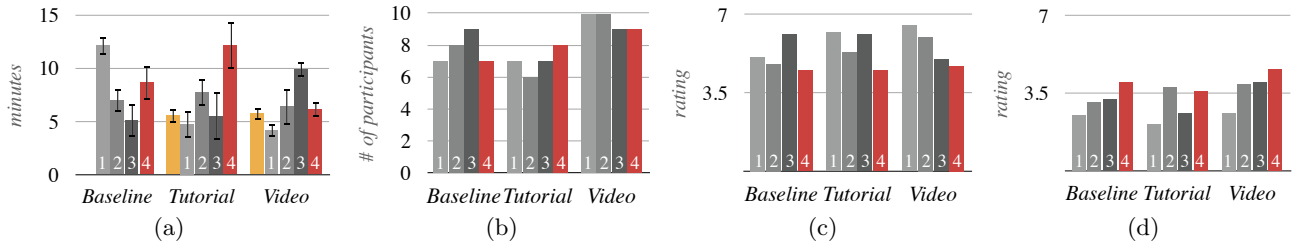

Figure 4: (a) Average time participants spent on the instructional materials and on programming each skill. (b) Number of participants who successfully programmed the four skills in each condition. Participants' average rating of the (c) success and (d) difficulty of the four skills they programmed in each condition.

though these were much less efficient than just undoing the error. For example, when participants accidentally deleted all the poses they had saved so far (a false positive for the CLEAR SKILL command) they would start from scratch, often showing signs of frustration. The participants' increased awareness about this functionally in the *video* condition may have contributed to their efficiency.

We also observed anecdotal examples illustrating how ignorance about the functionality leads to inefficiency. One participant made the arm stiff every time he wanted to save a pose—he gave three commands (HOLD R/L ARM – SAVE POSE – RELEASE R/L ARM) instead of one for each pose, and as a result progressed very slowly.

Tutorials can be problematic. We observed that tutorials were unlikely to be followed until the end (Fig. 5(c)). This could be partly because people felt it was unnecessary and wanted to directly move on to programming the actual skills rather than programming a practice skill (waving) as part of the tutorial. Participants in the *tutorial* condition agreed more strongly that the tutorial had redundant information (Fig. 5(c)), than the participants in the *video* condition did for the same statement regarding the video, although this difference was not significant.

In addition, speech recognition errors were more problematic in the *tutorial* condition than they were in other conditions. The tutorial has a certain progression that assumes errorless speech recognition. Participant are likely to continue following the step-by-step instructions as they appear in the tutorial, even though the robot might not be in the state that it needs to be at each step. They are also less likely to pay attention to the robot's response as they follow these instructions. In our experiment this led to undesirable consequences like participants trying to move the arm while it was stiff or saving poses before successfully creating a skill (which ended up not being saved). Participants who experienced these problems had worse performance than those who did not, resulting in a bi-modal distribution in the *tutorial* condition. This is reflected in the inconsistent progress and the large performance variance observed in this condition (Fig. 4(a)). Nonetheless, we observe some positive effects of the tutorial: in comparison to the *baseline* it significantly reduced the time spent programming the first skill ($t(18)$=3.16, p<.05) and had marginally fewer tech support interruptions ($t(18)$=4.26, p=.054).

Trial-and-error is essential. As mentioned earlier, programming the first skill took significantly longer in the *baseline* condition as compared to the experimental conditions. Participants in this condition learned and practiced the different commands while programming the first skill. We observe that the time they put into exploring the functionality while programming this simple skill helps them as they move on to more challenging skills. The time they spent programming the subsequent skills is less and less, despite the fact that the skills get more and more challenging (Fig. 4(a)). This points to the importance of trial-and-error in learning to program the robot.

Both the tutorial and the video made participants efficient in programming the first skill, however this effect was lost in more challenging skills. We believe that this was due to the simplicity of the example used in the video and the tutorial (a waving action that involves a gripper state change). This example provides sufficient information for programming the first skill (simple pick up and place) without needing to understand the commands in depth. However, as the skill to be programmed becomes more challenging, the information gained from the video/tutorial becomes insufficient so people spend more time on trial-and-error and referring to the user manual.

Participants in the *baseline* condition stated that 45% of their understanding of the commands came from trial-and-error on average, while the remaining 55% came from the user manual (Fig. 5(a)). In the experimental conditions, the additional instructional material was perceived to have a significant contribution to the participants' understanding (tutorial: 43.5%, video: 49.5%), both more than the other factors that participants rely on in the baseline. We see that the video is complemented more by trial-and-error, whereas the user manual remains more dominant when the primary source of information is the tutorial. The survey indicated consistently higher usage of the user manual by participants in the baseline condition (Fig. 5(b)), however the difference was significant only for the statement regarding the introductory paragraph of the manual (χ^2=6.38, p<.05).

Uniform and consistent progress with video. The participants' rating of the difficulty of programming each skill was consistent with our intention—people generally felt that the later skills were more challenging in all conditions ($F(1,26)$=14.15, p<.01, Fig. 4(d)). Despite this perception, participants in the *baseline* condition spent most time on the easiest skill (Fig. 4(a)), were more successful in later skills (Fig. 4(b)), and rated the success of their later skills as higher (Fig. 4(c)). This pattern was inverted in the *video* condition, resulting in a more intuitive outcome—participants' perception of the skill difficulty was inversely correlated with their success and efficiency in programming the skill. In other words, participants made consistent and

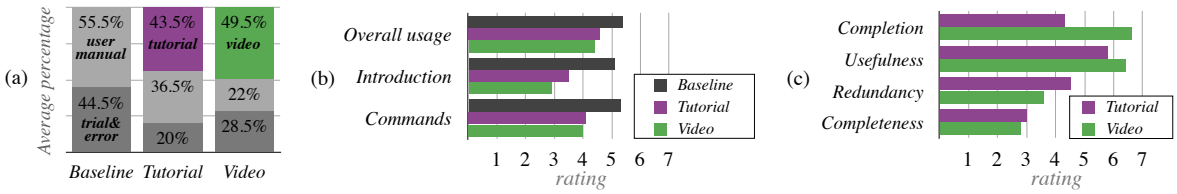

Figure 5: (a) The average contribution of the instructional materials and trial-and-error to the participants' understanding of the commands based on their self-assessment. Participants' self-reported usage of the (b) user manual in all three conditions and (c) tutorial or video in corresponding conditions.

uniform progress, learning more about the functionality as it was needed, as opposed to making most of the progress at the beginning. We believe that the slow progress at the beginning in the *baseline* may result in frustration that might be problematic in other contexts. In our experiment, this was manifested as an increased number of technical support requests.

6.2 System evaluation

The user study demonstrates the success of our system in enabling novice users to program new skills on a robot completely on their own. We saw that all participants in the *video* condition successfully programmed the first two skills and only one participant failed in the following two skills. Only one participant in this condition required technical support. Nine out of the 30 participants also had the opportunity to program a skill that they wanted—this included stacking bowls, high-five, assembling lego bricks, and putting a teddy bear to sleep. The system was also successful at capturing individual variations. Participants programmed various ways of folding a towel—grasping it from the middle versus the edge (Fig. 1), using the edge of the table to lay the towel versus using motion dynamics, *etcetera*.

The study also provided design feedback about different elements of the system interface, that are consistent with the literature and provide insights into how the system could be improved. Some observations relating to this point are given in the following[5]. Addressing some of these issues in the interface are important especially given our finding about the importance of trial-and-error in learning the functionality.

Appropriate feedback reduces learning load. We observed that speech responses by the robot to indicate error cases (*e.g.* "No skills after skill two", "Not enough poses in skill two") were helpful in letting participants know what to do next. For example out of the 14 participants who got the "No skills created yet" error, 10 responded with the command CREATE SKILL within the next ten seconds.

Inaccurate feedback can be problematic. Changes in the robot state in response to user commands may not be noticeable to novice users. For instance, in our experiment, participants did not initially know the difference between the released or holding arm stiffness. The speech response by the robot when this change happened was correct but not precise—it indicated the final state of the arms, without acknowledging whether the arm state had changed or not. It always responded to HOLD RIGHT ARM with "Right arm holding". This resulted in one participant to have an inaccurate understanding of the command. He assumed that the command told the robot to hold a certain pose, and he programed the first skill by forcefully pushing the arm, until he discovered the RELEASE R/L ARM command.

Distinctness of the lexicon is important. The most common invalid command error made by participants was to say "Release right/left hand" with the intent of opening the gripper. This mistake was made by 30 out of the 36 participants at least once. We believe that this was caused by the poor choice of the verb *release* for changing the arm stiffness. Opening the gripper while it has an object results in *releasing* the object. As a result, when participants wanted the robot to drop what it had in its gripper, they were compelled to say *release* instead of *open*.

Inconsistent feedback can be problematic. The second most common error among invalid commands was to use the name of the skill as part of the command; *e.g.* "Execute skill two", "Clear skill two." We believe this was due to the robot's use of the skill names in its feedback, *e.g.* "Starting execution of skill two", "Switched to skill two." The naming of the skill was essential in allowing the user to browse the skill through dialog, however it set the false expectation that the robot would understand name references to skills. This points to the importance of consistency in the input and output lexicon for the dialog.

7. LESSONS LEARNED

In this section we briefly reiterate the findings and observations from our user study in the form of design recommendations and provide examples of how this was applied to the revised design of the system after this user study.

1. **Show a video of the interaction.** Human-robot interactions, especially ones involving physical interactions, may be unique and completely novel to end-users. Showing a video of the intended interaction can efficiently convey the available functionalities and communicate details of the interaction that might not be evident to novices. For instance, in our study, one of the technical support requests in the *baseline* condition was to ask whether the participant was allowed to touch the robot; a video would have quickly mitigated this issue.

2. **User manuals should complement videos.** People are unlikely to read a user manual cover to cover. However videos are not well indexed for searching particular information on demand. Thus, a user manual can complement a video by allowing the user to easily browse information to find details about a functionality that they would have been exposed to through the video.

3. **Do tutorials in a sandbox.** When the interaction has high uncertainty, step-by-step tutorials should be avoided. The uncertainty can be reduced by making the

[5]Refer to video for examples, at `http://www.youtube.com/watch?v=NXZf_JjMAkQ`

robot aware of the tutorial step, or even letting the robot administer the tutorial. Another approach is to have an early step in the tutorial to expose the user to the uncertainty since this is something they eventually learn to deal with through trial-and-error.

4. **Give precise feedback.** The robot should not only indicate state changes, but also acknowledge the lack of state changes in response to commands. As suggested earlier, in our revised system the robot says "Right arm already holding" instead of "Right arm holding" in response to HOLD RIGHT ARM when the command has no effect.

5. **Handle errors with hints.** In response to commands that have no effect in the current state, it is useful to guide users towards states where the command would have an effect. In our system, the response "No skills created yet" was successful in getting participants to create a skill first.

6. **Choose vocabulary carefully.** Commands should not only capture the functionality and be intuitive individually, but also they should be distinct from one another. Commands with potential semantic overlap are likely to be confused since they would all be in short-term memory during the interaction. In the revised command set we have the verb RELAX for changing the arm stiffness, as RELEASE was often used incorrectly to try to open the gripper.

8. CONCLUSION

We present a Programming by Demonstration (PbD) system with a spoken dialog interface and investigate the use of instructional materials to support its learnability. The contributions of this paper are two-fold. We give empirical results regarding the impact of different instructional materials on learning how to program a robot and we present observations and recommendations regarding dialog interface design. Our findings have implications not only for PbD interactions, but also for any end-users interactions with complex robotic functionality. Second, we present a fully autonomous and robust PbD system that captures realistic manipulation tasks and has an intuitive user interface. Our study participants were able to use this system to program complex skills like folding a towel, without any instruction from an experimenter. The results of this experiment are informing the re-design of our PbD system.

9. REFERENCES

[1] S. Ainsworth. Deft: A conceptual framework for considering learning with multiple representations. *Learning and Instruction*, 16(3):183–198, 2006.

[2] B. Akgun, M. Cakmak, K. Jiang, and A. Thomaz. Keyframe-based learning from demonstration. *Journal of Social Robotics, Special issue on LfD*, 4(4), 2012.

[3] A. Bandura. *Social Learning Theory*. General Learning Corporation, 1971.

[4] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics*. Springer, 2007.

[5] 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.

[6] L. Dybkjær and N. O. Bernsen. Usability issues in spoken dialogue systems. *Natural Language Engineering*, 6(3&4):243–271, 2000.

[7] K. Fischer. How people talk with robots: Designing dialog to reduce user uncertainty. *AI Magazine*, 32(4):31–38, 2011.

[8] M. E. Foster, M. Giuliani, A. Isard, C. Matheson, J. Oberlander, and A. Knoll. Evaluating description and reference strategies in a cooperative human-robot dialogue system. In *IJCAI*, pages 1818–1823, 2009.

[9] J. Gustafson, A. Larsson, R. Carlson, and K. Hellman. How do system questions influence lexical choices in user answers? In *EUROSPEECH*, 1997.

[10] C. A. Kamm, D. J. Litman, and M. A. Walker. From novice to expert: the effect of tutorials on user expertise with spoken dialogue systems. In *ICSLP*, 1998.

[11] D. E. Kieras and S. Bovair. The role of a mental model in learning to operate a device. *Cognitive science*, 8(3):255–273, 1984.

[12] N. Koenig, L. Takayama, and M. Matarić. Communication and knowledge sharing in human–robot interaction and learning from demonstration. *Neural Networks*, 23(8), 2010.

[13] H. Nguyen, M. Ciocarlie, and K. Hsiao. Ros commander (rosco): Behavior creation for home robots. In *IEEE Intl. Conference on Robotics and Automation*, 2013.

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

[15] N. Otero, A. Alissandrakis, K. Dautenhahn, C. Nehaniv, D. Syrdal, and K. Koay. Human to robot demonstrations of routine home tasks: Exploring the role of the robot's feedback. In *Proc. of the Intl. Conf. on Human-Robot Interaction (HRI)*, 2008.

[16] J. Pearson, J. Hu, H. P. Branigan, M. J. Pickering, and C. I. Nass. Adaptive language behavior in hci: how expectations and beliefs about a system affect users' word choice. In *Proc. of the SIGCHI conference on Human Factors in computing systems*, 2006.

[17] R. C. Schank, T. R. Berman, and K. A. Macpherson. Learning by doing. In C. M. Reigeluth, editor, *Instructional-Design Theories and Models*. LEA, 1999.

[18] J. Schulman, A. Gupta, S. Venkatesan, M. Tayson-Frederick, and P. Abbeel. A case study of trajectory transfer through non-rigid registration for a simplified suturing scenario. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[19] H. 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.

[20] J. J. Van Merriënboer, P. A. Kirschner, and L. Kester. Taking the load off a learner's mind: Instructional design for complex learning. *Educational psychologist*, 38(1):5–13, 2003.

[21] A. Weiss, J. Igelsboeck, S. Calinon, A. Billard, and M. Tscheligi. Teaching a humanoid: A user study on learning by demonstration with hoap-3. In *IEEE RO-MAN*, pages 147–152, 2009.